Ein Framework für echtzeitfähige Ethernet-Netzwerke in der Automatisierungstechnik mit variabler Kompatibilität zu Standard-Ethernet

Vom Fachbereich Elektrotechnik und Informatik der Universität Siegen

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigte Dissertation

von

Frank Dopatka.

1. Gutachter:	Univ.-Prof. Dr. rer. nat. Roland Wismüller
2. Gutachter:	Univ.-Prof. Dr.-Ing. Günter Schröder
Vorsitzender:	Univ.-Prof. Dr. rer. nat. Udo Kelter
Datum der Prüfung:	12.09.2008

gedruckt auf alterungsbeständigem holz- und säurefreiem Papier

Kurzfassung

Im Umfeld der Automatisierungstechnik ist bereits seit einigen Jahren der Trend zu beobachten, etablierte Feldbusse durch echtzeitfähiges Ethernet zu ersetzen oder zu ergänzen. Das Ziel liegt darin, eine einzige Netzwerkinfrastruktur von der Leitebene im Bürobereich bis hin zu Feldgeräten wie Sensoren, Aktoren und Bedienpulten in industriellen Produktionshallen zu schaffen. Dieser Trend wird als vertikale Integration bezeichnet. Ein Hauptproblem liegt dabei darin, dass die weit verbreiteten Standard-Ethernet Netzwerke nicht den Anforderungen der industriellen Echtzeit genügen. Andererseits soll der Ethernet-Standard aber auch auf der Feldebene eingehalten werden, da ansonsten wiederum separate Feldbusse entstehen. Das Hauptziel dieser Arbeit besteht in der Entwicklung eines formal fundierten Frameworks, mit dessen Hilfe zwischen der Kompatibilität zum verbreiteten Standard-Ethernet und der Einhaltung von konkreten Echtzeitanforderungen einer automatisierten Anlage variiert werden kann.

Dabei erfolgt zunächst eine Übersicht und Klassifizierung der existierenden echtzeitfähigen Ethernet-Lösungen. Diese haben zum größten Teil gemeinsam, dass sie Subnetze bilden, in denen ausschließlich echtzeitfähige Geräte erlaubt sind. Es ist zwar asynchroner Datenverkehr möglich, ein angeschlossener Rechner ohne Kenntnis des Echtzeitprotokolls beeinträchtigt jedoch die Echtzeitfähigkeit des Netzes. Geräte auf der Basis von Standard-Ethernet können lediglich über Gateways angeschlossen werden. In diesen Fällen kann jedoch gleichermassen ein Feldbus mit Ethernet-Gateway zum Einsatz kommen. Des Weiteren basieren viele existierende Lösungen auf einem Master/Slave-Kommunikationsmodell. Sie beachten nicht konsequent den Trend zu dezentraler Peripherie, bei der unabhängige parallele Übertragungen im Netzwerk statt finden.

Im Rahmen dieser Arbeit wird durch das Framework eine neue Sichtweise auf echtzeitfähige Netzwerke der Automatisierungstechnik auf Ethernet-Basis erarbeitet. Zusätzlich dazu wird ein neuartiger Ansatz beschrieben, der auf einem TDMA-Zugriffsverfahren basiert und keine strikte Subnetz-Bildung erzwingt. Geräte wie Laptops oder Standard-PCs können transparent in das echtzeitfähige Netzwerk hinzugefügt und wieder entfernt werden. Sie können asynchrone Daten senden und empfangen, ohne das Echtzeitverhalten des Netzwerkes zu beeinflussen. Zur Realisierung des TDMA-Verfahrens werden auf Basis der im Vorfeld bekannten Echtzeit-Übertragungen Schedules berechnet. Des Weiteren wird die Fragestellung erörtert, an welcher Stelle des Netzwerkes die Schedules umgesetzt werden. Dabei wird ein zentraler Scheduler ebenso betrachtet wie dezentrale Schedules in den Switches sowie aktiv sendende und passive Geräte. Abschließend wird die Entwicklung eines neuartigen Switches skizziert, welcher Frames mit hohen Echtzeitanforderungen schneller als cut-through Switches deterministisch weiterleiten kann.

Abstract

Within the scope of automation technology, a trend to substitute established fieldbuses with realtime Ethernet networks or to integrate Ethernet frames into a fieldbus infrastructure can be observed since several years. The aim is to use a single network from the office area to the field devices like sensors, actuators or user-panels. This trend is characterized as vertical integration. A central issue is that widespread Standard-Ethernet networks do not satisfy the demands of industrial realtime in automation technology. However, the Ethernet standard ought to be observed in the field area as well. Otherwise, separate field bus infrastructures would arise again. The main objective of this work is the development of a formal profounded framework, which allows to vary between compability to the widespread Ethernet standard on the one hand, and the compliance with concrete realtime demands of an automated plant on the other hand.

First of all, an overview and classification of existing realtime Ethernet solutions is given. These solutions mostly build subnets consisting exclusively of realtime devices. Though it is possible to transfer data in an asynchronous manner, a connected personal computer without knowledge about the realtime protocol would impact the realtime capabilities of the network. Standard Ethernet devices can only be connected via gateways. However, in this case a common fieldbus with Ethernet gateway can be used in the same manner. Furthermore, many existing solutions are based on a master/slave communication model. They do not consequently follow the trend towards decentral periphery, which allows concurrent communication in the network.

Within the scope of this work, the developed framework allows a new perspective on realtime networks for automation technology based on Ethernet. In addition to this, a new approach is depicted, which is based on a TDMA access method without enforcing strict subnets. Devices like laptops or personal computers can be added to the network and removed from the network seamlessly. They can transmit and receive asynchronous data without influencing the realtime capabilities of the network. In order to realise the TDMA access method, offline schedules are calculated with regard to the well known realtime communication requirements. Furthermore, the location for implementing the schedules in the network is discussed. In this context, a central scheduler is considered as well as decentral schedules inside the switches. Active transmitting devices are included as well as passive devices. Finally, the design of a new switching hardware is outlined. This switch will be able to transmit Ethernet frames with high realtime requirements deterministically and even faster than cut-through switches.

Danksagung

Eine mehrjährige wissenschaftliche Arbeit zum Abschluss zu bringen, ist ohne die Unterstützung von anderen Personen undenkbar. Bei all diesen Personen möchte ich mich herzlich bedanken.

An erster Stelle danke ich Herrn Prof. Dr. Wismüller für die kontinuierliche Betreuung und Unterstützung meiner Arbeit. Er war jederzeit unabhängig von seinen vielfältigen Aufgaben innerhalb der Fachgruppe ansprechbar, wenn ich mit meinen Gedanken nicht weiter gekommen bin. Durch seine Kompetenz, die logische Argumentationsweise und seine Fokussierung hat er in fachlichen Diskussionen den Fortschritt dieser Arbeit in einer Art voran getrieben, die den Erfolg dieser Arbeit erst ermöglichte. Zusätzlich hat er ein angenehmes und sorgloses Arbeitsklima am Lehrstuhl geschaffen, das einen hohen Anteil an wissenschaftlicher Forschung ermöglicht. Mein Dank gilt auch Herrn Prof. Dr. Günter Schröder für die Bereitschaft, das Zweitgutachten für diese Arbeit zu erstellen. Er hat die vorliegende Arbeit von Beginn an begleitet und sicher gestellt, dass eine Relevanz aus Sicht der Automatisierungstechnik besteht. In diesem Zusammenhang bedanke ich mich auch bei Prof. Thomas Müller von der Zürcher Hochschule Wintherthur, der mich bei der Themenfindung unterstützt hat. Ein Dank geht auch an den Laboringenieur der Fachgruppe, Herrn Mannfred Stettner, für die sehr kompetente Hilfestellung in allen Hardwarefragen.

Zum Gelingen einer wissenschaflichen Arbeit tragen auch studentische Arbeiten bei. Hier ist insbesondere die ausserordentliche Leistung von Uwe Nowak im Rahmen seiner Diplomarbeit des Fachbereiches Mathematik zu nennen. Er hat den Kern der formalen Modellierung und die Beweisführung in einer beachtlichen Präzision durchgeführt. Die Ergebnisse der Diplomarbeit von Henning Westerholt sind ebenso hervorzuheben wie die Diplomarbeit von Samer Hijazi, der erste Tests der Realisierbarkeit der Switching-Hardware durchgeführt hat. Als besondere Leistung der Implementierung des Software-Frameworks, mit dessen Hilfe Algorithmen zur Berechnung von Echtzeit-Schedules modular ausgeführt und große Netze simuliert werden können, ist auch die Arbeit von Klaus Bernshausen, Jens Brennscheidt, Philip Gibson, Carsten Igel, Jens Kammer, Matthias Mielke, Stefan Poniewas, Simon Schöling und Kasbar Sulieman im Rahmen der Projektgruppe zu nennen. Als alternative Ansätze sind ebenso die Bachelor-Arbeit aus dem Fachbereich Mathematik von Ame Biova Gerard Agbanzo sowie die Studienarbeit von Marcus Merz zu erwähnen.

Schließlich bin ich auch meiner Familie, Freunden und Bekannten zu Dank verpflichtet. Obwohl ich in den letzten Jahren nicht sehr viel Zeit für sie hatte, haben sie mich stets durch ihr Vertrauen bestärkt, motiviert oder auch auf andere Gedanken gebracht. Somit haben auch sie letztlich zu dieser Arbeit beigetragen.

Inhaltsverzeichnis

Kapitel 1

Einleitung und Motivation

Bereits seit vielen Jahren hat sich der Ethernet-Standard in Büro-Netzwerken und im Heim-Bereich durchgesetzt [Ada01] [FG05]. Auf jeder neu erworbenen Mutterplatine eines PCs befindet sich heutzutage eine integrierte Netzwerkkarte, welche Datenraten von $100Mbit/s$ oder $1Gbit/s$ ermöglicht. Büro-Gebäude und sogar private Haushalte werden mit Netzwerkkabeln durchzogen, welche über preisgünstige Switches verbunden werden. Nahezu jede Person ist in der Lage, ein solches Netzwerk zu verkabeln, zu konfigurieren und in Betrieb zu nehmen.

Auf der anderen Seite existieren traditionelle Feldbusse [Sch03] wie ProfiBus [Pop00], InterBus-S [BM94], AS-i [KM99] oder CAN-Bus [Eng00], die im Umfeld von automatisierten Anlagen zum Einsatz kommen. Solche Anlagen fertigen beispielsweise Produkte aller Art in Massenproduktion oder führen komplexe mechanische Bewegungen aus. Diese speziellen Busse sind aus der *Automatisierungstechnik* entstanden und auf deren Bedürfnisse abgestimmt. Abbildung 1.1 zeigt als Beispiel eine automatisierte Anlage der Unitechnik AG zur Stahlproduktion, die von einem Leitstand aus überwacht wird:

Abbildung 1.1: Automatisierte Anlage der Unitechnik AG [Uni06]

Für die Konzeptionierung größerer Anlagen dieser Art existiert ein eigenes *Ebenenmodell*, welches in Abbildung 1.2 dargestellt wird. Die *Feldebene* beinhaltet Sensorik, unter anderem zur Abstands-, Geschwindigkeits-, Temperatur- und Druckmessung sowie Aktoren wie Servoantriebe für Walzen, Transportketten oder pneumatische Ventile. Auf dieser Ebene sind die *Feldbusse* verbreitet, welche eine stark deterministische Datenübertragung in kurzen Zyklen ermöglichen. Dabei werden nur wenige Bytes an Nutzdaten, meist einzelne Messwerte wie Soll- oder Istpositionen, übertragen. Wichtig ist jedoch, dass diese Daten mit hohen Echtzeitanforderungen in sehr kurzen Zeitintervallen und mit minimaler, möglichst konstanter Verzögerungszeit übertragen werden können, damit die Anlage in einem schnellen Takt produzieren kann, der nur durch mechanische und physikalische Grenzen der Konstruktion der Anlage begrenzt wird.

Gesteuert werden die Sensoren und Aktoren auf der zweiten Ebene, der *Zellebene*. Dies geschieht zumeist durch eine *speicherprogrammierbare Steuerung* (SPS), welche für einen Anlagenteil zuständig ist. Die Steuerungen der Anlagenteile stehen wiederum in Kommunikation zueinander. Auch Bedienpulte sind auf der Zellebene zu finden, mit deren Hilfe Maschinenbediener die Konfiguration eines Anlagenteils auf unterer Ebene vornehmen können, beispielsweise die Temperatur eines Ofens erhöhen oder verringern.

Auf der dritten Ebene, der *Leitebene*, wird die Produktionskontrolle und -überwachung der Gesamtanlage, wie in Abbildung 1.1 oben links und unten rechts zu erkennen ist, auf Anwender-Niveau vollzogen. Diese Ebene ist meist mit dem Büronetzwerk des Unternehmens verbunden, um beispielsweise Produktionsvorgaben und Auftragsverwaltung durchzuführen. Der Austausch erfolgt über Schnittstellen zu Enterprise-Resource-Planning Systemen (ERP) wie SAP R/3 oder Microsoft Navision. Auf dieser Ebene ist Standard-Ethernet als Netzwerk-infrastruktur dominierend.

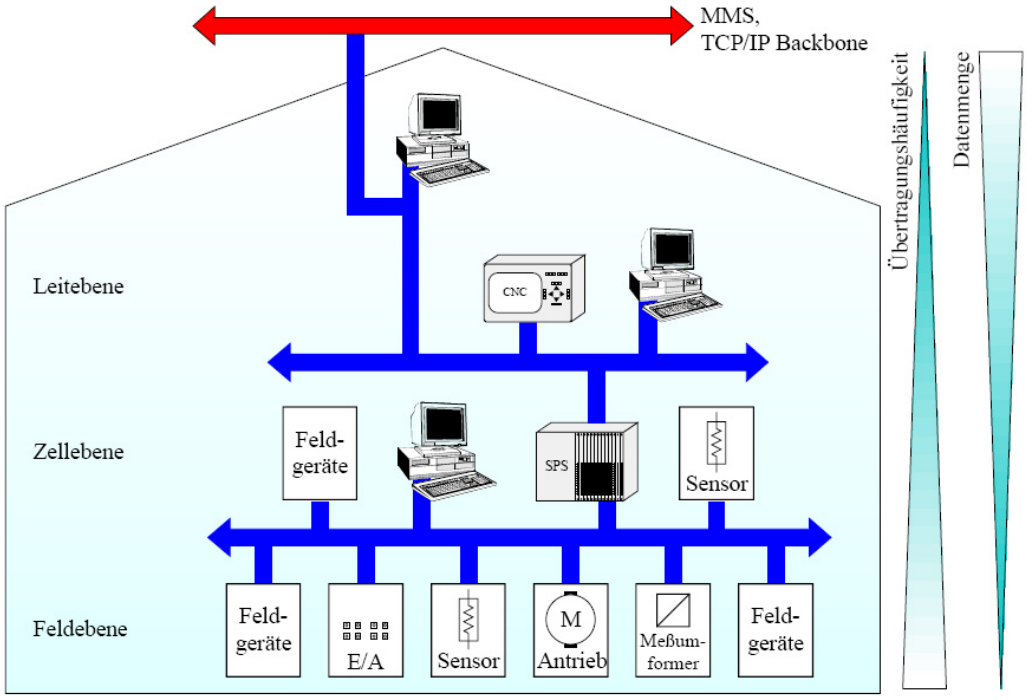


Abbildung 1.2: Ebenenmodell der Automatisierungstechnik [Var04]

Seit einigen Jahren ist jedoch der Trend zur *vertikalen Integration* [Ren06] zu beobachten. Dies bedeutet, dass durch den hohen Verbreitungsgrad, die kostengünstige Herstellung und die allgemeine Bekanntheit der Ethernet-Standard bis zur Feldebene durchgesetzt werden soll [Pop05], so dass sich eine einzelne Netzwerktechnologie durch das gesamte Unternehmen zieht. Die Feldbusse sollen dabei ersetzt oder zumindest ergänzt werden.

Dieser Trend bringt jedoch technologische Probleme mit sich [Jan02] [Fel00], da Ethernet ursprünglich nicht für die Bedürfnisse der Automatisierungstechnik bezüglich des Determinismus, der Datenraten und der minimalen Verzögerungszeiten entwickelt wurde. Diese Anforderungen sind gerade auf dem Gebiet der *Antriebstechnik*, auch als Motion Control bezeichnet, besonders strikt vorgegeben. Man spricht hier von harter Echtzeitfähigkeit. Ethernet hingegen bietet einen *best-effort* Ansatz, der auf der Verwendung von *Carrier Sense, Multiple Access with Collision Detection* (CSMA/CD) basiert. Kollidieren Datenframes auf einer Leitung, so werden sie nach einem nicht-deterministischen *Exponential Backoff* Algorithmus neu gesendet. Kommen ausschließlich Switches zum Einsatz, so wird das CSMA/CD-Verfahren nicht verwendet. Statt dessen puffern die Switches die Frames und können sie bei vollen Warteschlangen auch unter der Annahme verwerfen, dass die Frames neu übertragen werden. Demzufolge liegt auch in diesem Fall kein Determinismus vor.

Zusätzlich ist ein weiterer Trend der Automatisierungstechnik ersichtlich, der sich vom ursprünglichen *Master/Slave-Modell* mit zentralen speicherprogrammierbaren Steuerungen entfernt. Dieser Trend beinhaltet *dezentrale Peripherie* [Zha02] mit intelligenter Sensorik, die programmierbar ist und Teile der Datenauswertung selbst vornehmen kann [Har01]. Durch die Dezentralisierung der Programmierung wird jedes einzelne Modul einfacher zu handhaben, so dass auch komplexere Anlagen gesteuert werden können. Als Stichwort ist hier der „divide and conquer"-Ansatz zu nennen. Unterstützt wird die Intelligenz auf der Feldebene durch die technologischen Möglichkeiten, leistungsfähige Mikrocontroller preisgünstig zu produzieren. Durch die Dezentralisierung entsteht ein wachsender *Querverkehr* [Pop00] der Geräte untereinander, wodurch sich die Aufgaben der zentralen Steuerungen reduzieren. Auf diesen Trend müssen traditionelle Feldbusse erst abgestimmt werden, während Ethernet bereits Querverkehr aufgrund seiner Architektur zulässt.

Das Problem liegt insgesamt darin, den Ethernet-Standard beizubehalten und dennoch die Anforderungen der Automatisierungstechnik - insbesondere den geforderten Determinismus - zu erfüllen. Im Rahmen dieser Arbeit wird gezeigt, dass eine vollständige Einhaltung des Ethernet-Standards den harten Echtzeit-Anforderungen aus der Automatisierungstechnik nicht gerecht werden kann.

Eine Möglichkeit für einen Kompromiss zwischen Ethernet-Standard und Anforderungen der Automatisierungstechnik besteht in der Anpassung des Übertragungsmediums und in der Bereitstellung einer Schnittstelle, die dem Ethernet-Standard genügt. Dies bedingt jedoch eine Änderung der Hardware und führt dazu, dass wiederum geschlossene Subnetze mit Ethernet-Gateways entstehen. Diese Lösung lässt sich jedoch ebenso bei der Verwendung eines Feldbusses erreichen.

Ein anderer Ansatz verwendet Standard-Hardware und modifiziert die Übertragung oberhalb der Sicherungsschicht, indem ein deterministisches Zugriffsverfahren wie *Time Division, Multiple Access* (TDMA) [LP96] oder ein Token-basierter Mechanismus [VVTG06] eingesetzt wird. Auf diese Weise kann eine zeitliche Obergrenze für den Emp-

fang jeder Übertragung sowie eine Grenze für die Schwankung des Empfangszeitpunktes angegeben werden. Sind diese Grenzen genügend klein, können sie die Anforderungen der Automatisierungstechnik erfüllen. Möchte jedoch beispielsweise ein Manager über seinen Laptop Zugriff auf das Netzwerk erhalten, so verwendet er einen freien Ethernet-Port der Anlage. Da sein Laptop jedoch keine Kenntnis von dem veränderten Zugriffsverfahren besitzt, sendet es nach dem gewohnten Verfahren und zerstört somit die Echtzeitfähigkeit des Netzwerkes.

Es ist offensichtlich, dass ein Netzwerk bei höchsten Echtzeitanforderungen nicht in gleicher Form für asynchronen Datenverkehr - wie das Abrufen von e-Mails oder das Aufrufen von Internetseiten - geeignet sein kann. Ein solches Netzwerk unterscheidet sich ggf. sogar in seinen physikalischen Eigenschaften von Standard-Ethernet, soll dem Anwender jedoch vom Übertragungsmedium bis hin zur Anwendungsebene als Ethernet erscheinen und mit gewohnten Hardware- und Softwarekomponenten betrieben werden. Es herrscht demnach ein Spannungsfeld zwischen den Anforderungen der Automatisierungstechnik und der Einhaltung des Ethernet-Standards. Dennoch existieren bereits einige Lösungen wie Siemens ProfiNet [Pop05] [PW04], Ethernet PowerLink [Sch03d], EtherCAT [JB03] und GinLink [Ind04], die sich in der Praxis bewährt haben. Alle existieren Lösungen stellen jedoch jeweils einen festen Kompromiss zwischen der Kompatibilität zum Ethernet-Standard und den Forderungen der Automatisierungstechnik dar, wie es in Abbildung 1.3 skizziert ist. Diese existierenden Lösungen sind jeweils für bestimmte Randbedingungen effizient, die von der konkreten automatisierten Anlage abhängen. Die *Antriebstechnik* stellt dabei die härtesten Anforderungen.

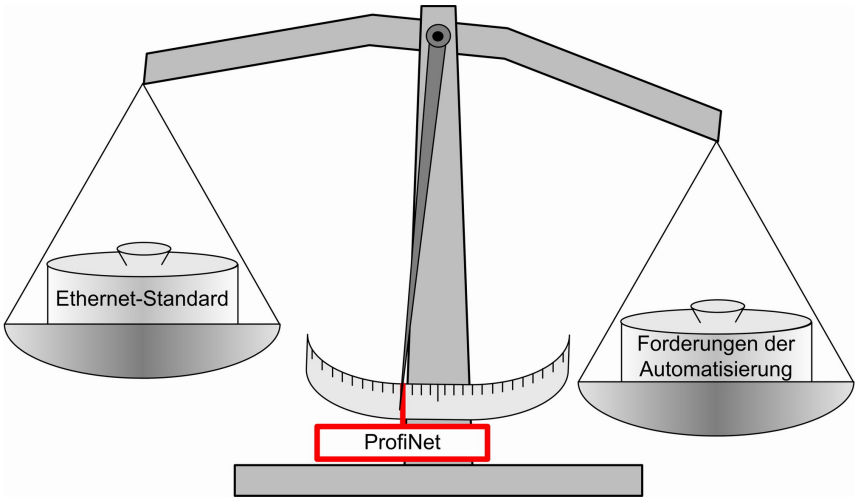


Abbildung 1.3: Kompromiss zwischen Ethernet-Standard und Automatisierung

Das Ziel dieser Arbeit besteht darin, eine abstraktere Modellierung eines echtzeitfähigen Ethernets zu entwickeln. Dabei wird von einer gegebenen Netzwerkinfrastruktur und vorgegebenen Übertragungen ausgegangen. Dies ist möglich, da die Infrastruktur im laufenden Betrieb der Anlage nicht verändert werden kann. Nach einer Veränderung ist eine Anpassung der Anlagenprogrammierung erforderlich. Die Übertragungen sind im Vorfeld bekannt, da sie bei der Programmierung der Anlage geplant werden. Aufgrund des notwendigen Determinismus werden auch für zeitkritische Ereignisse eigene *Zeitslots* im Vorfeld

geplant. Die Vorgehensweise besteht darin, zunächst die Infrastruktur und die Übertragungen formal zu modellieren. Dies geschieht zunächst unter starker Vereinfachung des Modells im Vergleich zur Realität. Diese Vereinfachungen werden schrittweise aufgehoben und neue auftretende Probleme der Modellierung gelöst. Dadurch soll ein *Framework* entstehen, welches es vor dem Bau der Anlage ermöglicht, zunächst unabhängig von einer konkreten Hardware-Implementierung einen individuellen Kompromiss zwischen der Echtzeitfähigkeit und der Einhaltung des Ethernet-Standards festzulegen. Der zeitliche Ablauf der Übertragungen wird dabei unter Erfüllung ihrer Echtzeit-Anforderungen geplant, so dass eine *Schedule* entsteht.

Das zweite Ziel der Arbeit besteht darin, Möglichkeiten zur Integration dieser Schedules in bestehende Netzwerk-Hardware zu diskutieren. Nach einer Gegenüberstellung dieser Möglichkeiten wird sowohl ein Simulationsframework, als auch ein Ansatz zur Modifizierung eines Ethernet-Switches skizziert.

Die Arbeit befindet sich also im *interdisziplinären* Spannungsfeld der Rechnernetze und der Erstellung von Schedules als Teilgebiete der Informatik und der echtzeitfähigen Netze als Nachfolger traditioneller Feldbusse der Automatisierungstechnik, siehe Abbildung 1.4. Für die Modellierung werden Werkzeuge aus dem Gebiet der Mathematik verwendet mit dem Ziel, zunächst das Netzwerk formal zu modellieren und schließlich die Schedules zu entwickeln, mit deren Hilfe ein Determinismus des Ethernets erreicht werden kann. Das Scheduling wiederum verbindet die Mathematik mit der Informatik, so dass sich die Arbeit der Methoden der Informatik, der Automatisierungstechnik und der Mathematik bedient.

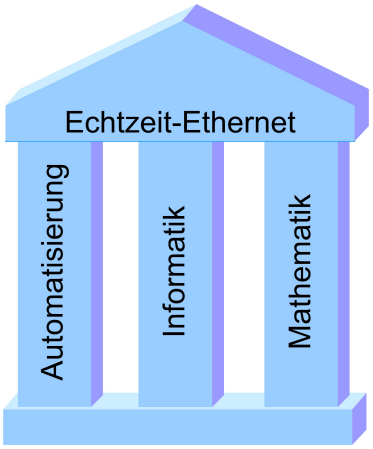


Abbildung 1.4: Inhaltliche Säulen dieser Arbeit

Diese Arbeit ist in 6 Hauptkapitel gegliedert. Im folgenden zweiten Kapitel werden grundlegende Begriffe und Vorgehensweisen aus den Gebieten der Automatisierungstechnik, der Ethernet-Netzwerke und der Graphentheorie behandelt, welche für den weiteren Verlauf der Arbeit von Bedeutung sind. Im dritten Kapitel werden die existierenden Ansätze (State-of-the-Art) zur Realisierung von echtzeitfähigem Ethernet vorgestellt und klassifiziert. Es erfolgt ein Vergleich der Ansätze und eine Bewertung aus Sicht der Automatisierungstechnik sowie aus Sicht der Nähe zu Ethernet-Standards. Dabei werden Mängel der bestehenden Ansätze aufgezeigt und die Notwendigkeit für ein umfassenderes Framework begründet. Abschließend wird die eigentliche Aufgabe präzisiert, der Gegenstandsbereich festgelegt und abgegrenzt.

Die Modellierung des Frameworks erfolgt dann im vierten Kapitel unter Verwendung der graphentheoretischen Grundlagen. Da die Beweisführung im Rahmen einer Diplomarbeit des Fachbereiches Mathematik durchgeführt wurde [Now06], konzentriert sich diese Arbeit auf die Bedeutung der Ergebnisse auf die Netzwerktechnik.

Zu Beginn des fünften Kapitels wird aufgezeigt, wie das formale Framework in einer Softwaresimulation umgesetzt werden kann. Zusätzlich wird diskutiert, wie die aus der Modellierung entstandenen Schedules in den Netzwerken zur Anwendung kommen können. Dabei wird auf die Problematik der Synchronisation der Switches ebenso eingegangen wie auf den Unterschied des Pollings von passiven Geräten und dem aktiven Senden von synchronisierten Geräten. Es wird skizziert, welche Maßnahmen notwendig sind, um die Schedules in das Netzwerk zu integrieren. Im abschließenden Teil des fünften Kapitels wird schließlich der Aufbau eines modifizierten Ethernet-Switches skizziert.

Eine kritische Bewertung der Ergebnisse sowie zukünftige Forschungsperspektiven runden den Verlauf dieser Arbeit schließlich ab.

Kapitel 2

Grundlagen

In diesem Kapitel werden grundlegende Begriffe und Techniken beschrieben, die für die weitere Arbeit von Bedeutung sind. Es ist nach den inhaltlichen Säulen dieser Arbeit (s. Abbildung 1.4) in die drei Teile gegliedert, welche einen Einfluss auf die zu entwickelnde Lösung besitzen.

Zunächst werden die Anforderungen an ein Netzwerk aus Sicht der Automatisierungstechnik beschrieben. Im Vordergrund liegt hier der Begriff der Echtzeit, des notwendigen Determinismus, der Synchronität sowie die Besonderheiten der Sendungen von Geräten wie Sensoren oder Aktoren in einem solchen Netzwerk.

Im zweiten Teil dieses Kapitels wird der Ethernet-Standard betrachtet mit den übertragenen Frames, der Adressierung und der Funktionsweise der Verteiler wie Hubs oder Switches. Des Weiteren werden gängige Verfahren im Ethernet, wie die Exponential Backoff Strategie und Algorithmen zum Erlernen der Weiterleitungstabellen von Switches skizziert. Der Fokus liegt hier auf der Fragestellung, welche Auswirkungen diese Verfahren auf den Determinismus des Netzwerkes besitzen. Abgeschlossen wird der zweite Teil durch die Vorstellung von gängigen Vorgehensweisen zur Zeitsynchronisation im Ethernet. Generell liegt der Zugriff auf die physikalische Schicht, der ersten Schicht des Referenzmodells der International Organization for Standardization / Open Systems Interconnection (ISO/OSI) [Tan03] [PD04], sowie der Aufbau der Sicherungsschicht (OSI-Schicht 2) im Fokus dieser Arbeit.

Wie bereits in der Einleitung erwähnt, werden in dieser Arbeit mathematische Verfahren als Werkzeuge zur Integration des Ethernets in das Umfeld der Automatisierungstechnik eingesetzt. Diese Verfahren werden einerseits für die Modellierung des Netzwerkes und andererseits für die Einführung von Determinismus durch die Erstellung von Schedules verwendet. Zum Abschluss dieses Kapitels werden die dazu notwendigen Grundlagen aus dem Bereich der Graphentheorie gelegt und die entsprechenden Definitionen vorgenommen.

2.1 Systeme der Automatisierungstechnik

Um Ethernet an die Bedürfnisse der *Automatisierungstechnik* anzupassen, werden diese Bedürfnisse in diesem Kapitel beschrieben. Aus dem Aufbau von automatisierten Anlagen werden die hohen Anforderungen von automatisierten Systemen anhand ihrer Ursache

aus der Regelungstechnik begründet. Grundlegende Definitionen aus dem Kontext der Echtzeitfähigkeit werden ebenso beschrieben wie ein Beispiel für einen Feldbus, der diese Eigenschaften besitzt.

2.1.1 Automatisierte Produktionsanlagen

Automatisierte *Anlagen*, z. B. Fertigungsstrassen von Automobilen, komplexe Roboter, Verpackungs-, Papier- oder Textilmaschinen bestehen aus drei Klassen von Komponenten, nämlich

- der Sensorik,
- der Aktorik und
- der Steuerung.

„*Die Regelungstechnik ist zusammen mit der Steuerungstechnik eine der Hauptsäulen der Automatisierung*“ [Dit90]. Die Sensorik, Aktorik und die Steuerung zusammen bilden einen *Regelkreis*. Dabei wird der aktuelle Zustand der automatisierten Anlage im ersten Schritt mit Hilfe der Sensorik erfasst, beispielsweise die Position von Paketen auf einem Förderband mittels einer Lichtschranke. Der Inhalt der Pakete kann z. B. über einen Radio Frequency Identification Transponder (RFID) ausgelesen werden. Ebenso kann das Gewicht und die Temperatur, beispielsweise bei verderblicher Ware, ermittelt werden. Die ermittelten Daten werden über ein Netzwerk zu der Steuerung gesendet und können über ein angeschlossenes Bediengerät, z. B. über einen Industrie-PC, als Istwerte vom Bediener der Anlage eingesehen werden. Der Bediener kann Soll- und Grenzwerte eingeben, die ebenfalls zur Steuerung gesendet werden. Die Steuerung erfasst und interpretiert die eingehenden Daten entsprechend ihres Programmes und gibt notwendige Signale an die Aktoren für den Fortgang der Produktion aus. Bei den Aktoren kann es sich in diesem Beispiel um Weichen und Motoren des Förderbandes, um Kühlelemente einer Kühlstrecke oder um Greifarme von Robotern zur Paketbeförderung handeln. Abbildung 2.1 stellt den Ablauf des automatiserten Prozesses graphisch dar.

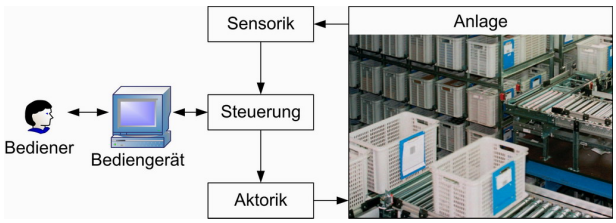


Abbildung 2.1: Prinzip der automatisierten Fertigung

In diesem Kapitel werden die Komponenten der automatisierten Anlagen vorgestellt unter dem Blickwinkel der Informatik. Die präzise interne Funktionsweise muss dabei nicht beachtet werden, da lediglich die Anforderungen an ein Netzwerk zur Übertragung von Daten von Interesse sind.

2.1.1.1 Sensoren

Ein *Sensor* ist ein technisches Bauteil, das physikalische oder chemische Eigenschaften wie beispielsweise Temperatur, Druck, Feuchtigkeit, Schall, Helligkeit oder Beschleunigung als Messgröße erfasst. „*Sensoren haben die Aufgabe, Informationen über den Prozeßzustand abzufragen und an die Automatisierungsgeräte weiterzuleiten*“ [Pho98]. Ursprünglich wurden die gemessenen Signale direkt mit den Eingängen einer Steuerung verdrahtet. Durch die sehr hohe Anzahl an Sensoren heutiger Anlagen haben sich *Feldbusse* etabliert, welche ein einheitliches Netzwerk bilden. Sensoren dienen als Datenquellen und senden ihre Messwerte über einen Feldbus an die Steuerung, wobei mehrere Sensoren in einer Linientopologie hintereinander geschaltet werden können. Die Übertragung von analogen Signalen, s. Abbildung 2.2a und 2.2b, ist kaum noch von Bedeutung. Statt dessen werden die Signale direkt im Sensor quantisiert, diskretisiert und mit einem Analog-Digital Umsetzer (ADU) digital übertragen. Auf diese Weise ist eine standardisierte Verarbeitung, z.B. eine Signalverstärkung oder eine Weiterleitung in einem Netzwerk, wesentlich leichter möglich. Dazu kommt, dass ein Automatisierungsgerät wie eine Steuerung die Daten ohnehin digital verarbeitet. Aus diesen Gründen sind die in Abbildung 2.2c dargestellten Sensoren heutzutage vorherrschend.

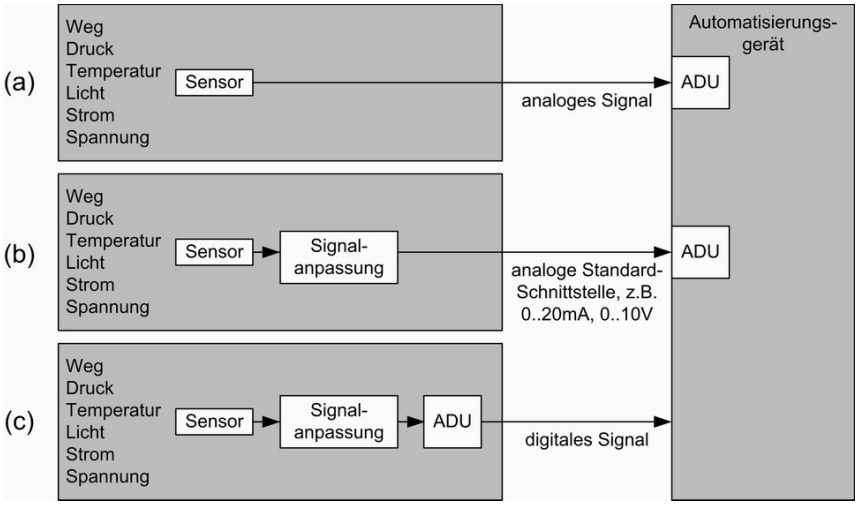


Abbildung 2.2: Arten von Sensoren und deren Datenübertragung [Pho98]

Durch diese Vereinheitlichung wird der Aufwand zur Verkabelung minimiert und durch den eingesetzten Feldbus standardisiert. Neben den oben genannten elementaren Sensoren zur Erfassung und Übertragung einzelner Messgrößen können auch komplexere Geräte, wie RFID-Lesegeräte, als Sensoren aufgefasst werden. Diese können Datenmengen von einigen Kilobyte auslesen und an die Steuerung übermitteln.

Viele Sensoren, wie Lichtschranken oder Endschalter, besitzen ein rein binäres Verhalten. Sensoren zur Erfassung von physikalischen Größen sind oft in der Lage, neben der Signalanpassung eine Vorverarbeitung der gemessenen Größen zu übernehmen, so dass sie der Steuerung lediglich die Über- bzw. Unterschreitung von Grenzwerten mitteilen oder Status- und Fehlerinformationen liefern [KM99]. Weiter fortgeschrittene Sensoren sind sogar so programmierbar, dass sie selbst Daten auswerten und direkt an den zuständigen

Aktor (s. Kapitel 2.1.1.2) senden können. Diese Art der Kommunikation wird als Querverkehr bezeichnet. Auf diese Weise wird die zentrale Steuerung entlastet. Während ein diskreter Sensor lediglich Meßwerte aufnimmt und ein integrierter Sensor die Signale zusätzlich umwandelt, wird ein Sensor mit eigenem Mikrorechner als *„intelligenter Sensor“* bezeichnet. Ein solcher Sensor besitzt den höchsten Integrationsgrad.

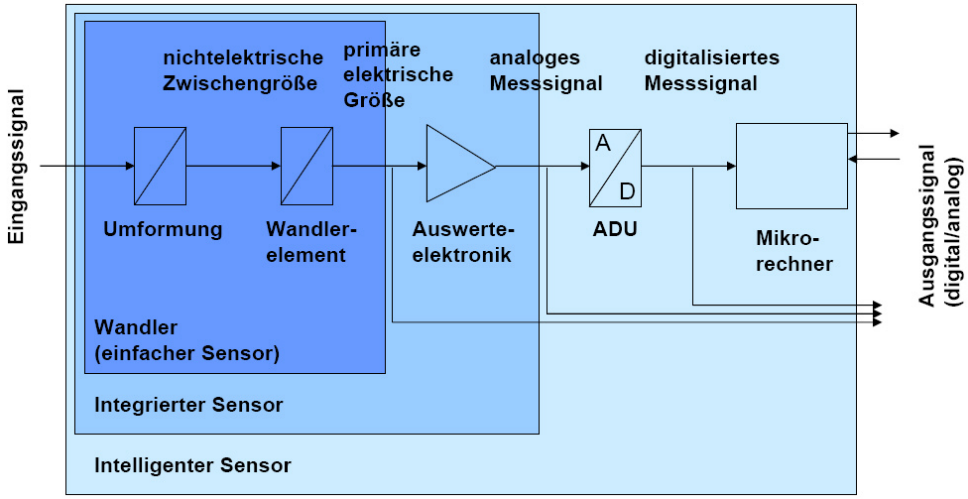


Abbildung 2.3: Integrationsgrad von Sensoren [HF03]

„Überall ist der Trend zur Dezentralisierung zu beobachten. Die Feldgeräte übernehmen immer mehr Funktionen, die früher zentral [. . .] erledigt wurden. Ein intelligenter Sensor kann die Meßwertaufbereitung und/oder die Überwachung bezüglich oberer und unterer Grenzwerte vornehmen, ja er kann sogar gleich die Regelaufgabe übernehmen, also den Regelalgorithmus ausführen“. [Rei02]

2.1.1.2 Aktoren und Frequenzumrichter

Aktoren oder Aktuatoren bilden das Stellglied in einem Regelkreis. Sie setzen ankommende Signale einer Regelung in mechanische Arbeit, meist in Bewegungen um. Typische Aktoren sind Elektromotoren, elektromechanische Aktoren, Schütze und Relais, Hydraulik- und Pneumatik-Ventile.

Wie bereits im Bereich der Sensorik vorgestellt, hat sich auch bei den Aktoren der Begriff der Intelligenz etabliert. *„Intelligente Aktoren [...] zeichnen sich dadurch aus, dass in der Regel ein Rückmeldekanal in den Aktuator integriert ist. Damit kann die Erreichung der Endposition signalisiert werden. Es kann aber auch eine Fehlerrückmeldung erfolgen [.../“* [KM99]. Aus Sicht der Netzwerktechnik sind Aktoren vorwiegend Datensenken, die Sollwerte empfangen. Jedoch muss berücksichtigt werden, dass neben den bereits erwähnten Daten auch Istwerte der Aktorik an die Steuerung zurückgesendet werden können. Dies geschieht insbesondere im Fall der Antriebsregelung, s. Kapitel 2.1.2.

Die Antriebe werden klassifiziert in Gleichstrommotoren, Wechselstrommotoren und *Schrittmotoren*. Letztere werden direkt über binäre Signale angesteuert, die ein Magnetfeld in einem Eisenkern rotieren lassen. Die Inkremente einer Rotation werden über eine

Datenfolge vorgegeben, das Funktionsprinzip ist z. B. in [Pho98] beschrieben. Die präzise Ansteuerung des Motors ermöglicht eine exakte Positionierung innerhalb der Anlage. Gleich- und Wechselstrommotoren werden über eine Spannung angesteuert, wobei die Drehzahl eines Gleichstrommotors über die Höhe der Spannung und die Drehzahl eines Wechselstrommotors über die Frequenz der Spannung gesteuert wird.

Für die Ansteuerung von Wechselstrommotoren wird ein *Frequenzumrichter* benötigt. Diese Geräte sind so komplex, dass sie über einen eigenen Mikrocontroller verfügen und programmierte Verfahrbewegungen selbstständig abarbeiten können. Frequenzumrichter für die Antriebstechnik verfügen in der Regel über mehrere Betriebsarten und optimieren dabei das Anlauf- und Drehzahlverhalten des zu steuernden Motors. Moderne Frequenzumrichter können dabei über einen Feldbus angesteuert werden.

2.1.1.3 (De-)zentrale Steuerungen

Die eingehenden Daten der Sensorik und die von den Aktoren benötigten ausgehenden Steuersignale werden durch eine Steuerung miteinander verbunden. Die Steuerung enthält die benötigte Intelligenz des Regelkreises. Man unterschreidet prinzipiell *verbindungsprogrammierte Steuerungen* (VPS), *speicherprogrammierbare Steuerungen* (SPS) und *mikrocontrollerbasierte Steuerungen*.

Verbindungsprogrammierte Steuerungen realisieren ihre Anwendung durch eine nicht programmierbare Verdrahtung der Sensorik mit der Aktorik. Eine Umprogrammierung ist mit einer neuen Verkabelung verbunden, weswegen verbindungsprogrammierte Steuerungen als nicht mehr aktuell betrachtet werden.

Mit der Weiterentwicklung der Mikroelektronik haben sich speicherprogrammierbare Steuerungen etabliert, die heutzutage in industriellen Fertigungsprozessen überwiegend zum Einsatz kommen. Die Funktion der Steuerung wird in eine programmierbare CPU als zentrale Verarbeitungseinheit konzentriert.

Die Funktionsweise einer SPS ist auf industrielle Anforderungen optimiert und unterscheidet sich daher von einer mikrocontrollerbasierten Steuerung. Eine SPS besteht neben der CPU aus einer Anzahl von Baugruppen, die modular zum System hinzugefügt werden können und die Sensorik und Aktorik ansteuern. Das Betriebssystem der CPU einer SPS sorgt für einen zyklischen Ablauf des Programms, der stets in drei Teile untergliedert ist:

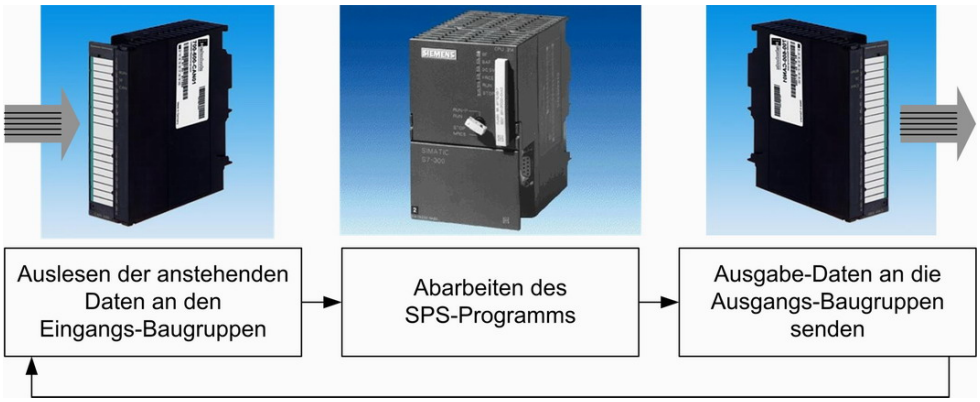


Abbildung 2.4: Ablauf eines SPS-Programms

Zu Beginn eines Zyklus wird der Zustand der mit den Eingangsbaugruppen verdrahteten Eingänge in das *Prozessabbild der Eingänge* (PAE) übertragen, danach bleibt ein Signalwechsel an den Eingängen in diesem Zyklus ohne Folgen. Die Übertragung wird von dem Betriebssystem der SPS durchgeführt. Im Anschluss daran wird das SPS-Programm seriell abgearbeitet, wobei auf das PAE lesend Zugriff genommen werden kann. Während des Programmablaufes können auch Wertzuweisungen an die Ausgänge getätigt werden, die jedoch nicht unmittelbar an die Ausgabebaugruppen weiter gereicht werden. Statt dessen werden diese Wertzuweisungen in dem *Prozessabbild der Ausgänge* (PAA) abgelegt. Nach Beendigung eines Programmablaufes wird das PAA durch das Betriebssystem der SPS an die Ausgabebaugruppen übertragen. Erst dadurch ändern sich die Zustände der Ausgänge. Im Anschluss daran beginnt ein neuer Programmzyklus. [Bri04]

Innerhalb einer Abarbeitung des SPS-Programms werden „*im Prinzip alle Steueranweisungen bearbeitet, also auch solche, die im Augenblick gar nicht relevant sind.*“ [Rei02] Dies führt zu einem Overhead, der von der Rechenleistung der CPU abzuziehen ist. Bei der *Zykluszeit* T_Z handelt es sich um die Zeit, welche die Abarbeitung des SPS-Programms benötigt.

Reißenweber arbeitet den Unterschied zwischen der Zykluszeit und der Totzeit heraus. Denn „*im ungünstigsten Fall ändert sich das Sensorsignal erst unmittelbar nachdem es von der Steuerung abgefragt wurde. Dann wird der nachfolgende Bearbeitungszyklus der SPS mit dem alten Signalwert durchlaufen. Erst im nächsten Zyklus wird aus dem neuen Sensorwert ein neuer Aktorwert berechnet. Im anderen Extremfall, der günstigsten Situation, ändert sich das Sensorsignal kurz bevor es eingelesen wird; dann ist die Verzögerung vernachlässigbar klein. Über viele Schaltvorgänge hinweg gemittelt, erhält man im Durchschnitt eine zeitliche Verzögerung von einer halben Zykluszeit.*“ [Rei02]

Die Zeit, die eine Änderung eines Sensorsignals bis zu seiner Auswirkung auf den Aktor benötigt, wird als *Totzeit* T_t bezeichnet. Die *mittlere Totzeit* $T_{t,mittel}$ bezeichnet die über viele Zyklen gemessene Totzeit. Addiert man die Verzögerung der Signalerfassung mit der Zykluszeit einer SPS, so ergibt sich die im Millisekunden-Bereich liegende mittlere Totzeit mit $T_{t,mittel} = 1.5 \cdot T_Z$.

Die Arbeitsweise einer mikrocontrollerbasierten Steuerung basiert auf einem Multitaskingsystem in Kombination mit einem entsprechenden echtzeitfähigen Betriebssystem. Ähnlich wie bei einer SPS verfügt auch eine mikrocontrollerbasierte Steuerung über einen Overhead bei der Programmuausführung. Dieser ist durch den Rechenaufwand bei einem Taskwechsel begründet. Durch Priorisierung kann die Wichtigkeit der Tasks in solch einem System eingestuft werden. Eine Priorisierung kann mitlerweile auch in einer SPS durchgeführt werden. Die entsprechenden Mechanismen sind nach International Electrotechnical Commission (IEC) 61131 [JT00] genormt.

Werden die Sensordaten zyklisch mit der Abtastzeit T_A gepollt, so ergibt sich je nach Eintreffen der Signaländerung eine mittlere Totzeit von $T_{t,mittel} = 0.5 \cdot T_A + T_B$, wobei T_B die Bearbeitungszeit des Mikrocontrollers darstellt.

Unter harten Echtzeitbedingungen (s. Kapitel 2.1.3.4), insbesondere bei einer Gefahrensituation, ist die geforderte *Reaktionszeit* besonders gering. In solchen Fällen wird bei einer mikrocontroller-basierten Steuerung nicht zyklisch abgetastet, sondern mit hochprioren Interrupts gearbeitet. „*Der Prozessor unterbricht das Programm [...] und springt zu [...] dem Interrupthandler. Dieser Interrupthandler kann selbst die Reaktion einleiten*

und/oder einen hochprioren Task starten. So werden nach der sogenannten Interruptreaktionszeit T_R, die beim heutigen Stand der Technik im Bereich weniger Mikrosekunden liegt, die notwendigen Ausgangsgrößen berechnet und an die Aktoren ausgegeben.“ [Rei02] Die mittlere Totzeit ist in diesem Falle sehr gering und beträgt $T_{t,mittel} = T_R + T_B$.

2.1.2 Die Antriebsregelung: Motion Control

Eine weit verbreitete Anwendung mit sehr harten Echtzeitbedingungen ist die *Antriebsregelung*, deren Grundlagen in diesem Kapitel skizziert werden. „*Ein typischer Anwendungsfall ist der Drehzahlgleichlauf mehrerer Antriebe bei kontinuierlichen Prozessen mit einer durchlaufenden Warenbahn (Walzwerk, Wickler)*“ [Wen03], die in der Abbildung 2.5 skizziert wird. Die Schwierigkeit einer durchlaufenden Warenbahn, bei der z. B. Bleche oder Papier in hoher Geschwindigkeit befördert werden, ist darin begründet, dass eine geringe Abweichung der Drehzahl eines einzelnen Antriebs zu einem Abriss der Ware führt und damit zu einer Unterbrechung der gesamten Produktion. Dabei kann beispielsweise eine Blechrolle mit einem Gewicht von mehreren Tonnen zu Produktionsausschuß werden. Zusätzlich dazu muss die Warenbahn neu „eingefädelt“ werden: Im Beispiel muss demnach eine neue Rolle in die Produktion geführt werden. Erst im Anschluss daran kann die Produktion neu anlaufen.

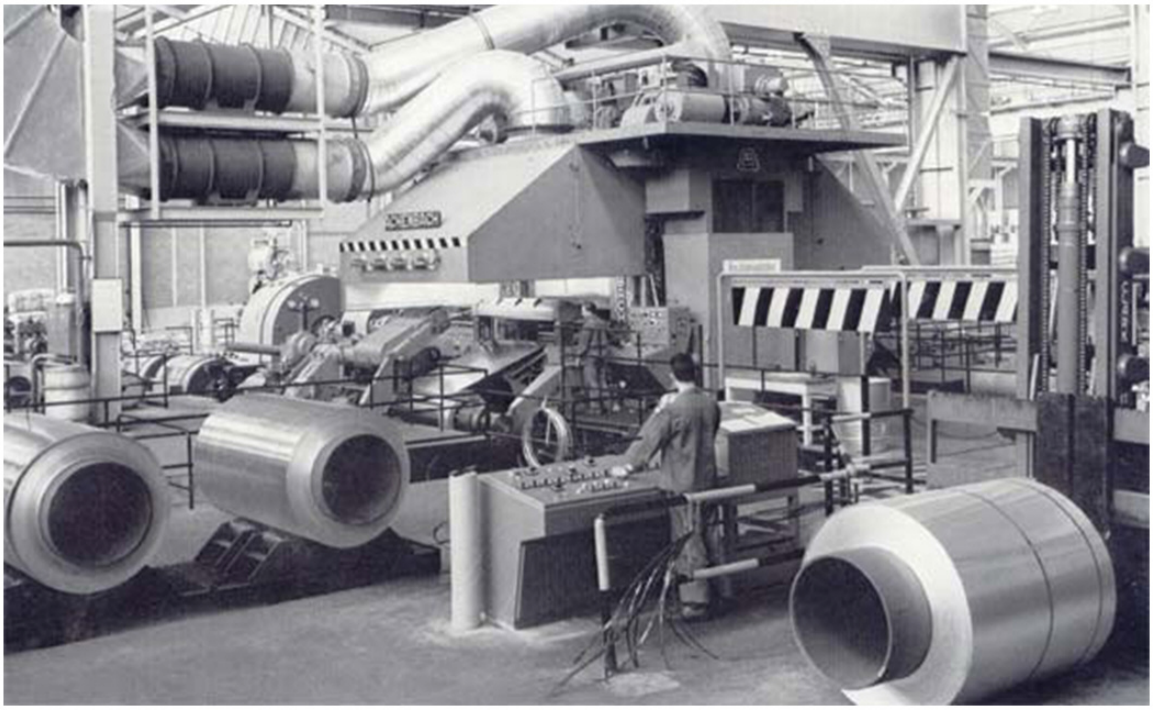


Abbildung 2.5: Steuerung eines Walzwerkes [Nov05]

Die Antriebe müssen also streng synchronisiert sein. In der traditionellen Automatisierung wird eine solche Synchronisierung über eine zentrale mechanische Welle vorgenommen, deren Umlauf den Produktionstakt vorgibt. Komplexe Bewegungen im Takt der Produktion können in diesem Fall über sogenannte *Kurvenscheiben* vorgenommen werden. Abbildung 2.6 zeigt eine solche Kurvenscheibe, die eine Zange mechanisch steuert:

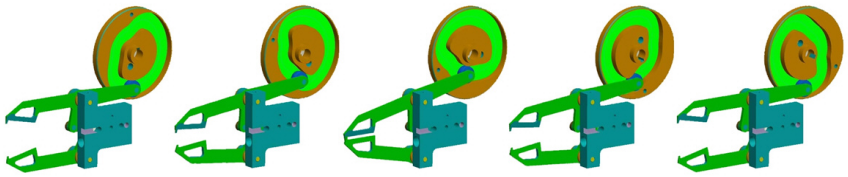


Abbildung 2.6: Über eine Kurvenscheibe gesteuerte Zange [Pem06]

Aus Sicht der Regelungstechnik ist eine Antriebsregelung ein *kaskadierter Regelkreis*, der aus einer Stromregelung, einer Drehzahlregelung und einer Positionsregelung besteht. Dabei besteht ein Regelsystem [Mey87] generell aus dem eigentlichen Regler, einer Regelstrecke und einem Messgeber, wie in Abbildung 2.7 skizziert ist. Der Regler erhält eine Führungsgröße w als Eingabe, die einen Sollwert darstellt. Ebenso erhält der Regler den aktuellen Istwert x über den Messgeber. Der Istwert, auch Regelgröße genannt, wird über einen Sensor ermittelt. Aus der Differenz zwischen Soll- und Istwert ermittelt der Regler die Stellgröße y als Ausgabe, mit der auf die Regelstrecke eingewirkt wird. Der Sollwert soll dabei „*so schnell, so genau und so schwingungsarm wie möglich*“ [Mey87] erreicht werden. Auf die Regelstrecke wirkt sich eine Störgröße z aus, die den Istwert verändert oder aus Gründen des Produktionsprozesses einen anderen Sollwert nötig macht. So ergibt sich eine neue Differenz zwischen dem Soll- und Istwert, die möglichst effizient ausgeglichen werden muss.

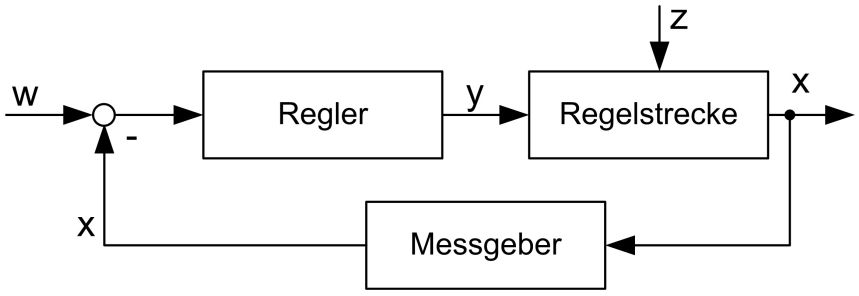


Abbildung 2.7: Prinzipdarstellung eines Regelsystems [Mey87]

Der kaskadierte Regelkreis der Antriebsregelung kann an verschiedenen Stellen aufgespalten werden. Die Daten werden dabei über ein Netzwerk übertragen. Im Folgenden wird jede Stufe des kaskadierten Regelkreises der Antriebsregelung kurz skizziert, um im Anschluss daran generelle Probleme zu diskutieren, die bei einer Aufspaltung des Regelkreises und der Übertragung der Daten in einem Netzwerk entstehen.

2.1.2.1 Stromregelung

Die *Stromregelung* begrenzt den Spitzen- und Dauerstrom des jeweiligen Antriebes, damit dieser im Rahmen seiner Spezifikation und auch im Rahmen der Anlagenmechanik arbeitet [Jet01]. Als Stromregler wird dabei meist ein Regler mit einem Proportional- und einem Integralteil (PI-Regler) verwendet. In Abbildung 2.8 ist ein Gleichstromantrieb skizziert, auf dessen Leistungsteil die Änderung des Stromes übertragen wird. Im Falle eines Gleichstrommotors wird dabei ein Stromrichter verwendet und im Falle eines Wechselstrommotors ein Frequenzumrichter. Der Regelkreis wird geschlossen, indem der Istwert des Motorstroms abgegriffen und als Regeldifferenz mit seinem Sollwert verglichen wird.

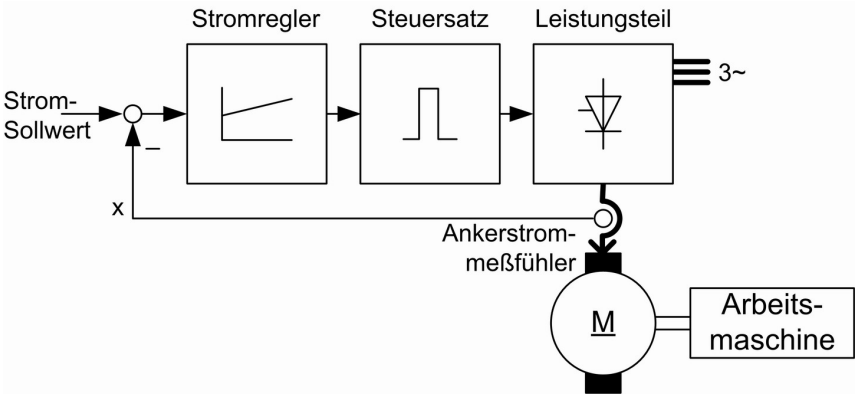


Abbildung 2.8: Stromregelung eines Gleichstromantriebes [Hof79] [Mey87]

Der Sollwert des Stroms stellt dabei gleichzeitig die Sollvorgabe des Drehmomentes dar, mit welcher der Antrieb fahren soll. Daher wird dieser Regelkreis auch als Momentenregelung [Sch05] bezeichnet. Die Zykluszeit für die Stromregelung sollte $\leq 1ms$ sein, um einen störungsfreien Betrieb zu gewährleisten, was für eine Übertragung in einem Netzwerk eine harte Zeitvorgabe darstellt.

2.1.2.2 Drehzahlregelung

Der zweite Teil der Regelung verwendet ebenso wie die Stromregelung einen PI-Regler und liefert als Ausgabe den Stromsollwert für die Stromregelung. Dieses etablierte Verfahren wird als „*Drehzahlregelung nach dem Stromleitverfahren*“ [Mey87], „*Kaskadenregelung der Drehzahl*“ [Mey87] oder als Geschwindigkeitsregelung [Sch05] bezeichnet. Moderne Frequenzumrichter sind in der Lage, direkt über den Ankerstrom des Antriebs die Istdrehzahl zu ermitteln und eine sensorlose Regelung [Aic04] durchzuführen. Die Zykluszeit für eine Aktualisierung der Drehzahl beträgt bis zu $10ms$. In Abbildung 2.9 wird die Istdrehzahl über ein Tachodynamo ermittelt und in den Regelkreis eingespeist.

2.1.2.3 Positionsregelung

Der Drehzahlsollwert wiederum ist die Ausgabe des dritten Teils der Kaskade, der *Positionsregelung*. In einem Positioniervorgang muss ein Antrieb, z. B. der Antrieb für die z-Positionierung eines Roboterarms, „*in einer gegebenen Zeit von einer Lage α_1 in eine Lage α_2 (Punktsteuerung) oder entsprechend einer gegebenen Funktion $\alpha(t)$ (Bahnsteuerung)*

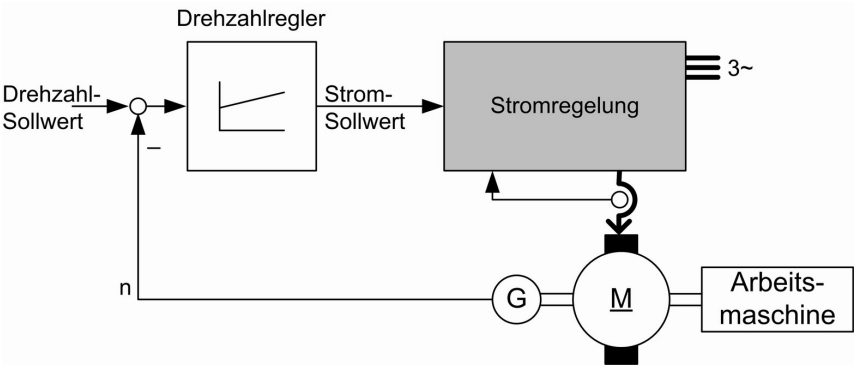


Abbildung 2.9: Drehzahlregelung eines Gleichstromantriebes [Mey87]

zu führen“ [Mül94] sein. In dem Positionsregler wird nun „*die Verfahrstrecke definiert, die festlegt, welche Strecke in positiver und negativer Richtung verfahren werden kann*“ [Jet01]. Die Überwachung der Istposition erfolgt durch eine geeignete Sensorik. Dabei kann es sich um Abstandssensoren und/oder Endschalter handeln, welche verhindern, dass die am Antrieb angeschlossene Arbeitsmaschine zulässige Grenzpositionen überschreitet. Dies würde in der Regel zu einer Beschädigung der Anlage führen oder sogar Menschenleben gefährden: „*Soll beispielsweise eine hohe Positioniergenauigkeit erreicht werden, muss die Zeit zwischen der Flanke des binären Eingangssignals im Sensor und dem Umschalten des Aktors möglichst klein sein. Erst recht entsteht eine solche Anforderung, wenn eine Gefahrensituation vorliegt, wenn das Eingangssignal beispielsweise von einem Grenzwertmelder kommt und darauf praktisch sofort mit dem Schalten eines Aktors reagiert werden muss.*“ [Rei02] Diese Zeiten liegen in der Regel im Bereich von $10ms$ bis $100ms$.

Da die Regelung auf dieser Ebene sehr komplex, anwendungsspezifisch und von einer Vielzahl von Sensordaten abhängig sein kann, werden hier meist mikrocontroller- oder SPS-basierte Lösungen eingesetzt. Abbildung 2.10 skizziert eine Positionsregelung.

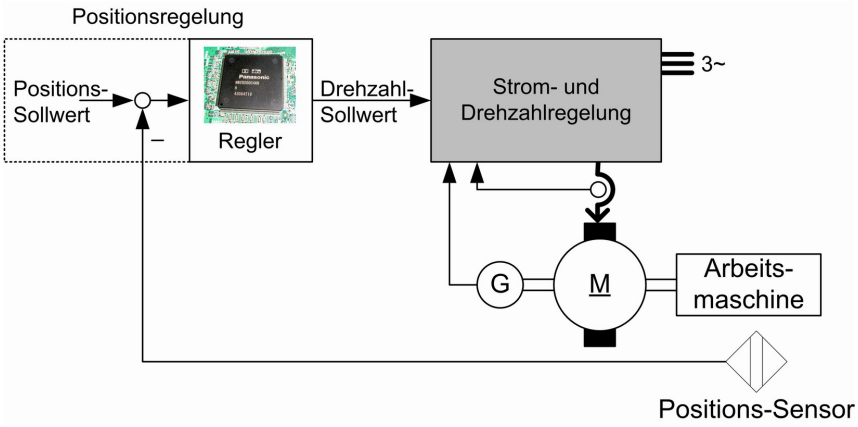


Abbildung 2.10: Positionsregelung eines Gleichstromantriebes [Mey87]

2.1.2.4 Probleme bei der Übertragung in einem Netzwerk

Die beschriebenen Antriebe stehen in Verbindung mit einer zentralen oder dezentralen Steuerung, welche die Gesamtanlage überwacht. Die Daten, welche zwischen der Steuerung und den Antrieben übertragen werden, sind abhängig davon, wo welche Regelung durchgeführt wird. Findet, wie in Abbildung 2.11 dargestellt, die gesamte Regelung in den Antrieben selbst statt, so müssen die Daten im Abstand von $10ms$ bis $100ms$ aktualisiert werden. Ein Beispiel für eine Gesamtregelung innerhalb des Antriebs ist die Technologie Siemens PROFIdrive. Ein PROFIdrive-Positionierantrieb „*enthält zusätzlich zur Antriebsregelung eine Positioniersteuerung. Über Profibus werden Positionieraufträge an den Antriebsregler übergeben und gestartet*“. [Din01]

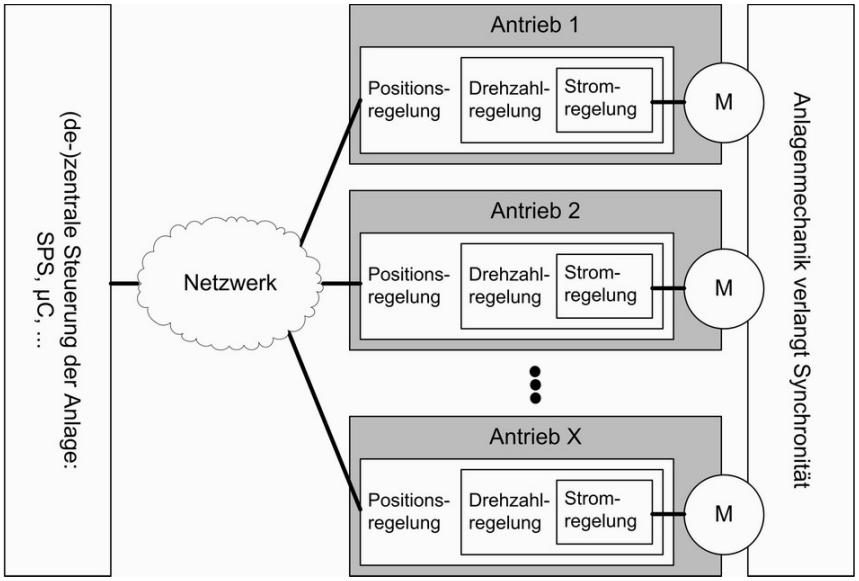


Abbildung 2.11: Integrierte Regelkaskade in den Antrieben

In der Standardvariante enthält ein PROFIdrive-Antrieb lediglich eine integrierte Drehzahlregelung, so dass in diesem Fall die Sollwerte der Drehzahl zyklisch an den Antrieb gesendet werden. Soll eine Positionsregelung durchgeführt werden, so beträgt deren Zykluszeit je nach Anwendungsfall $5ms$ bis $10ms$. Die Anforderungen an die Übertragung der Sollwerte sind demnach im Vergleich zur integrierten Positionsregelung gestiegen. Abbildung 2.12 zeigt den Extremfall, dass die gesamte Regelung in der Anlagensteuerung durchgeführt wird, der Regelkreis also in der Stromregelung aufgebrochen wird. Dadurch muss das Netzwerk die Aktualisierungen der Daten in Zeitbereichen durchführen, die unterhalb von $1ms$ liegen.

Ein weiteres Problem liegt in der hohen Synchronität der Antriebe zueinander. Diese Anforderung resultiert aus der Anlagenmechanik, die gerade bei einer durchlaufenden Warenbahn kritisch ist. „*Für einige technologische Prozesse wird ein Gleichlauf mehrerer Arbeitsmaschinen oder ein festes Drehzahlverhältnis zueinander gefordert. Dies kann ohne Kopplung mechanischer Wellen durch eine elektrische Welle erfüllt werden*“ [Vog98]. In Verbindung mit modernen taktsynchronen Feldbussen verwenden Moning und Lanz sogar den Begriff der „Software-Welle“ [ML06]. Diese Synchronität muss bei einer Übertragung in einem Netzwerk bestehen bleiben. „*Wie bei der Ablaufsteuerung brachte der*

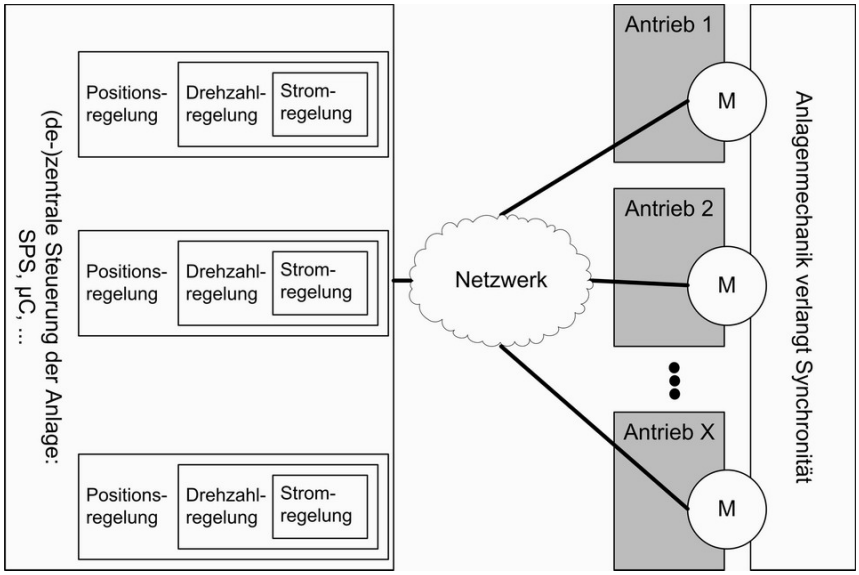


Abbildung 2.12: Von der Anlagensteuerung durchgeführte Regelung

Übergang von der Hardware- zur Software-Lösung aber mit sich, dass jetzt nicht mehr ständig, sondern nur noch zu bestimmten Zeitpunkten die Stellgröße berechnet wird. [...] Für ein korrektes Arbeiten der Regelung ist zwingend notwendig, dass die Abtastzeit exakt eingehalten wird. Irgendwelche Unregelmäßigkeiten [...] führen zu einer Verschlechterung des Regelverhaltens.“ [Rei02] „*Eine Totzeit verschlechtert generell das Regelverhalten, und zwar umso stärker, je größer die Totzeit ist. Die Konsequenz daraus muss sein, die Totzeit möglichst klein zu halten, was [...] durch eine kleine Abtastzeit [...] erreicht wird.*“ [Rei02]

Wie bereits in den vorherigen Kapiteln beschrieben wurde, ist die Antriebsregelung ein Problem der Regelungstechnik. Die Parameter der einzelnen verwendeten Regler sind im Vorfeld bekannt und bleiben relativ konstant, die Streckenparameter sind bekannt und auch vorhandene Totzeiten können bei der Auslegung der Regler berücksichtigt werden.

Ein Problem tritt auf, wenn die Übertragung der Daten, welche die Regelung betreffen, von äußeren Einflüssen abhängig ist. Dies ist beispielsweise der Fall, wenn der Frame, der einen neuen Drehzahlsollwert enthält, durch einen gefüllten Puffer eines Switches (s. Kapitel 2.2.3.2) verzögert wird. Dieses Problem tritt nochmals bei der Istwerterfassung von der Sensorik auf. Der Füllstand des Zwischenspeichers ist wiederum abhängig von anderem Datenverkehr über diesen Switch, der in keinem regelungstechnischen Zusammenhang steht.

Ein weiteres Problem tritt bei einer linienförmigen Verdrahtung auf, die zur Minimierung des Verkabelungsaufwandes in der Automatisierungstechnik bevorzugt zum Einsatz kommt. Dadurch erhalten die angeschossenen Geräte die erforderlichen Daten, welche meist von einer Master-Achse versendet werden, nicht zum selben Zeitpunkt. Dies erschwert den synchronen Lauf der Antriebe. Dadurch werden neue, variable Totzeiten im Drehzahlregelkreis geschaffen. Die Forderung liegt hier in einer ausreichend geringen Verzögerungszeit mit minimaler Schwankung, so dass der Stromregler trotzdem rechtzeitig mit neuen Daten versorgt werden kann. Diese harten Anforderungen an die Echtzeitfähigkeit werden in Kapitel 2.1.3.3 klassifiziert.

2.1.3 Netzwerke in der Automatisierungstechnik

Nachdem die technischen Hintergründe der hohen Echtzeitanforderungen aus der Automatisierungstechnik skizziert wurden, werden die Echtzeitanforderungen in diesem Kapitel spezifiziert. Dazu gehört die Integration der Feldebene mit ihren Sensoren und Antrieben in die Infrastruktur des Unternehmensnetzwerkes ebenso wie grundlegende Definitionen zur Echtzeitfähigkeit und deren Klassifikation. Nach einer Gegenüberstellung der Anforderungen aus dem Büro- mit dem Anlagenbereich erfolgt eine exemplarische Betrachtung eines Feldbusses der Antriebsregelung, der den hohen Echtzeitanforderungen gerecht wird.

2.1.3.1 Kommunikationsmodelle

Größere industrielle Anlagen können nicht mehr durch eine einzelne zentrale Steuerung verwaltet werden. Zu diesem Zweck gliedert man die Anlage in logisch zusammenhängende Anlagenteile und führt eine dezentrale Regelung durch. Die Steuerungen der einzelnen Anlagenteile schließen die Regelkreise der Sensoren und Aktoren, für die sie zuständig sind und synchroniseren sich mit den anderen Teilen der Anlage. *„Dezentrale SPS-Systeme, vorwiegend in der Anlagentechnik, Energieversorgung oder Gebäudeautomatisierung eingesetzt, erfordern eine parallele autonome Bearbeitung einzelner Verarbeitungsaufgaben, geographisch getrennte Verarbeitungsknoten und einen asynchronen Datenaustausch.“* [JT00]

Die Dezentralisierung und Etablierung einer verteilten Intelligenz erfolgt also nicht nur auf der Ebene der Sensorik und Aktorik, sondern auch auf der Ebene der Steuerungen. Damit wird eine industrielle Anlage zu einem *verteilten System*, welches nach der Definition von Löhr, Haustein und Otto [LHO03] aus einer Menge von Prozessoren bzw. Prozesse besteht, die keinen gemeinsamen Speicher besitzen und über Nachrichten kommunizieren. Zusätzlich dazu verfügen die Geräte eines verteilten Systems über keine gemeinsame Uhr [Mei05], so dass eine Uhrensynchronisation (s. Kapitel 2.2.5.2) erforderlich wird.

Die von John und Tiegelkamp definierten Erfordernisse von dezentralen SPS-Systemen setzen ein hohes Maß an Nachrichtenaustausch zwischen den einzelnen Geräten voraus und damit in Verbindung eine geeignete Kommunikationsinfrastruktur. Im Bereich der Netzwerktechnik und der industriellen Automation haben sich drei Arten von Zugriffsverfahren auf Daten etabliert, nämlich das *Client/Server-Modell*, das *Master/Slave-Modell* und das *Publisher/Subscriber-Modell*.

Das *Client/Server-Modell* ist vor allem im Internet und auf auf den Leitebenen der Automatisierungstechnik (s. Kapitel 2.1.3.2) verbreitet. Dabei erfragt sich der Client beim Server die vom Sever im Rahmen eines Dienstes bereit gestellten Daten durch das Abschicken einer Anfrage (Request). Der Server antwortet mit einem Frame, der die angeforderten Daten enthält (Response). Abbildung 2.13 zeigt das Client/Server-Modell am Beispiel eines Zugriffs auf einen HTTP-Server (Hypertext Transfer Protocol).

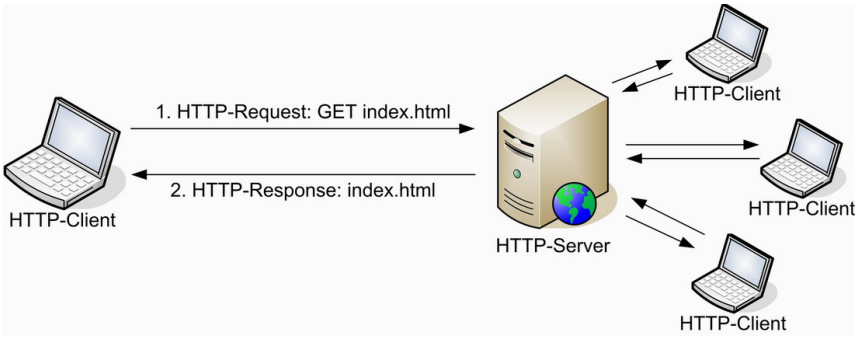


Abbildung 2.13: Client/Server-Modell

Während im Client/Server-Modell eine beliebige Anzahl an gleichberechtigten Geräten im Subnetz existieren, die gleichzeitig anfragen können, lässt das in der traditionellen Automatisierungstechnik vorherrschende *Master/Slave-Modell* nur einen aktiven Master zu einem Zeitpunkt zu. Existieren mehrere Master, so wird der aktive Master in der Regel durch Priorisierung oder durch Token-Vergabe (s. Profibus, Kapitel 2.1.3.6) ermittelt. Auf diese Weise sind die Zugriffe auf das Subnetz planbar.

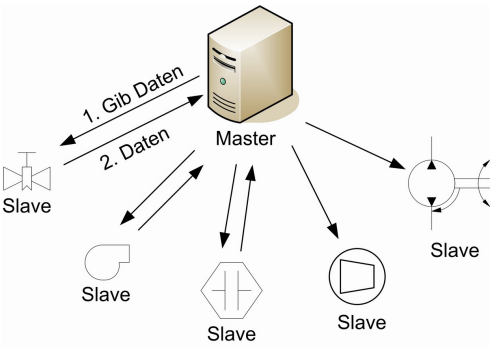


Abbildung 2.14: Master/Slave-Modell

Ein weiterer Anwendungsfall im Umfeld der Automatisierungstechnik liegt darin, dass eine einzelne Hauptachse oder ein Sensor eine größere Anzahl an Geräten gleichzeitig und in regelmäßigen kurzen Zeitabständen mit Daten versorgen soll. Für diesen Fall wird in der Automatisierungstechnik zumeist das *Publisher/Subscriber-Modell* angewendet. „*Ein Sensor veröffentlicht seinen Wert (engl. publish) mit einem Telegramm, das an alle Teilnehmer adressiert ist und die Kennzeichnung des Datenpunktes und den Wert enthält. Jeder andere Teilnehmer kann sich jetzt auf diesen Sensor abonnieren (engl. subscribe) und den aktuellen Wert übernehmen. Er muss dazu noch eine Zeitüberwachung einbauen, um veraltete Informationen selber erkennen zu können.*“ [Fel00] Abbildung 2.15 skizziert diese Idee. Da programmierbare Geräte gleichzeitig Publisher und Subscriber von Daten sein können und damit Teile der Datenauswertung selbst vornehmen können, wird die zentralisierte Verarbeitung von Sensor- und Aktor-Daten insbesondere bei größeren automatisierten Anlagen aufgelöst. Die Verwendung von leistungsfähigen Mikrocontrollern in Verbindung mit dem Publisher/Subscriber-Modell ermöglicht somit den Trend zur *dezentralen Peripherie* [Zha02].

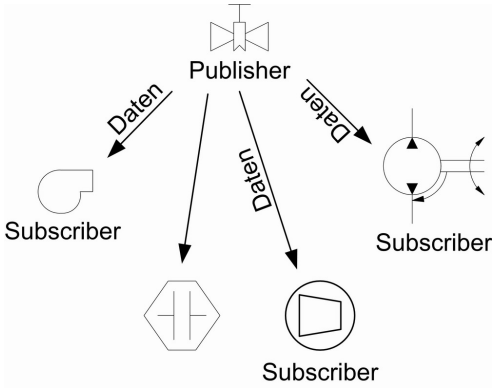


Abbildung 2.15: Publisher/Subscriber-Modell

2.1.3.2 Die vertikale Integration

Die *Pyramide der Automatisierungstechnik* unterscheidet organisatorische Ebenen innerhalb eines produzierenden Unternehmens, deren Begrifflichkeit nicht streng genormt ist. So unterscheiden sich die Definitionen beispielsweise von Varchmin (s. Abbildung 1.2) [Var04], Lauber und Göhner [LG99], Popp [Pop00] sowie von Schnell und Wiedemann [SW06] sowohl in der Anzahl der Ebenen, als auch in der Beschreibung jeder einzelnen Ebene. Im weiteren Verlauf dieser Arbeit werden die von Lauber und Göhner in [LG99] beschriebenen Definitionen der Prozeßführungs- und Leitebenen verwendet, da sie exakt formuliert sind und einen hohen Detailierungsgrad besitzen. Deren Aufteilung der organisatorischen Ebenen ist in Abbildung 2.16 dargestellt.

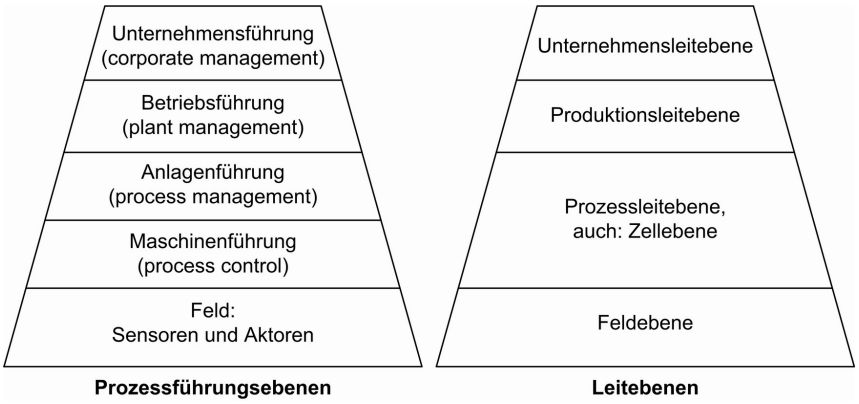


Abbildung 2.16: Pyramide der Automatisierungstechnik [LG99]

Die unterste Ebene stellt stets die *Feldebene* mit den Sensoren und Aktoren, welche Prozessgrößen messen und Antriebe und Ventile steuern, dar. Über der Feldebene existiert die Prozess- und die Produktionsleitebene. Auf der feldnahen *Prozessleitebene* wird die Prozessüberwachung einer Anlage oder eines Anlagenteils - z. B. einer Presse oder einer Gießmaschine - durchgeführt, sowie die Prozessoptimierung und die Fehlerdiagnose und -behandlung. Die Maschinenbediener sorgen dafür, dass die laufende Produktion optimal und störungsfrei durchgeführt wird. Der Fokus dieser Arbeit liegt auf der Feld- und auf der Prozessleitebene. Die *Produktionsleitebene* ist im Gegensatz zur Prozesslei-

tebene bereits globaler ausgerichtet. Hier findet die Kapazitätsplanung statt, z. B. unter Berücksichtigung der Anzahl an zur Verfügung stehenden Mitarbeiter sowie der zur Verfügung stehenden Resourcen, sowie die Terminüberwachung und die Qualitätssicherung der Produktion einer gesamtheitlichen Anlage oder einer gesamten Produktionsstätte. An der obersten Ebene steht das Management des Unternehmens, die Unternehmensleitebene. Hier findet die langfristige Produktionsplanung und statistische Überwachung statt sowie die Auftragsvergabe und Auftragsabwicklung. In diesem Bereich sind Mensch-Maschine-Schnittstellen unter Verwendung von ERP-Systemen vorherrschend.

Die Aussage der Pyramide besteht darin, dass auf jeder Ebene ein für die jeweiligen Anforderungen geeignetes Netzwerk existiert und dass die Ebenen untereinander mit geeigneten Gateways verbunden sind.

Reißenweber [Rei02] unterteilt die dargestellten Geräte in zwei Gruppen. Die Kommunikation und Interaktion mit dem Bediener werden durch die *Anzeige- und Bedienkomponenten* (ABK) realisiert, die sich auf einer höheren Ebene befinden. Alle Geräte zur Prozessbeobachtung und Prozessregelung - also die Sensoren, Aktoren und Steuerungen - werden zu den *prozessnahen Komponenten* (PNK) mit besonderen Echtzeitanforderungen zusammengefasst. Der vorherrschende Trend der *vertikalen Integration* [SL07b] besagt nun, dass es aus mehreren Gründen sinnvoll ist, diese Grenzen der Ebenen aufzuheben.

„*Die Vertikale Integration beinhaltet die Vernetzung der IDA-Geräte (Interface for Distributed Automation) mit der Büro-EDV, dem Intranet und dem Internet.*“ [SW06] Dabei verhalten sich die verschiedenen Geräte und Anwendungen trotz der dezentralen, verteilten Struktur wie eine einzige Applikation. „*Ein Steuerungssystem, bestehend aus dezentral angeordneten physikalischen Geräten, kann virtuell wie eine zentrale Steuerung betrachtet werden.*“ [SW06]

Diese Auflösung der Ebenen begründen Schnell und Wiedemann insbesondere mit der „*transparenten und durchgängigen Kommunikation, sowohl zwischen den Automatisierungsgeräten innerhalb der Anlage, als auch hinein in die Office-Umgebung und das Internet*“ [SW06], also von der Feldebene bis hin zur Unternehmensleitebene.

Nun stellt sich die Frage, warum das in Kapitel 2.2 beschriebene Ethernet trotz seiner Probleme und Unzulänglichkeiten als Lösung für die vertikale Integration, sogar als „*universeller Feldbus*“ [Huh05] propagiert wird. Ein Grund dafür ist die nahezu konkurrenzlose Verbreitung von Ethernet auf der Produktions- und der Unternehmensleitebene. Popp [Pop05] spricht in diesem Zusammenhang sogar von einem Paradigmenwechsel in der Automatisierungstechnik:

„*Während man es vor ein paar Jahren noch nicht für wahrscheinlich gehalten hat, dass Ethernet basierte Kommunikationssysteme in den Feldbereich vordringen, ist dies heute längst Wirklichkeit geworden. Die Grenzen zwischen den einzelnen Ebenen der Automatisierungspyramide haben mitlerweile keine scharfen Kanten mehr. Übrig geblieben sind im Wesentlichen die Ebene der Manufacturing Execution Systems (MES) und der Feldbereich.*“

Ein weiterer Vorteil des Ethernets ergibt sich aus seiner weiten Verbreitung, auch im Rahmen des Home-Office Bereiches. Die benötigte Hardware ist im Vergleich zu meist proprietären Feldbussen in hohen Stückzahlen verfügbar und kostengünstig. Zusätzlich dazu ist die Konfiguration ebenso leicht durchführbar wie die Durchführung von Messungen unter Verwendung eines entsprechenden Protokollanalysators wie Ethereal [Eth07].

Während es für schwache Echtzeitanforderungen bereits Ethernet-Lösungen gibt, ist die Etablierung von Ethernet in der Antriebsregelung problematisch, obwohl bereits einige Lösungsansätze existieren, die im dritten Kapitel dieser Arbeit vorgestellt und diskutiert werden.

2.1.3.3 Die Echtzeitklassen der IAONA

Die *Industrial Automation Open Network Alliance* (IAONA) teilt die Anforderungen der Automatisierungstechnik an die Netzwerke in vier Klassen ein. Abbildung 2.17 zeigt die Anforderungen und die typischen Ebenen in der Pyramide der Automatisierungstechnik. Wie bereits erwähnt, nehmen die Nutzdaten pro Übertragung zur Feldebene hin ab, während die Anforderungen an die Verzögerung und den Jitter drastisch zunehmen.

Echtzeit-Klasse:	1	2	3	4
Anforderungen:	gering	mittel	hoch	sehr hoch
typische Ebene:	Anlagenführung	Maschinen- und Anlagenführung	Feld und Machinenführung	Feld
max. Nutzdaten:	500.000 Byte	500 Byte	32-200 Byte	20 Byte
max. Verzögerung:	5000 ms	500 ms	5 ms	1 ms
max. Jitter:	>1000 µs	100-3000 µs	10-400 µs	1 µs

Abbildung 2.17: Echtzeitklassen der IAONA [Sch03b] [Wis06]

Scheitin [Sch03c] ordnet den Klassen typische Anwendungen zu, die bis hin zur Antriebsregelung mit den härtesten Anforderungen reichen. Schwager [Sch04b] weist darauf hin, dass die Anforderungen der Klasse 1 bereits mit konventionellen Ethernet erreichen lassen. Für die zweite Klasse sind bereits „*optimierte Produkte*“ notwendig. Dies können beispielsweise Switches mit Unterstützung von priorisierten Frames sein. Abbildung 2.18 zeigt, dass dies bereits für einen Großteil der Automatisierungsanlagen ausreichend ist. Mit erhöhten Echtzeitanforderungen steigen auch die Anforderungen an das Netzwerk: „*Mit dem konventionellen Ethernet lassen sich Anlagen, die in die Echtzeitklassen 3 und 4 fallen, nicht realisieren*“ [Sch04b]. Insbesondere für den Bereich der Antriebsregelung ist demnach eine Modifikation des herkömmlichen Ethernets notwendig, so dass isochroner Datenverkehr (s. Kapitel 2.1.3.5) zugelassen wird. Hier liegt also momentan die Grenze der vertikalen Integration.

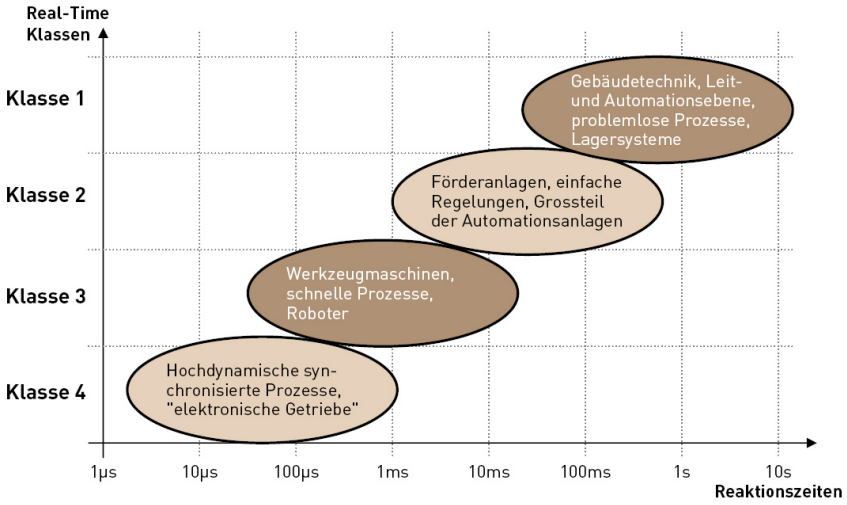


Abbildung 2.18: Echtzeitklassen und typische Anwendungen [Sch03c]

2.1.3.4 Der Begriff der Echtzeitfähigkeit

„*Automatisierungssysteme sind - in Bezug auf die Art der Datenverarbeitung - stets Echtzeitsysteme*“ [LG99]. Lauber und Göhner definieren auch die Hauptforderungen der Echtzeitsysteme, die auch unter extremen Lastbedingungen der industriellen Anlage zu erfüllen sind, mit den Begriffen der

- Rechtzeitigkeit,
- Gleichzeitigkeit,
- Vorhersehbarkeit und
- Verlässlichkeit.

Die *Rechtzeitigkeit* besagt, dass die Erfassung und Auswertung von Prozeßdaten, sowie die geeignete Reaktion stets pünktlich innerhalb einer vorgegebenen Zeitschranke ausgeführt werden muss. Die Art der Zeitschranke kann dabei unterschiedlicher Art sein. Die verschiedenen Arten werden meist in Zeit/Nutzen-Diagrammen festgelegt. Diese stellen dar, zu welcher Zeit eine Information, z.B. ein übertragener Meßwert, von Nutzen für die Anlage ist.

Abbildung 2.19a zeigt das Diagramm der weichen Echtzeit. Hier existiert keine harte Schranke, nach der die Information plötzlich ihren Nutzen verliert. Abbildung 2.19b stellt harte Echtzeit mit einer festen oberen Schranke t_d dar. Diese Anforderung an *Pünktlichkeit* lässt sich erfüllen, indem die Information spätestens bei Erreichen von t_d an die Senke übertragen worden sind. Die härteste Echtzeit, wie sie insbesondere zur Synchronisation in der Antriebsregelung auftritt, ist in Abbildung 2.19c dargestellt. In diesem Fall ist der zeitliche Nutzen einer Information an einen Zeitpunkt t_d gebunden, der zusammen mit einer zulässigen Schwankung eine obere und untere Schranke bildet, wobei diese beiden Schranken zeitlich nah beieinander liegen. Die industrielle Anlage kann in diesem Zusammenhang nicht außerhalb dieser Schwankung auf das Eintreffen eines Signals „warten“. Statt dessen

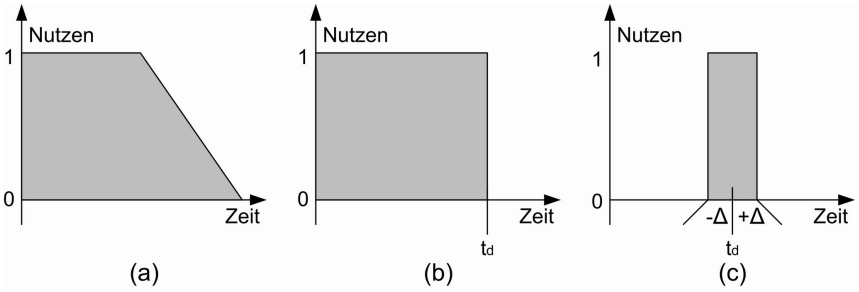


Abbildung 2.19: Zeit/Nutzen-Diagramme der Echtzeit [FHKMS04] [Sch05]

verliert die übertragene Information ihren Nutzen, was zu einem Fehlverhalten der Anlage führen kann. Der *Jitter* ist definiert als die maximal zulässige zeitliche Abweichung $\pm\Delta$ beim Eintreffen der Information an der Informationssenke.

„*Die Forderung nach Gleichzeitigkeit besagt, dass ein Echtzeitsystem in der Lage sein muss, auch auf mehrere gleichzeitig ablaufende Vorgänge des technischen Prozesses zu reagieren. [. . .] Insbesondere wird von jedem Rechenprozess erwartet, dass er alle Operationen vor Ablauf einer in der Einplanung angegebenen Zeitschranke erfolgreich abschließt.*“ [LG99] Die Anforderung nach Gleichzeitigkeit ist insbesondere bedeutend bei der Ausführung von nebenläufigen Rechenprozessen. „*Zeitbedingungen werden üblicherweise in Form von Einplanungen für die Aktivierung, Ausführung und Beendigung von nebenläufigen Rechenprozessen angegeben.*“ [LG99] Diese Anforderung ist jedoch auch innerhalb eines Netzwerkes zu beachten, wenn mehrere Übertragungen gleichzeitig über einen Verteiler verlaufen. Auch in diesem Falle darf der zulässige Jitter nicht überschritten werden.

Ein System ist *vorhersehbar*, wenn alle Reaktionen planbar und deterministisch sind. Es dürfen nur definierte und reproduzierbare Zustände auftreten. Dabei ist besonders von Bedeutung, dass sich keine zufälligen Komponenten im System befinden.

Zur *Verläßlichkeit* zählt die Zuverlässigkeit, Sicherheit und Verfügbarkeit eines Systems. Im Bereich der industriellen Netzwerke bedeutet dies, das ein Netzwerk unter Industriebedingungen ordnungsgemäß arbeitet [KHT00]. Zu den Anforderungen zählen Standhaftigkeit gegenüber elektromagnetischer Belastung, mechanischer und chemischer Beschädigung, Temperatur, Vibration und Feuchtigkeit.

Ein Verstoß gegen eine dieser Forderungen kann zur Fehlfunktion der gesamten Anlage führen und Material- und insbesondere auch Personenschaden mit sich bringen. Der Begriff der *Echtzeitfähigkeit* fasst nun die vier Hauptforderungen der Echtzeitsysteme zusammen. „*Wenn ein System in der Lage ist, unter allen Betriebsbedingungen auf alle Ereignisse korrekt und innerhalb der erwarteten Zeitbeschränkungen zu reagieren, dann ist es echtzeitfähig. Entsprechend, wenn ein Kommunikationssystem alle zeitlichen Anforderungen für den Datenaustausch der Komponenten einer bestimmten Anwendung erfüllt, ist es - bezogen auf diese Anwendung - echtzeitfähig.*“ [Sch05]

Die Verlässlichkeit wird größtenteils auf der Ebene des Übertragungsmediums realisiert und wird durch geeignete Materialen, Abschirmung, Stecker und Kontakte und Hardware-Redundanz erhöht. Maßnahmen zur Erhöhung der Verläßlichkeit stehen nicht im Fokus dieser Arbeit. Die Vorhersehbarkeit, also das Erreichen einer deterministischen Übertragung, ist hingegen ein zentraler Bestandteil, da Ethernet diesen Determinismus nicht von Hause aus bietet. Es sind also Verfahren zu untersuchen, auf welche Weise Determinismus

in das Ethernet integriert werden kann. Ebenso sind Methoden zur Erreichung einer genügenden Rechtzeitigkeit in Kombination mit minimalem Jitter sowie Gleichzeitigkeit im weiteren Verlauf zu diskutieren.

2.1.3.5 Anforderungen an ein Netzwerk der Automatisierungstechnik

Die Definition der Echtzeitfähigkeit besagt nicht, dass ein Netzwerk generell echtzeitfähig oder nicht echtzeitfähig ist. Die benötigte Echtzeitfähigkeit ist eine Frage der Anwendung, also der konkreten industriellen Anlage. Andererseits bedeutet dies, das die restliche freie Bandbreite eines Netzwerks auf der Feldebene für andere Aufgaben verwendet werden kann, solange die Echtzeitfähigkeit für die konkrete Anlage ausreicht.

Im Folgenden werden die Anforderungen der oberen Leitebenen - in welchen Ethernet eine dominierende Rolle einnimmt - den Anforderungen der Feldebene gegenüber gestellt. Während die Größe der Datenframes und die Übertragungszeiten zur Feldebene hin abnehmen (s. Abbildung 1.2), steigen die Anforderungen an Vorhersehbarkeit und Verläßlichkeit. Aufgrund der zyklischen Verarbeitung der PNK-Daten in Kombination mit der Anforderung der Antriebstechnik nach höchster Synchronisation ergibt sich ein isochroner Datenverkehr, der sich in Abständen im Bereich von einigen hundert Mikrosekunden wiederholt und einen geringen Jitter erfordert.

Isochroner Echtzeitverkehr - auch als Isochronous Realtime Traffic (IRT) bezeichnet - ist dann gegeben, wenn jeder Beginn eines Buszyklusses mit höchstmöglicher Genauigkeit startet. Jeder *IRT-Übertragung* wird dabei ein bestimmtes Zeitquantum zugeordnet, wobei sich die Geräte über die Länge dieses Quantums einig sind. Das Netzwerk garantiert eine Langzeitstabilität des Taktes, vgl. [Pop05].

Die zeitlichen Abweichungen, also der bereits definierte Jitter, sind dabei minimal zu halten. Der isochrone Echtzeitverkehr erfüllt aufgrund seiner Definition genau die Anforderungen an die in Abbildung 2.19c dargestellte harte Echtzeit, so dass auch die Anforderungen bei den Positioniervorgängen der Antriebstechnik erfüllt werden können. Netzwerke mit isochronen Echtzeitverkehr gelten als ein Lösungsansatz zur Erreichung der härtesten vierten IAONA-Echtzeitklasse.

„*Die sonst bei allen Bussen übliche Fehlererkennung bzw. Korrektur ist [...] bei isochroner Übertragung nicht vorgesehen.*“ [SW06] Ebenso wie bei der Übertragung von Audio- oder Videodaten ist eine Neuübertragung im Fehlerfall sinnlos, da die Information durch die Zeitverzögerung bereits ihren Nutzen verloren hat. Popp [Pop05] fasst den Umgang mit isochronen Daten im Kontext von Siemens ProfiNet (s. Kapitel 3.1.3.1) zusammen:

„*IRT basiert dabei auf zeitlich geplanten Übertragungswegen. Dafür wird für die Planung der Kommunkation die Kenntnis der Netzwerkinfrastruktur vorausgesetzt. Die Infrastruktur hat Einfluss auf die Eigenschaften der Kommunikation zwischen den IRT-Geräten hinsichtlich Performance und Synchronisationsgenauigkeit. Änderungen an der Infrastruktur bewirken Änderungen der Systemeigenschaften. Dies muss bei der Projektierung berücksichtigt werden. [...] Da die IRT-Kommunikation höchste Anforderungen an zwischen den beteiligten Feldgeräten abgestufte Zeitplanung stellt, ist die Synchronisation aller IRT-Geräte auf ein gemeinsames Taktsystem unbedingte Voraussetzung.*“

Für die in Abbildung 2.19b dargestellte Echtzeit ist ein isochrones Verhalten des Netzwerkes nicht unbedingt notwendig. Daher werden Datenframes mit Echtzeitbedingungen, die auch ohne isochronen Datenaustausch erfüllt werden können, als *Echtzeitverkehr* bzw.

als Realtime Traffic (RT) bezeichnet. In der Automatisierungstechnik treten die Kommunikationsanforderungen mit lediglich einer oberen Schranke zwar auch meist zyklisch auf, sie können jedoch meist bereits mit der Einführung einer VLAN-Priorisierung (s. Kapitel 2.2.1.4) erfüllt werden.

Kommunikation, welche keine Anforderungen an die Echtzeitfähigkeit stellt, wird im Folgenden als *asynchroner Datenverkehr* bzw. als non-Realtime Traffic (nRT) bezeichnet. Dabei kann es sich sowohl um Übertragungen handeln, die unabhängig von der automatisierten Anlage erfolgen und nur deren Netzwerk als Medium verwendet, z. B. das Abrufen von e-Mails an einem freien Ethernet-Port der Anlage. Andererseits können auch einzelne Gerätedaten asynchron ausgelesen oder konfiguriert werden. nRT-Traffic verwendet dabei zumeist den TCP(UDP)/IP/ETH-Protokollstapel.

Auf den Leitebenen kann prinzipiell jedes Gerät direkt mit jedem anderen Gerät kommuizieren. Es existieren jedoch einzelne Server oder Gateways, welche zentrale Punkte der Kommunikation darstellen. Die Feldebene hingegen war über einen langen Zeitraum von einer Master/Slave-Architektur geprägt, wie sie z. B. das Aktor/Sensor-Interface (AS-i) besitzt [KM99]. Bereits Profibus [Pro02] erfüllt jedoch seit seiner Standardisierung 1991 nach DIN 19245 die Multimaster-Fähigkeit, ebenso der Controller Area Network Bus (CAN-Bus) und PNet-Bus mit maximal 32 Mastern [Eng00]. Dieser wurde 1984 entwickelt und teilt jedem einzelnen Master nacheinander die Sendeberechtigung mittels eines Token zu. Wird eine Nachricht zu einem Slave gesendet, so antwortet er unmittelbar. Neuere Feldbusse wie Profibus DP/DPv1 [Pop00] (Dezentrale Peripherie, Version 1) mit der Funktionserweiterung der weit verbreiteten Norm IEC 61158 [Fel02] gehen noch einen Schritt weiter und definieren den Begriff des Querverkehrs, denn „*aus Performancegründen müssen bei elektrisch gekoppelten Achsen Sollwerte auch direkt zwischen den Antrieben ausgetauscht werden.*“ [Wen03] „*Unter dem Begriff Querverkehr versteht man das außenden von Daten eines Slaves [. . .] an einen oder mehrere Slaves.*“ [Pop00]

Die Slaves werden also in die Lage versetzt, direkt Daten miteinander auszutauschen. Im Falle von Profibus geschieht dies unter der vollen Kontrolle des Masters. Dieser weist den sendenen Slave an, seine Daten als Broadcast zu versenden. Auf diese Weise wird verhindert, das die Information vom Slave zuerst zum Master und dann im nächsten Zyklus erst zu den Empfängern weiter gereicht wird.

Neben der Art des Datenaustausches ist noch die Netzwerkinfrastruktur zu betrachten. Während sich auf den Leitebenen durch die Verwendung von Switches eine baumförmige Topologie etabliert hat, ist auf der Feldebene keine einzelne Topologie dominierend. Um den Aufwand der Verkabelung zu minimieren, werden Geräte in einer Linientopologie hintereinandergeschaltet. Ein einzelner Defekt in der Verkabelung führt sowohl bei der Linien-, als auch bei der Baumtopologie zu einem Stillstand der Anlage, da der Ausfall der Anbindung eines einzelnen Gerätes in der Anlage nicht geduldet werden kann. Auch wenn die Produktion fortlaufen könnte, würde ein solcher Ausfall in der Regel zu einer deutlichen Qualitätsverminderung des erzeugten Produktes und damit zu Ausschuß und hohen Materialkosten führen. Eine Redundanz liefern (mehrfache) ringförmige Topologien. Hier ist zu beachten, dass der alternative Weg bei einem Ausfall einer Ringkomponente die Anforderungen an die Echtzeitfähigkeit auch noch erfüllen muss.

Während auf den Leitebenen meist ein Teil der Netzwerkinfrastruktur in das Gebäude integriert ist, z. B. eine Glasfaserleitung als Backbone quer durch alle Stockwerke

des Bürogebäudes, so können dennoch zusätzliche Verteiler und Geräte in das Netzwerk oder in Subnetze integriert werden. Auf der Feldebene hingegen ist die Verdrahtung der echtzeitfähigen Geräte statisch. Eine Änderung ist zumindest mit einem Neustart der automatisierten Anlage verbunden, meist sogar mit einer Änderung in deren Konfiguration. Andererseits ist ein Hinzufügen oder Entfernen von Geräten innerhalb einer Anlage unüblich. Abbildung 2.20 fasst die Anforderungen der Leitebenen und der Feldebene zusammen.

Bürobereich Leitebenen	Anlagenbereich Feldebene
Datenfluß	
keine oder geringe Echtzeitanforderungen	harte Echtzeitanforderungen
große Datenframes (Dokumente, Bilder,...: kByte/MByte-Bereich)	kleine Datenframes (Soll-/Ist-Werte: einige Bytes)
Übertragungszeit im sek-Bereich	Übertragungszeit im µs-Bereich
unregelmäßger, asynchroner Datenverkehr	meist zyklischer, isochroner Datenverkehr
vorwiegend dezentrale Kommunikation, ggf. einzelne Server und Gateways	Master/Slave-Kommunikation vorherrschend, Trend zur Dezentralisierung vorhanden
geringe bis mittlere Vorhersehbarkeit	sehr starke Vorhersehbarkeit
Infrastruktur	
baumförmige Netztopologie	flexible Topologie (stern-/baum-/ring-förmig)
feste Grundinstallation im Gebäude	anlagenabhängige Verkabelung
variabler Geräteanschluß	statische Anzahl und Position der echtzeitfähigen Geräte
geringe bis mittlere Verläßlichkeit	hohe Verläßlichkeit

Abbildung 2.20: Anforderungen aus dem Büro- und Anlagenbereich [Sch05]

2.1.3.6 Ein Feldbus der Antriebsregelung

Die Struktur von gängigen Feldbussen wird in der Norm IEC 61158 nach den OSI-Schichten 1, 2 und 7 organisiert. Die Profile und Protokolle der Busse sind darauf aufbauend in der IEC 61784 organisiert, eine Übersicht dazu liefert Felser [Fel02]. In diesem Kapitel wird exemplarisch ein konventioneller Feldbus vorgestellt, der den Anforderungen der härtesten IAONA-Echtzeitklasse genügt und für die Antriebsregelung geeignet ist. Dabei ist die Frage von Interesse, mit welchen Mitteln diese hohe Echtzeitfähigkeit erreicht wird.

Profibus Dezentrale Peripherie, Version 2 (DPv2) ist eine Erweiterung von Profibus DP, der aufgrund seiner Flexibilität und seiner Integration in Siemens S7 Steuerungen eine große Bedeutung erlangt hat. Im Folgenden wird zunächst Profibus DP vorgestellt und im Anschluss daran die Erweiterungen zu Profibus DPv2. Profibus DP ist nach DIN 19245, Teil 3, Euronorm EN 50170 und IEC 61158 genormt [Fel02], erreicht eine Übertragungsgeschwindigkeit bis zu $12 Mbit/s$ und erlaubt maximal 124 Geräte über alle Bus-Segmente und je Bussegment maximal 32 Geräte. Die Segmente werden in einer Linientopologie über eine geschirmte verdrillte Zweidrahtleitung oder über Lichtwellenleiter durch die Verwendung von Repeatern miteinander verbunden [KHT00]. Profibus DP ist

multimasterfähig, wobei der Buszugriff der Master untereinander über ein Token-Passing Verfahren erfolgt. Besitzt ein Master das Token, so erfolgt ein Master/Slave-Zugriff durch Polling zu den anderen Geräten am Bus, wie in Abbildung 2.21 skizziert ist.

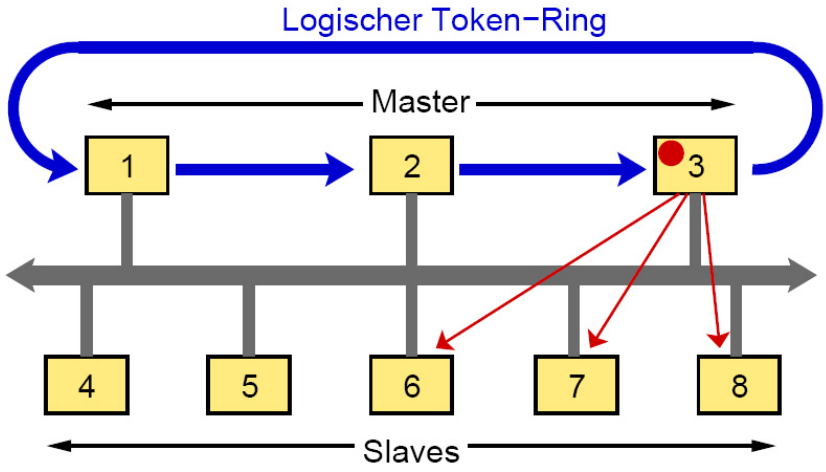


Abbildung 2.21: Buszugriff mit Profibus DP [Wis06]

Während der Multimaster-Betrieb in der Regel für Diagnosezwecke vorbehalten ist, erfolgt der Normalbetrieb nur mit einem Master und Polling [KHT00]. Die Slaves senden nur auf Anfrage durch den Master, so dass die Slaves günstig hergestellt werden können. Ist ein Master mit 10 Slaves am Bus, so kann bei einer Übertragungsgeschwindigkeit von $12 Mbit/s$ eine Zykluszeit von $1ms$ erreicht werden, wodurch eine große Nähe zu den Anforderungen der Antriebsregelung entsteht. Die Buslänge darf dabei $100m$ nicht überschreiten. Abbildung 2.22 zeigt die Zykluszeiten in Abhängigkeit der Slaveanzahl und der Übertragungsgeschwindigkeit. Jeder Slave besitzt jeweils $2 Byte$ Eingangs- und Ausgangsdaten.

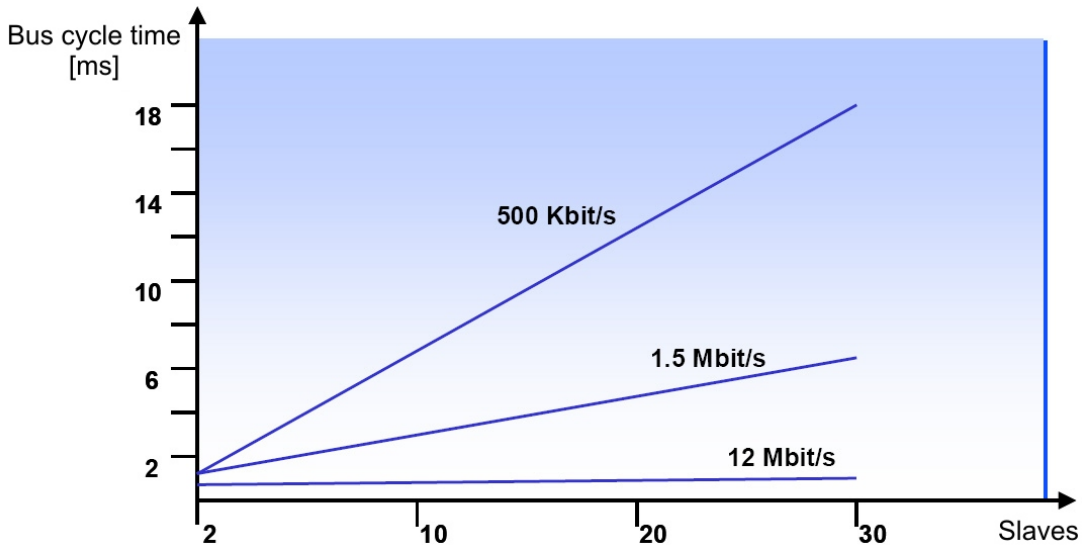


Abbildung 2.22: Zykluszeiten in Profibus DP [Pro02]

Die Datenübertragung erfolgt durch die Verwendung von Universal Asynchronous Receiver/Transmitter Zeichen (UART). Dabei kann es sich um Informationsframes ohne einen Nutzdatenanteil oder um Datenframes handeln [Som04]. Die Arten der Frames werden durch das Startzeichen unterschieden. Informationsframes besitzen eine feste Länge von $6Byte$.

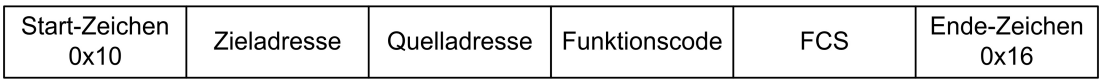


Abbildung 2.23: Imformationsframe von Profibus DP [Som04]

Direkt nach dem Startzeichen wird bereits die ein Byte große Zieladresse angegeben, so dass eine Weiterleitung schon nach dem zweiten Byte erfolgen kann. Der Funktionscode beinhaltet die eigentliche Information. Die Prüfsumme (FCS) wird unter Einsparung von Rechenzeit durch einfaches Aufsummieren der Bytes innerhalb der angegebenen Länge berechnet, wobei ein Überlauf ignoriert wird. Der Master signalisiert den Beginn eines neuen Telegramms mit einer Pause von mindestens 33 Bit, in welcher der Bus in den Ruhezustand versetzt wird.

Frames mit einer festen Datenlänge haben den in Abbildung 2.24 dargestellten Aufbau. Die 8 Datenbytes genügen für den Austausch von Sensor/Aktor-Daten, eine zusätzliche Längenangabe ist nicht erforderlich.

Abbildung 2.24: Datenframe fester Länge von Profibus DP [Som04]

Mit einer variablen Datenlänge können bis zu $246Byte$ an Nutzdaten übertragen werden. Zur Weitergabe des Tokens und zur Bestätigung erhaltener Daten werden sehr kurze Frames verwendet.

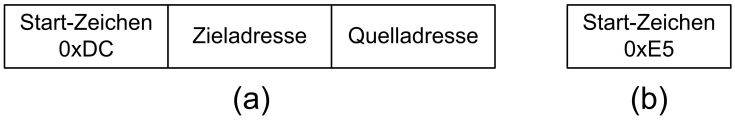


Abbildung 2.25: Frame zur Tokenweitergabe (a) und zur Bestätigung (b) [Som04]

Um die härteste Echtzeitklasse der IAONA zu erreichen und für die Antriebsregelung geeignet zu sein, ist die Zykluszeit von ca. $1ms$ bereits ausreichend. Zusätzlich wurde in der zweiten Version von Profibus DP ein isochroner Buszyklus eingeführt, der eine taktsynchrone Regelung des Masters und der Slaves ermöglicht. Dazu wird ein zyklisches und äquidistantes Taktsignal vom Master an alle Geräte übertragen. Auf dieses Signal können sich die Master und Slaves miteinander synchronisieren mit einer Abweichung, die unterhalb einer Mikrosekunde liegt.

Die Profibus Nutzerorganisation e. V. (PNO) [Pro02] beschreibt diese Synchronisation, die von einer hochgenauen Uhr im Master ausgeht. Der Master sendet dabei vor Beginn jedes Zyklus eine Broadcastübertragung als „global control“, auf den sich alle Geräte am

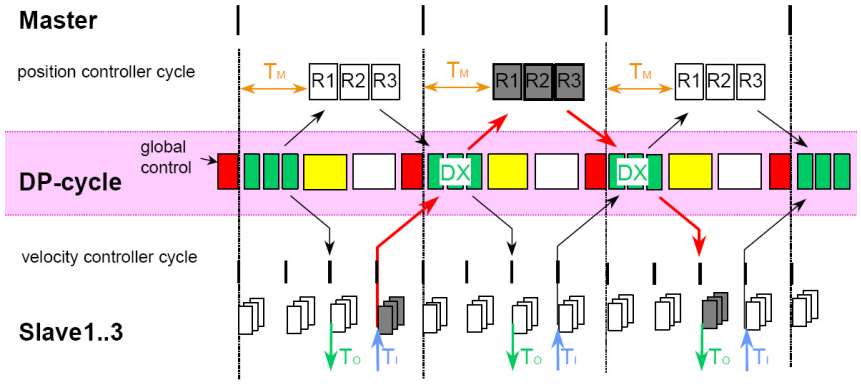


Abbildung 2.26: Isochroner Buszyklus von Profibus DPv2 [Pro02]

Bus synchronisieren können. Abbildung 2.26 zeigt die Zeitfenster für den Datenaustausch (DX), die dem Frame zur Synchronisation folgen. Im Anschluss an die drei DX-Frames wird im anschließenden Zeitfenster der Zugriff eines zweiten Masters gestattet. Vor Beginn des nächsten Zyklus, der wiederum durch ein „global control“-Broadcast initiiert wird, wird eine zusätzliche Reservezeit einkalkuliert. Die Pfeile kennzeichnen den Weg von der Datenerfassung der Istwerte T_I über die Regelung R1, R2, R3 bis hin zur Soll-Datenausgabe T_O. Dieser Weg benötigt in der Regel zwei Buszyklen.

Die *Antriebsregelung* wird von Profibus DPv2 durch die Unterstützung von *Querverkehr* gefördert, die von Moning und Lanz [ML06] beschrieben wird. Beide Erweiterungen sind abwärtskompatibel und wurden so gestaltet, dass alte Geräte die neuen Protokolle ignorieren. Ein Slave (Publisher) veröffentlicht alle seine Variablen per Broadcast, so dass neben dem Master alle anderen Slaves mithören können.

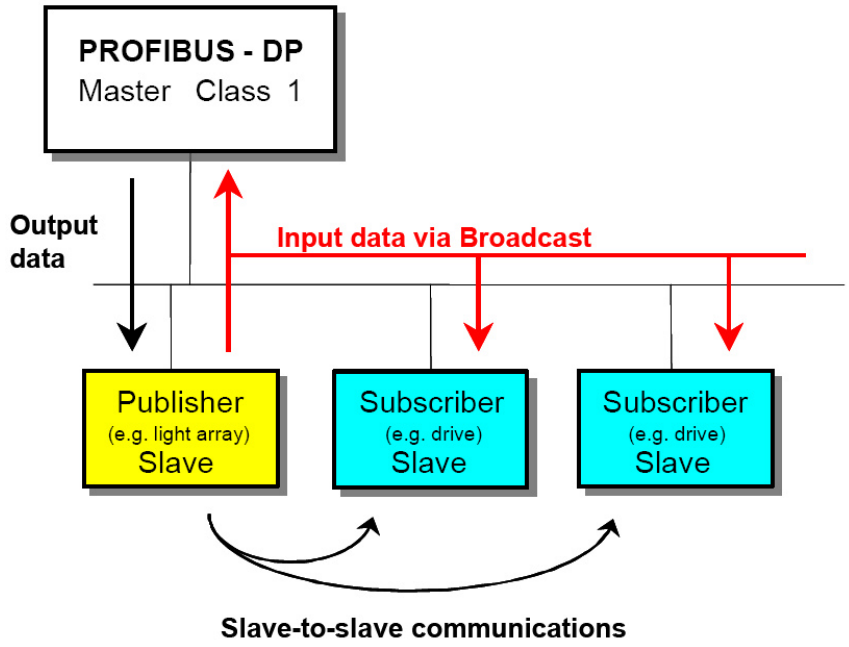


Abbildung 2.27: Querverkehr mit Profibus DPv2 [Pro02]

Jeder Slave, der diese Daten empfangen will (Subscriber), muss eine entsprechende Zeitüberwachung implementieren, um die ankommenden Broadcastdaten korrekt interpretieren zu können [ML06]. Der Querverkehr wird also über ein *Publisher/Subscriber-Modell* realisiert. Diese Kommunikation ist optimal zur Implementierung einer verteilten Regelung und kann z.B. für die Lageregelung bei der Synchronisation von Antrieben eingesetzt werden. Die empfangenden Slaves können die Information des Senders unmittelbar auswerten und ihre eigene Position anpassen; die Regelung erfolgt also dezentral in den intelligenteren Slaves selbst. Man spricht in diesem Zusammenhang in Anlehnung an mechanische oder elektrische Wellen zur Synchronisation von Antrieben von einer „*Software-Welle*“ [ML06]. Auf diese Weise „*werden die Reaktionszeiten am Bus bis zu 90 % reduziert*“ [Pro02].

Als Zusammenfassung ist zu sagen, dass die Bandbreite allein für ein hart echtzeitfähiges Netzwerk unzureichend ist. ProfiNet bietet auf den Einsatz der Feldebene optimierte, kurze Frames an; der Determinismus wird durch Polling mittels des *Master/Slave-Prinzips* erreicht. Die Multimaster-Fähigkeit resultiert aus dem langsameren Token-Passing Verfahren. Zusätzlich zu der für einen Feldbus hohen Bandbreite erreicht Profibus DPv2 durch die Einführung von isochron getakteten und vom Master verwalteten Zyklen die Anforderungen der Antriebstechnik. Zusätzlich erhöht das Prinzip der „Software-Welle“, umgesetzt durch Broadcastübertragungen nach dem Publisher/Subscriber-Prinzip, die Möglichkeiten dieses Feldbusses in der Antriebstechnik.

2.2 Der Ethernet-Standard

Im Home-Office Bereich ist der Ethernet-Standard heutzutage allgegenwärtig. Nahezu jeder Privathaushalt besitzt ein $100 Mbit/s$-Netzwerk [Ada01] [Kau02], an dem PCs und Laptops angeschlossen sind. Über preisgünstige DSL-Anbindungen sind diese privaten Netze an das Internet angebunden, in dem die gleiche Technologie vorherrscht: Der Datenaustausch innerhalb eines solchen Local Area Networks (LAN) erfolgt mit Hilfe von Ethernet-Frames. Nahezu jeder Anwender ist in der Lage, ein solches Netzwerk zu installieren und zu konfigurieren.

Diese Technologie ist ebenso nahtlos in den Büros von Unternehmen im Umfeld von automatisierten Anlagen integriert. Mitarbeiter nutzen das Intranet eines Unternehmens im Rahmen von Business-to-Employee Portalen (B2E), zur Anbindung an ERP-Systeme oder lediglich zum Abrufen ihrer e-Mails. Produktionsaufträge können eingegeben werden, der Status der aktuellen Produktion kann eingesehen und überwacht werden. Ein Beispiel für ein solches MES ist der HYDRA-Leitstand der MPDV Mikrolab GmbH [MPDV07] [Klu07]. Der Ethernet-Standard ist also bereits mit automatisierten Anlagen verbunden. Dennoch hat Ethernet die Automatisierungstechnik nicht in der Form durchdrungen, wie es in anderen Gebieten der Fall ist: Die Feldbusse sind immer noch dominierend [MK04].

In diesem Unterkapitel werden die Grundlagen des Ethernets beschrieben, wobei besonders auf die Eigenschaften des Ethernet-Standards eingegangen wird, welche Einfluss auf die Anforderungen der Automatisierungstechnik besitzen. Das Ziel liegt darin, existierende Feldbusse abzulösen oder zumindest die Integration der Feldbusse mit dem Ethernet-Standard zu erhöhen.

2.2.1 Ethernet-Frames

Um den Ethernet-Standard in der Automatisierungstechnik einzubringen, muss dieser Standard zunächst betachtet werden. Dazu zählt die im Ethernet typische Art der Adressierung der Geräte ebenso wie die eingesetzten Protokolle und Verfahren. Zusätzlich ist es von Bedeutung, die Eigenschaften der eingesetzten Ethernet-Hardware in Bezug auf die Anforderungen der Automatisierungstechnik zu beschreiben.

2.2.1.1 Geräte

Ethernet-Frames werden grundsätzlich zwischen Geräten ausgetauscht. Ein *Gerät* bezeichnet dabei im Allgemeinen einen Teilnehmer des Netzwerkes, der genau ein Ethernet-Interface besitzt und eine Quelle und/oder Senke für Nachrichten darstellt. Dabei kann es sich um ein echtzeitfähiges Gerät handeln, welches in die automatisierte Anlage als Sensor oder Aktor eingebunden ist. Andere Geräte, wie angeschlossene PCs oder Laptops, werden als nicht-echtzeitfähige oder auch asynchron sendende Geräte bezeichnet.

2.2.1.2 Adressierung im Ethernet

Damit ein sendendes Gerät den oder die Empfänger spezifizieren kann, müssen diese adressiert werden. Die Adressierung im Ethernet erfolgt durch die Angabe von *Media-Access-Control Adressen* (MAC) auf OSI-Schicht 2. Dabei wird jedem Ethernet-Interface genau eine weltweit eindeutige, $6Byte$ große MAC-Adresse zugeordnet. Diese Adresse beinhaltet die Hersteller-Identifikation des Ethernet-Gerätes, welche als Organizationally Unique Identifier (OUI) bezeichnet wird. Innerhalb der OUI befindet sich ein Bit, welches die Adresse als Unicast- oder als Gruppenadresse ausweist. Des Weiteren existiert ein weiteres Bit zur Unterscheidung von lokal verwalteten Adressen für private Netze und global eindeutigen Adressen, die von dem Institute of Electrical and Electronics Engineers (IEEE) vergeben werden. [Pla05]

Die unteren 24 Bit einer MAC-Adresse beinhalten eine eindeutige Adapter-Identifikation, also eine eindeutige Nummer für jede einzelne Netzwerkkarte dieses Herstellers. Die Adapter-Identifikation wird vom Hersteller eines Ethernet-Interfaces frei vergeben.

Die Adressierung von Empfängern kann im Ethernet auf drei verschiedene Weisen erfolgen. So beschreibt eine *Unicastübertragung* die Versendung eines Frames von einem Ethernet-Interface zu genau einem anderen Ethernet-Interface. Es handelt sich also um eine 1:1-Kommunikation. Eine Unicastkommunikation im Ethernet wird durchgeführt, indem man die Quell-MAC Adresse des Senders und die eindeutige Ziel-MAC eines Empfänger-Interfaces im Ethernet-Frame angibt. Das Bit zur Indikation der Gruppen-Adresse darf dazu in der OUI der Ziel-MAC nicht gesetzt sein. Damit wird signalisiert, dass es sich bei dem Zielgerät um eine Adresse eines einzelnen Gerätes handelt.

Eine *Broadcastübertragung* beschreibt hingegen die Versendung eines Frames von genau einem Ethernet-Interface zu allen anderen Geräten innerhalb einer Broadcastdomäne. Es handelt sich um eine 1:n-Kommunikation, wobei n die Anzahl der Endgeräte in der Broadcastdomäne des Senders darstellen. Eine *Broadcastdomäne* bezeichnet den Teil eines Netzwerkes, der beim Versenden einer Broadcastnachricht durch ein Gerät dieses Netzes erreicht wird. Broadcastnachrichten werden nicht über Router auf OSI-Schicht 3

weiter geleitet, jedoch grundsätzlich über Switches und Hubs. Eine Broadcast-Kommunikation im Ethernet ist gekennzeichnet durch Angabe der reservierten Ziel-MAC Adresse FF:FF:FF:FF:FF:FF.

Eine *Multicastübertragung* beschreibt die Versendung eines Frames von genau einem Gerät zu einer bestimmten Menge von anderen Geräten oder an eine geschlossene Gerätegruppe, die als m bezeichnet wird. Bei einer Multicast-Kommunikation handelt es sich also um eine 1:m-Kommunikation mit $m \leq n$, wobei n wiederum die Anzahl der Endgeräte in der Broadcast-Domäne des Senders darstellt.

Die Ethernet-Frames werden bei einer Multicastübertragung an den betreffenden Verteilern der Route vervielfältigt. Als Zieladresse wird eine spezielle Multicast-Adresse angegeben, für die jedes betreffende Gerät registriert sein muss. Wie auch bei der Broadcast-Adresse ist bei Multicast-Adressen das Bit der Gruppen-Adressierung in der Ziel-MAC gesetzt.

Multicast auf der Ethernetebene ist nicht weit verbreitet. Die IEEE hat den Adressbereich 01:00:5E:xx:xx:xx für Ethernet-Multicasting reserviert [Cho00], wodurch 24 Bit für die Multicast-Adressierung verwendet werden können. Es wurde in der Requests for Comments (RFC) 1112 [Dee89] standardisiert, dass IP Multicast-Adressen der Klasse D, welche die IP-Adressen 224.0.0.0 bis 239.555.555.555 umfasst, in diesen MAC-Adressraum direkt abgebildet werden.

Für die Automatisierungstechnik ist die Adressierung im Multicastverfahren von großer Bedeutung, um eine Anzahl von Geräten nach dem Publisher/Subscriber-Modell (s. Kapitel 2.1.3.1) gleichzeitig anzusteuern. Erfolgt dies nacheinander, so wird dafür mehr Zeit benötigt. Von größerer Bedeutung ist noch, dass in diesem Falle die Übertragungszeit der einzelnen Nachrichten mit berücksichtigt werden muss, damit alle Empfänger zum gleichen Zeitpunkt aktuelle Daten besitzen.

Da Hubs stets alle Frames als Broadcast weiterleiten, findet die Multicast-Adressierung auf MAC-Ebene keine Anwendung. Switches gehen auf verschiedene Weise mit Multicast-MAC-Adressen um, die in Kapitel 2.2.3.2 beschrieben wird.

2.2.1.3 Ethernet-II Frame und Frame nach IEEE 802.3 (Raw)

Bereits 1980 wurde von den Firmen Digital Equipment Corp., Intel und Xerox (DIX) die Ergebnisse ihrer Bemühungen zur Standardisierung des Ethernets in der DIX Ethernet V1.0 festgehalten [Hei98]. Diese Spezifikation wurde in das Local Network Standards Committee, der Arbeitsgruppe 802 des IEEE eingebracht. Parallel dazu vervollständigte das DIX-Konsortium seine eigene Spezifikation, welche nach der Standardisierung der Ethernet-Frames durch die IEEE wiederum angepasst werden musste [Hei98].

Als Ergebnis wurde 1982 das DIX Ethernet V2.0 veröffentlicht, welches unter dem Namen Ethernet-II heutzutage weiter verbreiteter ist als der von der IEEE genormte 802.3-Frame. Da die Standardisierung der Frames durch die DIX-Gruppe und der IEEE in Abstimmung erfolgte und die erste Version des DIX-Standards bereits einen großen Einfluß auf die IEEE-Norm hatte, unterscheidet sich ein Ethernet-Frame nach IEEE 802.3 [IEEE05] kaum vom einem DIX 2.0 Frame.

Ein *Ethernet-Frame* beginnt mit einer $8Byte$ großen *Präambel*, die der Synchronisierung der empfangenen Ethernet-Station auf die Taktrate des Frames dient. Im IEEE-Frame ist die Präambel lediglich $7Byte$ groß, s. Abbildung 2.28. Das achte Byte wird als

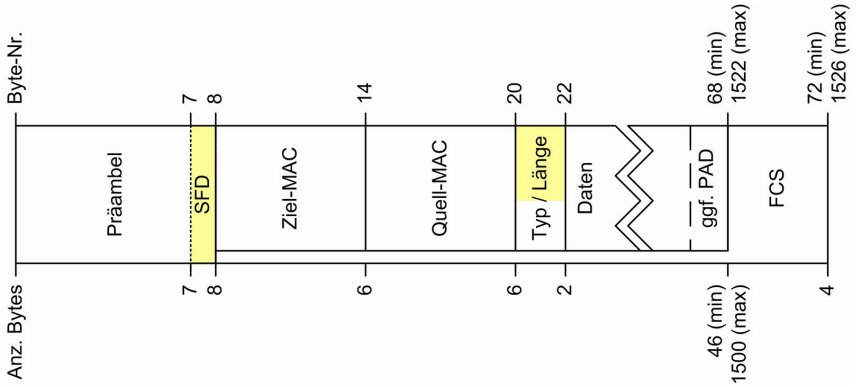


Abbildung 2.28: Ethernet-II / DIX 2.0 Frame und Frame nach IEEE 802.3 (Raw)

Start Frame Delimiter (SFD) bezeichnet. Auf Signalebene ist die Präambel jedoch mit der eines Ethernet-II Frame identisch, so dass die Kompatibilität der beiden Frames auf der Bitübertragungsschicht gewahrt bleibt.

Im Anschluss daran erfolgt für alle Frametypen die Angabe der jeweils $6Byte$ großen MAC-Adressen. Zuerst wird die Ziel-MAC übertragen und dahinter die Quell-MAC.

Den Adressen folgt ein $2Byte$ großes Typfeld, welches das überliegende Protokoll im Datenteil des Ethernet-Frames spezifiziert [Hel98]. Dieses Feld wird in der Literatur oft als *Ethertype* beschrieben [IEEE05b]. Für die Standardisierung ist wiederum die IEEE verantwortlich. Bekannte Werte sind beispielsweise 0800_{Hex} für IPv4 und $86DD_{Hex}$ für IPv6. Viele Hersteller von echtzeitfähigen Ethernet-Lösungen haben sich eigene Ethertypes reservieren lassen, um die Akzeptanz ihres Standards zu erhöhen.

Der einzige Unterschied in der Protokoll-Spezifikation zwischen einem Ethernet-II / *DIX 2.0 Frame* und einem Frame nach *IEEE 802.3 (Raw)* besteht in der Ersetzung des Typfeldes durch ein Längenfeld, s. Abbildung 2.28. Auf diese Weise kann bereits auf Ethernetebene die Anzahl an Nutzdaten ermittelt werden. Die DIX-Gruppe und die IEEE haben sich darauf geeinigt, dass die Normungen der Ethertypes durch die IEEE erst bei 0600_{Hex} beginnt, da aufgrund der Spezifikation eines Frames nach IEEE 802.3 (Raw) bei darunter liegenden Werten die Nutzdatenlänge des Frames angegeben wird. Ab einem Wert von 0600_{Hex} wird der Frame dann automatisch als Ethernet-II Frame identifiziert.

Dem Typfeld folgt der Datenteil des Ethernet-Frames, in dem Protokolle der höheren Schichten gekapselt werden. Um eine sichere Kollisionserkennung durchführen zu können (vgl. Kapitel 2.2.2.1), muss ein Ethernet-Frame eine Mindestlänge von $72Byte$ incl. Präambel besitzen. Da die anderen Felder konstanter Länge sind, muss der Datenteil mindestens $46Byte$ betragen. Sollen weniger Nutzdaten übertragen werden, so füllt die MAC-Schicht die fehlenden Bytes mit einem Padding auf. Da das Padding aus 00_{Hex}-Bytes besteht und keine Angabe der Anzahl der Nutzdaten in der Ethernet-II Spezifikation vorhanden ist, muss die tatsächliche Länge des Paketes im überliegenden Protokoll festgehalten werden. In IPv4 geschieht dies beispielsweise durch eine Längenangabe innerhalb des IP-Headers [KR02]. Des Weiteren existiert eine Obergrenze für die Größe eines Ethernet-Frames, die bei $1526Byte$ liegt. Bei größeren Datenmengen ist das Protokoll der höheren OSI-Schicht für die Segmentierung und Reassemblierung der Pakete verantwortlich.

Für die Automatisierungstechnik ist der Nutzdatenanteil von $46Byte$ in der Regel ausreichend. Problematischer ist der Overhead des Ethernet-Frames, der für einen Nutzdatenanteil von unter 64 % bei einem minimal großen Frame verantwortlich ist [Wes06].

Abgeschlossen wird der Ethernet-Frame durch die $4Byte$ große *Frame Check Sequence* (FCS). Dabei handelt es sich um einen Cyclic Redundancy Check (CRC), der von der Ziel-MAC bis zum Ende des Datenteils gebildet wird [Pla05].

2.2.1.4 Ethernet-Frame mit VLAN-Tagging nach IEEE 802.1p/q

Durch die Einführung eines Taggings können Ethernet-Frames priorisiert bei der Weiterleitung durch Switches behandelt werden. Daduch kann in gewissen Grenzen ein Determinismus für hochpriore Frames erreicht werden, sofern diese in geringer Zahl auftreten. Ein solcher Determinismus ist für die Automatisierungstechnik von großer Bedeutung (s. Kapitel 2.1.2) und wird von einigen Lösungen für echtzeitfähiges Ethernet, beispielsweise von Ethernet/IP (s. Kapitel 3.1.1.1), verwendet.

Dieses *Tagging* wurde ursprünglich in Zusammenhang mit Virtual Local Area Networks (VLANs) eingeführt und ist nach IEEE 802.1q [IEEE06] genormt, kann jedoch auch unabhängig angewendet werden. Dadurch werden die Ethernet-Frames um $4Byte$ nach dem Absenden durch ein Gerät verlängert [Tan03]. Dies hat zur Folge, dass die maximale Länge eines Ethernet-Frames nach IEEE 802.3 innerhalb des Netzwerkes überschritten werden kann. In der Norm IEEE 802.3ac wurde jedoch die Möglichkeit der Erweiterung eines Ethernet-Frames um genau $4Byte$ spezifiziert:

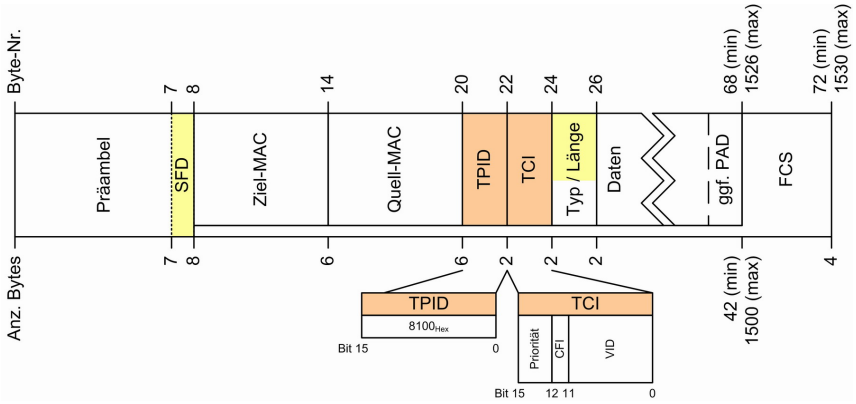


Abbildung 2.29: Ethernet-Frame mit VLAN-Tagging nach IEEE 802.1q

Zunächst wird eine $2Byte$ großer Tag Protocol Identifier (TPID) mit dem festen Wert 8100_{Hex} eingefügt. Diesen Identifier kann als Protokoll-Indikator für VLANs ansehen werden, zumal er an der gleichen Stelle des Frames steht wie des Typfeld eines konventionellen Ethernet-Frames. Die Informationen des VLANs liegen in der Tag Control Information (TCI). Von Interesse sind hier die obersten 3 Bit, die der Priorisierung des Frames dienen. Auf diese Weise bilden sich 8 Prioritätsklassen, die nach IEEE 802.1p genormt sind [IEEE98].

Abbildung 2.30 zeigt bereits bei einer *Priorisierung* in 3 Klassen, dass die Verzögerung für hochpriore Klassen auch bei sehr hoher Netzlast ρ konstant niedrig bleibt. Lediglich Frames niederer Priorität verschlechtern sich in ihrer Zeit zur Weiterleitung erheblich im

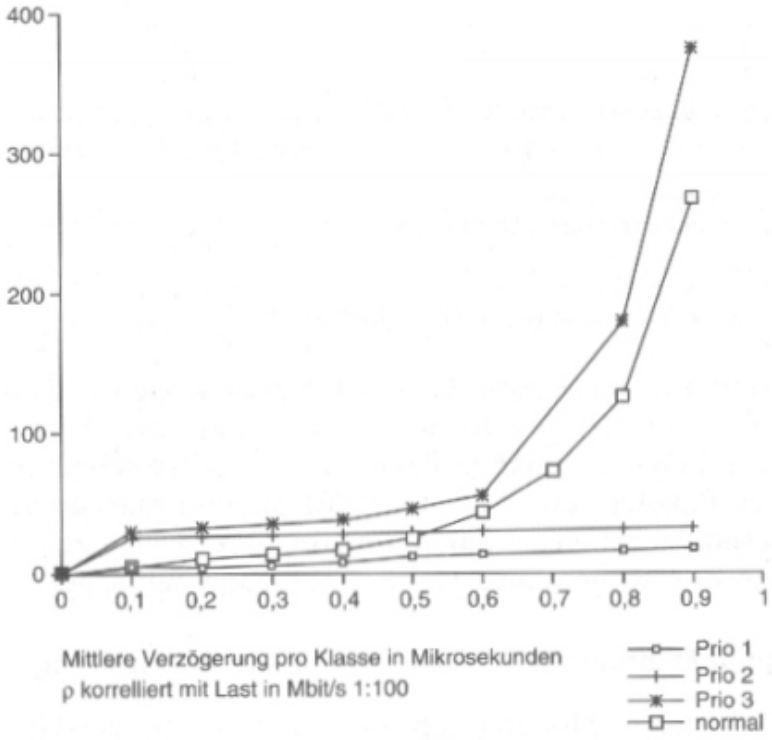


Abbildung 2.30: Verzögerungszeiten in Abhängigkeit der Priorisierung [DT97]

Vergleich zu einer fehlenden Priorisierung. Werden also Echtzeitdaten hoch priorisiert, so können diese auch bei hoher Last mit einer relativ konstanten Verzögerungszeit übermittelt werden. Werden im Netzwerk jedoch fast ausschließlich hochpriore Daten übertragen, wie es auf der Feldebene üblich ist, so verliert die Priorisierung wiederum ihren Effekt.

Ein Problem tritt auch dann auf, wenn gerade mit der Weiterleitung eines großen niederprioren Frames begonnen wurde. Trifft zu diesem Zeitpunkt ein Frame mit hoher Priorität ein, der an den gleichen Port weiterzuleiten ist, so wird dieser zunächst zwischengespeichert. Dies erhöht die Verzögerungszeit bei der Sendung des hochprioren Frames. Wird ein solcher Frame kurze Zeit später nochmals übertragen, so kann er mit hoher Wahrscheinlichkeit direkt weitergeleitet werden. Dies sorgt für eine hohe Schwankung der Verzögerungszeit.

2.2.2 Algorithmen und Protokolle im Ethernet

In diesem Kapitel werden Verfahren beschrieben, die im heutigen Ethernet Verwendung finden und von Bedeutung für diese Arbeit sind. Diese Verfahren können die Einführung von Echtzeitfähigkeit beeinträchtigen, wenn sie nicht-deterministisch ausgeführt werden und die Weiterleitung von IRT-Frames beeinflussen.

2.2.2.1 CSMA/CD und Exponential Backoff

Das CSMA/CD-Verfahren mit der Strategie des *Exponential Backoff* [Zen99] wird lediglich im Halbduplexmodus (HD) des Ethernets [KR02] verwendet. Im *Halbduplexmodus* kann

ein Gerät zu einem Zeitpunkt entweder senden oder empfangen, nicht jedoch gleichzeitig senden und empangen. Im Vollduplexmodus (VD), wie er in Netzen mit Switches üblich ist, ist das Netzwerk kollisionsfrei und das CSMA/CD-Verfahren kommt nicht zur Anwendung. Ein Gerät im *Vollduplexmodus* kann zur gleichen Zeit Daten senden und empfangen.

Bei *CSMA/CD* wird zunächst das gemeinsame Medium abgehört und nur dann mit dem Senden begonnen, wenn das Medium frei ist. Während der Sendung wird die Leitung weiter abgehört. Liegt nicht das gesendete Signal an, so wird eine Kollision festgestellt. Dies führt zu einer unmittelbaren Sendung eines Jam-Signals und diese Sendung wird als misslungen eingestuft. Ist der Sendeversuch 16 Mal erfolglos, so wird die Sendung mit einem Fehler abgebrochen. Ansonsten wird eine Wartezeit unter Verwendung des Exponential Backoff Algorithmus berechnet. Dabei wird die Wahrscheinlichkeit für ein gleichlanges Warten von mehreren Geräten durch die Einführung einer Zufallskomponente verringert. Nach jedem fehlgeschlagenen Versuch erhöht sich die Zeit für eine Neuübertragung, da von einem ausgelasteten Netzwerk ausgegangen wird. Zu dieser Wartezeit wird eine Zufallszahl hinzugefügt, damit zwei Sender sich nicht nach einer einmaligen Kollision bei weiteren Sendeversuchen gegenseitig blockieren.

Jasperneite [Jas02] beschreibt des Weiteren Probleme bei der regulären Ausführung des CSMA/CD-Protokolls. Der Capture-Effekt tritt bei einer kleinen Anzahl an Geräten und gleichzeitiger hoher Last auf. Dabei kann es vorkommen, dass ein einzelnes Gerät direkt aufeinander folgende Pakete versenden kann, obwohl mindestens ein weiteres Gerät auf den Bus zugreifen will. Jasperneite beschreibt diesen Effekt als Monopolisierung durch ein Gerät. Zusätzlich dazu existiert der Effekt des Packet-Starvation. Da mit jedem erfolglosen Sendeversuch die Wartezeit für einen erneuten Sendeversuch steigt, wird das ohnehin verspätete Paket weiter verzögert. Auf diese Weise können einzelne Pakete um das bis zu 100-fache im Vergleich zur mittleren Latenzzeit verzögert werden oder der Sendeversuch wird letztlich ganz eingestellt.

Um eine Kollision zu erkennen, muss ein sendendes Gerät mit CSMA/CD-Algorithmus noch aktiv senden, wenn das auf der Leitung entstehende Jam-Signal wieder bei dem Sender eintrifft. Damit dies gewährleistet ist, ist für jeden Übertragungsstandard eine minimale Framegröße eingeführt worden. Zusätzlich darf ein maximaler Abstand zwischen zwei Geräten innerhalb einer Kollisionsdomäne nicht überschritten werden. Tritt an einem Port eines Switches (s. Kapitel 2.2.3.2) eine Kollision auf, so wird diese nicht an die anderen Ports weiter geleitet. Ein Switch trennt also Kollisionsdomänen.

Zwei Geräte gehören dann zu einer gemeinsamen *Kollisionsdomäne*, wenn sie bei gleichzeitigem schreibenden Zugriff auf das gemeinsame Medium eine Kollision auslösen.

Im Gegensatz zu token-basierten Verfahren kann mit CSMA/CD mit Exponential Backoff also keine obere Zeitschranke angegeben werden, nach der ein Frame auf jeden Fall beim Sender eintrifft. Im Worst-Case werden Versuche zum Versenden eines Frames sogar eingestellt. Die Verbreitung dieses Algorithmus ist neben seiner Einfachheit gegenüber token-basierte Ansätzen darauf zurückzuführen, dass er bei geringer Netzlast einen höheren Datendurchsatz ermöglicht, da ein Sender bei geringer Netzlast nicht auf die Zuteilung einer Sendeberechtigung warten muss. Die Anforderungen der Automatisierungstechnik erfordern jedoch einen *Determinismus*, der den Empfang mit kurzen Verzögerungszeiten und minimalem Jitter garantiert.

2.2.2.2 Das Spanning Tree Protocol (STP)

Netzwerke, bei denen ausschließlich Switches (s. Kapitel 2.2.3.2) verwendet werden, müssen nicht zwangsläufig eine Baumtopologie aufweisen. Statt dessen kann die Netzwerkinfrastruktur beliebige Zyklen enthalten. Damit Ethernet-Frames nicht zyklisch zwischen Switches gesendet werden, unterstützen viele Switches das nach IEEE 802.1d genormte *Spanning Tree Protocol* [Cho00]. Die Frames des komplexen Protokolls [Hel98] [Cho00], welche Switches untereinander austauschen und die als Bridge Protocol Data Units (BPDUs) bezeichnet werden, sorgen für den Aufbau einer aktiven Baumtopologie innerhalb des Netzwerkes. Dabei werden Ports von Switches deaktiviert, die für Zyklen verantwortlich sind. Für die Deaktivierung werden die alternativen Routen untersucht, indem die Übertragungsleitungen gewichtet bzw. anhand einer Kostenfunktion bewertet werden. Letztlich bleibt die Route aktiv, von welcher der höchste Datendurchsatz erwartet wird. Damit das Netzwerk beim Ausfall einer Leitung oder eines Switches nicht vollständig inaktiv wird, werden ca. alle 10 Sekunden Hello-BPDUs von einem ausgewählten STP-Wurzelswitch versendet. Bei fehlerhafter Hardware kann dadurch eine neue aktive Baumtopologie ermittelt werden, wodurch die Fehlerredundanz des Netzwerkes erhöht wird. Die Reorganisation der logischen Topologie kann bei STP mehr als 10 Sekunden in Anspuch nehmen, in denen das Netz nicht oder nur teilweise betriebsbereit ist.

Das nach IEEE 802.1w genormte *Rapid Spanning Tree Protocol* (RSTP) [Cis07] ist eine abwärtskompatible Weiterentwicklung des STP mit geringeren Reaktionszeiten. Die Initialisierung mit dem Aufbau der ersten aktiven Baumtopologie beträgt ca. $400ms$, die Reaktionszeit auf einen Fehler liegt bei ca. $500ms$. Diese Fehlerreaktionszeit kann jedoch nicht garantiert werden. Im Gegensatz zu STP arbeitet bei RSTP die aktive Baumtopologie trotz Ausfall einer bevorzugten Verbindung so lange weiter, bis eine neue logische Baumstruktur im regulären Zeitintervall berechnet ist.

Während der Hochlaufphase einer automatisierten Anlage ist die Ausführung des (R)STP-Protokolls unkritisch, da zu Beginn der Konfiguration der Anlage noch keine deterministischen Anforderungen an das Netzwerk bestehen. Wird ein solches Verfahren jedoch während des laufenden Betriebes selbständig von Switches initialisiert, so ist bereits bei einer Unterbrechung von einigen Millisekunden die Echtzeitfähigkeit des gesamten Netzes nicht mehr gewährleistet. Im Echtzeitbetrieb darf demnach kein Spanning Tree Algorithmus unkontrolliert ausgeführt werden. Daher muss die Ausführung dieses Algoithmus auf die Hochlaufphase der Anlage beschränkt werden. Die Konfiguration der Ausführung des (R)STP-Protokolls ist insbesondere bei höherwertigen Switches möglich.

2.2.2.3 Das Lernen von Weiterleitungstabellen

Ein weiteres Problem, welches den Determinismus eines echtzeitfähigen Ethernets beeinflussen kann, liegt darin, dass die Switches zu Beginn keine Kenntnis davon besitzen, an welchen Port sie einen eintreffenden Frame mit einer bestimmten Zieladresse weiterzuleiten haben. Die MAC-Adresse der Quelle wird zusammen mit dem Port, an dem der Frame eingetroffen ist, in der *Weiterleitungstabelle* des Switches abgelegt [Wis07]. Somit kann ein Frame, der die MAC-Adresse der Quelle enthält, an diesen Port weitergeleitet werden. Um den zur noch unbekannten Zieladresse passenden Ausgangsport zu ermitteln, existieren zwei gängige Verfahren [Kau02].

Die eine Möglichkeit besteht darin, einen Frame wie bei einem Hub (s. Kapitel 2.2.3.1) als Broadcast an alle anderen Ports außer dem Eingangsport weiterzuleiten. Antwortet der Empfänger auf diesen Frame, so erfolgt dies über einen bestimmten Port dieses Switches, der dann die MAC-Adresse in seine Weiterleitungstabelle einträgt. Diese Möglichkeit wird als *Flooding* bezeichnet.

Die zweite Möglichkeit besteht darin, den Frame an einen beliebigen Ausgangsport weiterzusenden. Dies wird als *Hot-Potato* Verfahren beschrieben. Dazu geht man von einem zuverlässigen Protokoll wie TCP auf höherer Ebene aus. Kommt der Frame nicht beim Empfänger an, so wird davon ausgegangen, dass durch den Sliding-Window Algorithmus des TCP ein Timeout ausgelöst und der Frame nochmals versendet wird. Der Vorteil liegt darin, dass das Netzwerk mit dieser Methode weniger belastet wird und durch die Vermeidung des Floodings zuvor kein STP eingesetzt werden musste.

Ein weiteres Problem liegt darin, dass für die Weiterleitungstabellen nur ein endlicher Speicher zur Verfügung steht. Kauffels [Kau02] geht davon aus, dass ein Switch 1000 Adressen für jeden Port speichern sollte und dass im Backbone-Bereich über 10000 Adressen pro Port verwaltet werden müssten. Bei Low-Cost Switches kann jedoch an diesem Speicher gespart werden. Ist der Speicher voll, so werden alte Port-zu-MAC Zuordnungen überschrieben. Dies ist auch sinnvoll, da Geräte aus einem Netzwerk entfernt werden können. Andererseits müssen Geräte in regelmäßigen Abständen über alle relevanten Switches aktiv senden, damit ihre MAC-Adresse im Netzwerk bekannt bleiben. Die Geräte mit hohen Echtzeitanforderungen werden im laufenden Betrieb der automatisierten Anlage jedoch nicht entfernt. Senden solche Geräte über einen Zeitraum nicht aktiv, so dürfen diese Geräte dennoch nicht aus den Weiterleitungstabellen entfernt werden, da ansonsten kritische Frames unter Umständen verzögert zugestellt werden.

Werden VLANs verwendet, so werden die Adresstabellen der Switches um die VLAN-IDs erweitert und das VLAN-Tagging verwendet. Der erste Switch fügt dabei den Tag zu dem Frame hinzu, der von den folgenden Switches interpretiert wird. Eine genaue Beschreibung der Vorgehensweise hat Kauffels [Kau02] beschrieben. Die bekannten IDs werden in regelmäßigen Abständen, üblicherweise zwischen 1 oder 2 Minuten, von einem Switch an alle anderen Switches gesendet. Diese speichern dann die Zugehörigkeit der IDs zu ihrer Port-Nummer, an dem der sendende Switch angeschlossen ist, ebenfalls in der Weiterleitungstabelle. Solche Sendungen dürfen weder den Betrieb der Switches, beispielsweise durch Verzögerungen, beeinflussen, noch dürfen sie den Auslöser für eine verzögerte Weiterleitung von echtzeitkritischen Frames darstellen.

Beide beschriebenen Möglichkeiten des Erlernens der Weiterleitungstabellen sowie die Aktualisierung der VLAN-Informationen tragen dazu bei, dass der Netzwerkbetrieb in gewissen Zeiträumen stark beeinträchtigt wird. Ein deterministisches Verhalten des Netzes kann in diesen Zeiträumen nicht garantiert werden. Für die Automatisierungstechnik ist es essentiell, dass solche Verfahren ausschließlich in der Hochlaufphase der Anlage statt finden oder dass ein Weg gefunden wird, dass diese Verfahren den zu erzeugenden Determinismus des Netzwerkes nicht beeinträchtigen. Dazu dürfen die Einträge der Weiterleitungstabelle von echtzeitfähigen Geräten nicht nach einem Timeout verworfen werden. Sie sind also als statisch zu kennzeichnen, was bei konfigurierbaren Switches möglich ist. Ein Portwechsel eines echtzeitfähigen Gerätes im laufenden Betrieb der Anlage ist nicht zu befürchten.

2.2.2.4 Das Simple Network Management Protokoll (SNMP)

Ein weiteres Protokoll, welches die Echtzeitfähigkeit des Netzwerkes negativ beeinflussen kann, ist SNMP. Das von der Internet Engineering Task Force (IETF) entwickelte Protokoll [Har02] dient dazu, Geräte und Verteiler von einer zentralen Station aus zu überwachen und zu steuern. Das in der dritten Version vorliegende Protokoll unterstützt komplexe Fernkonfiguration, Überwachung und Fehlererkennung der unterstützten Geräte unter Verwendung der Management Information Base (MIB). Der Aufbau des Protokolls sowie dessen Funktionsweise wurde von Stevens [Ste04], ein Anwendungsfall von Dutine [Dut06] beschrieben.

Wie STP darf auch SNMP keine isochrone Übertragungen der automatisierten Anlage beeinflussen. Dies kann entweder durch die zusätzlichen protokollbedingten Übertragungen, oder auch durch die Beeinflussung der Switching-Hardware bei der Ausführung von SNMP-Kommandos geschehen. Zusammenfassend ist zu sagen, dass alle im Ethernet ausgeführten Protokolle gegenüber isochrone Übertragungen zurückgestellt werden müssen, um eine *harte Echtzeitfähigkeit* des Netzwerkes zu garantieren. Die Ausführung dieser Protokolle sind bei konfigurierbaren Switches in verschiedenen Ausprägungen kontrollierbar.

2.2.3 Ethernet-Verteiler

In diesem Kapitel werden Ethernet-Verteiler bis OSI-Schicht 2 betrachtet. Diese sind im Wesentlichen in Repeater/Hubs auf OSI-Schicht 1 und in Switches auf OSI-Schicht 2 zu unterteilen. Da der Fokus dieser Arbeit auf der Switching-Technologie liegt, wird deren Arbeitsweise im folgenden Kapitel genauer betrachtet. Im Zusammenhang mit der Automatisierungstechnik ist der Begriff der Verzögerungszeit von großer Bedeutung, da diese minimal zu halten ist.

Die *Verzögerungszeit in einem Verteiler* wird definiert als die Zeit, die von dem Eintreffen des ersten Bits eines Frames am Eingangsport des Verteilers bis zum messbaren Erscheinen dieses Bits am Ausgangsport vergeht. Die *Verzögerungszeit einer Übertragung* wird definiert als die Zeit, die das erste Bit eines Frames vom Ethernetport des sendenden Gerätes bis zu seiner Ankunft am Ethernet-Port des empfangenen Gerätes benötigt.

2.2.3.1 Repeater und Hubs

Bei einem *Repeater* handelt es sich um einen Signalregenerator, der in der Bitübertragungsschicht ein Signal empfängt, dieses Signal neu aufbereitet und wieder außendet. Ein Repeater verbindet also zwei Netzwerksegmente und besitzt daher zwei Anschlüsse, die Ports genannt werden. Ein *Hub*, auch Multiportrepeater genannt, besitzt mehr als zwei Ports, erfüllt jedoch genau die Aufgabe eines Repeaters.

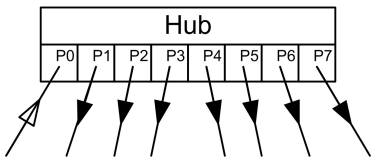


Abbildung 2.31: Datenfluss in einem Hub

Die Abbildung skizziert, wie ein Hub einen Frame auf Signalebene an einem ankommenden Port P0 empfängt, das Signal aufbereitet und stets an alle anderen Ports wieder absendet. Ebenso werden Kollisionen an alle Ausgangsports weiter geleitet. Ein Hub arbeitet ausschließlich im Halbduplexmodus, er kann also zu einem Zeitpunkt entweder senden oder empfangen. Der Zugriff auf die physikalische Ebene wird mittels CSMA/CD geregelt.

2.2.3.2 Switches

Im Gegensatz zu einem Hub interpretiert ein *Switch* die Zieladresse eines Frames und leitet ihn nur an die Ports weiter, an die auch Empfänger dieses Frames angeschlossen sind. Abbildung 2.32 zeigt dies für eine Unicast- und eine Multicastsendung. Ein Switch lernt den Zusammenhang zwischen den Zieladressen und den Ports durch den Aufbau einer Weiterleitungstabelle.

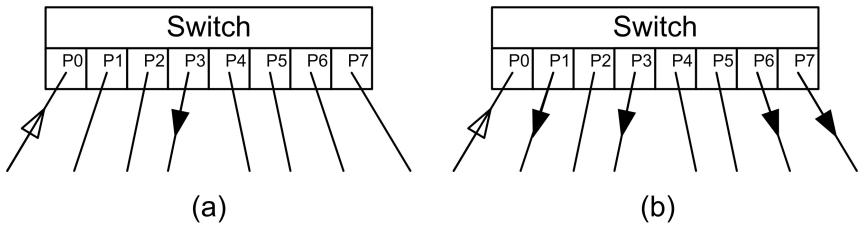


Abbildung 2.32: Datenfluss einer Unicast- (a) und Multicastsendung (b) in einem Switch

Ein Switch ist aufgrund der getrennten Twisted-Pair Leitungen für das Senden und Empfangen in der Lage, auf jedem Port im Vollduplexmodus zu arbeiten [Kau02] und damit Kollisionen vollständig zu vermeiden. Somit wird der nicht-deterministische CSMA/CD-Algorithmus nicht angewendet. Ist ein Port eines Switches mit einem Halbduplex-Endgerät oder mit einem Hub verbunden, so wechselt dieser Port durch Autonegotiation [Hel98] in die Halbduplexbetriebsart. Bei einer auftretenden Kollision wird diese nicht an andere Ports des Switches weiter geleitet. Ein Switch trennt also Kollisionsdomänen.

Werden Frames von vielen Eingangsports an wenige Ausgangsport eines Switches gesendet, so werden diese an den Ausgangsports zwischengespeichert, s. Abbildung 2.33. Dies ist notwendig, da die Ausgangsleitung einen Engpaß darstellt. Senden beispielsweise die Geräte 1 bis 3 jeweils mit $100Mbit/s$, so können über die Leitung von Port P3 des Switches lediglich Daten mit $100Mbit/s$ zu Gerät 4 weitergeleitet werden. Die Frames stauen sich also in dem Switch auf. Diese Zwischenspeicherung erhöht jedoch die Verzögerungszeit von Frames in einem Switch in Abhängigkeit seiner aktuellen Last. Ist der Zwischenspeicher voll, so werden weitere ankommende Frames verworfen in der Hoffnung, dass diese durch ein übergeordnetes Protokoll wie TCP nochmals gesendet werden und die Senderate aufgrund des verworfenen Frames durch Anwendung des Additive Increase / Multiplicative Decrease Verfahrens [Ste04] von TCP gesenkt wird. Aufgrund dessen ist eine deterministische Datenübertragung trotz der Vermeidung des CSMA/CD-Algorithmus nicht mehr gewährleistet: Eine zeitliche Obergrenze, wann ein zu sendender Frame beim Empfänger eintrifft, kann also nicht genannt werden.

Auch wenn der Zwischenspeicher nicht überläuft, so kann sich dadurch die Übertragung der Frames im Worst-Case erheblich verzögern, wie Schwager [Sch04b] skizziert. *Store-*

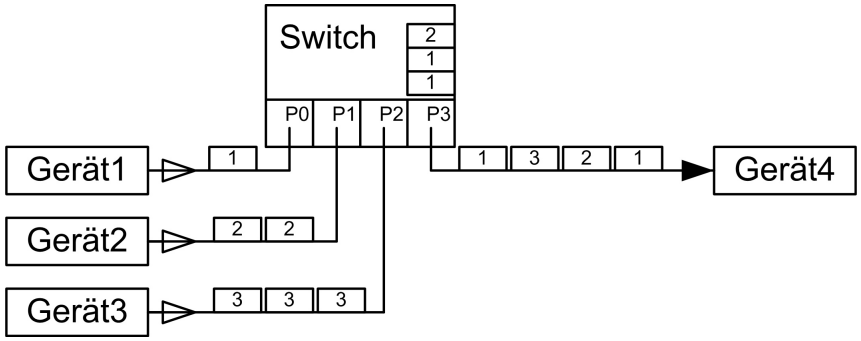


Abbildung 2.33: Zwischenspeichern von Frames in einem Switch [Sch04b]

and-forward Switches speichern jeden Ethernet-Frame vollständig, prüfen die FCS und leiten nur fehlerfreie Frames an die angegebene MAC-Adresse des Ziels weiter, während fehlerhafte Frames verworfen werden. Da Ethernet-Frames eine Größe zwischen $72Byte$ und $1530Byte$ incl. VLAN-Tag besitzen, ist die Zeit für den Empfang eines vollständigen Frames unterschiedlich groß. Die Verzögerungszeit ist damit abhängig von der Framelänge.

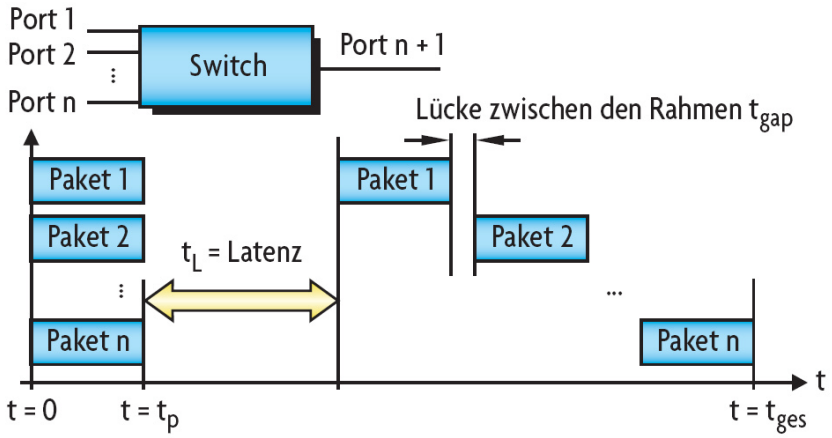


Abbildung 2.34: Übertragungverzögerung eines store-and-forward Switches [Sch04b]

Zum Zeitpunkt $t = 0$ sollen n Nachrichten an n verschiedenen Ports des Switches an einen einzelnen Ausgangsport geleitet werden. Die Übertragungsdauer aller Nachrichten ergibt sich dann mit $t_{ges} = t_f + t_L + n \cdot t_f + (n-1) \cdot t_{gap}$. Dabei ist t_f die benötigte Zeit zur Übertragung eines Frames, t_L die Verzögerungszeit innerhalb des Switches und t_{gap} die Interframe Gap (IFG). Diese IFG muss notwendigerweise zwischen zwei aufeinander folgenden Frames liegen, damit diese von den Netzwerkkarten und Switches als getrennte Frames erkannt werden. Die Größe der Zeiten ist abhängig von der Übertragungsrate, deren Standards in Kapitel 2.2.4 beschrieben werden.

Ein *(virtual) cut-through Switch* führt hingegen keine Prüfung der FCS durch. Ein solcher Switch speichert den eintreffenden Frame so lange, bis er die MAC-Adresse des Ziels auslesen kann. Damit müssen $14Byte$ incl. Präambel zwischengespeichert werden, vgl. Abbildung 2.28. Unterstützt der Switch VLAN-Netzwerke oder lediglich die priorisierte Weiterleitung mit VLAN-Tags [DT97], so müssen auch diese VLAN-Tags eingelesen und interpretiert werden. In diesem Fall sind $24Byte$ incl. Präambel zwischenzuspeichern, vgl.

Abbildung 2.29. Sobald der Ausgangsport bekannt ist, werden die ersten Bytes aus dem Zwischenspeicher direkt an diesen Ausgangsport weitergeleitet, sofern auf diesem Port nicht gerade ein anderer Frame versendet wird. Ist dies der Fall, so wird der betreffende Frame zwischengespeichert. Die benötigte Zeit zur Weiterleitung ist dadurch unabhängig von der Framelänge und nahezu konstant, sofern keine Zwischenspeicherung des Frames aufgrund eines belegten Ausgangsports durchgeführt wird.

Ein *fragment-free Switch* arbeitet ähnlich wie ein cut-through Switch, prüft aber zusätzlich, ob ein Frame die genormte minimale Länge von $72Byte$ incl. Präambel besitzt. Eine Prüfung der FCS wird nicht vorgenommen. Da diese Anzahl an Bytes vom Switch vor der Weiterleitung ausgelesen werden muss, arbeitet ein Fragment-Free Switch etwas langsamer als ein cut-through Switch, jedoch auch mit einer frame-unabhängigen Verzögerungszeit. Der Grund für die Existenz solcher Switches liegt darin, dass Fragmente unter $72Byte$ meist Teile einer vorangegangenen Kollision durch Netzwerksegmente im Halbduplexmodus sind, die keinen sinnvollen Frame ergeben [Cho00]. Fragment-free Switches sind nicht weit verbreitet und meist im Backend von Firmennetzen vertreten.

Adaptive Switches werden auch als error-free cut-through Switches bezeichnet, sind ebenso selten und eher im professionellen Bereich zu finden. Sie vereinigen mehrere der bislang genannten Methoden. Ein adaptiver Switch arbeitet zunächst im cut-through Modus mit dessen Verzögerung. Zusätzlich wird jedoch eine Prüfung der FCS von einer Kopie des Frames im Speicher des Switches vorgenommen, die jedoch auf diesen Frame keinen Einfluss mehr hat. Statt dessen wird ein Zähler inkrementiert, der die Fehlerzahl pro Zeiteinheit repräsentiert. Überschreitet dieser Zähler einen Grenzwert, so wechselt der Switch in den store-and-forward Modus. Ein zweiter Zähler kann die Anzahl der Fragmente mit einer geringeren Länge als $72Byte$ messen und einen fragment-free Modus ermöglichen. Mit diesen beiden Methoden wird die Fehleranfälligkeit des Gesamtnetzes reduziert. Für den Einsatz in der Automatisierungstechnik bedeutet dies eine nicht vorhersehbare Änderung der Verzögerungszeit, da dieser Switch zwischen dem cut-through Modus und dem store-and-forward Modus wechselt und damit eine variable Verzögerung der Ethernet-Frames erzeugt.

Von Leitner [Lei03] hat beschrieben, dass alte Switches Multicast-MAC-Adressen ignorieren und die Frames verwerfen oder den Multicast-Adressraum komplett freischalten, also die Multicast-Nachrichten als Broadcast versenden. Heutzutage eingesetzte Switches, die nach 1995 hergestellt wurden, unterstützen eine andere Form des Multicasts. Sie sind durch das Abhören von Paketen des Internet Group Message Protokolls (IGMP) [CDKFA02] in der Lage, IGMP-Nachrichten zu interpretieren. Dieser Vorgang wird als IGMP-Snooping bezeichnet [Cro04]. Über diese Nachrichten treten Geräte einer Multicast-Gruppe bei. Ist der Beitritt erfolgreich, so kann der Switch dies durch die Analyse der Nachrichten erkennen und nur die Ports freischalten, an denen sich Empfänger der Daten befinden. Dies geschieht durch eine Anpassung der Weiterleitungstabelle.

2.2.4 Kenngrößen der Übertragungsstandards

Nachdem in den vorherigen Kapiteln die Adressierung und die Arten von Verteilern vorgestellt wurden, liegt der Fokus dieses Kapitels auf den Kenngrößen der Übertragungsstandards. Dabei wird zunächst das historische $10Mbit/s$-Ethernet skizziert. Im Anschluss

daran werden die Kenngrößen des gängigen $100Mbit/s$-Ethernet vorgestellt sowie die Verzögerungszeiten typischer $100Mbit/s$-Verteiler. Da es sich bei dem $100Mbit/s$-Ethernet um den gängigen Standard handelt und etablierte echtzeitfähige Ethernet Lösungen auf diesem Standard basieren (s. Kapitel 3.1.1), wird im folgenden Verlauf dieser Arbeit von einem $100Mbit/s$-Netzwerk ausgegangen. Die formale Modellierung wird jedoch von konkreten Kenngrößen abstrahieren und daher allgemeingültig sein. Abschließend werden in diesem Kapitel die Daten des Gigabit-Ethernets vorgestellt, welches in naher Zukunft mit hoher Wahrscheinlichkeit den Markt dominieren wird [Hei98].

2.2.4.1 Der 10 Mbit/s-Standard

Bei dem Ethernet mit einer Übertragungsrate von $10Mbit/s$ nach IEEE 802.3 [IEEE05] handelt es sich ursprünglich um eine *Linientopologie* nach 10Base5 oder 10Base2 [Hel98], die ausschließlich im Halbduplexmodus betrieben werden kann. Die Linientopologie ist auch in der Automatisierungstechnik weit verbreitet, um den Verkabelungsaufwand innerhalb der Anlage zu reduzieren. Als Alternative dazu existiert die 10BaseT-Verkabelung [Hel98] mit *Baumtopologie*, welche bereits beim $10Mbit/s$-Ethernet eine Vollduplex-Übertragung zulässt. Als Verteiler haben sich Hubs etabliert; $10Mbit/s$-Switches sind nur selten im Einsatz. Der $10Mbit/s$-Standard des Ethernet gilt als veraltet und wird hier nur noch erwähnt, da bestimmte Einschränkungen des heutigen Ethernets auf Kompatibilitätsgründe zum $10Mbit/s$-Ethernet zurückzuführen sind.

Die Kenngrößen des $10Mbit/s$-Ethernets sind darauf ausgerichtet, dass ein Sender bei einer Halbduplexübertragung auf jeden Fall eine auftretende Kollision erkennen kann. Neben einer Obergrenze für die maximale Distanz zwischen einem Sender und einem Empfänger spielt dabei die minimale Länge eines Frames eine wichtige Rolle. Denn ein minimal großer Frame muss in der Lage sein, ein Netz mit maximaler Ausdehnung zu fluten [Kau02]. Aus diesem Grunde wurden folgende Parameter spezifiziert:

- min. Framegröße (o. Präambel): $64Byte$
- max. Framegröße (o. Präambel): $1518Byte$
- Slotzeit: $51.2\mu s$
- Pause zwischen 2 Frames (IFG): $9.6\mu s$
- Zeit zur Übertragung eines min. Frames: $67.2\mu s$

Die IFG ist sowohl im Halbduplex-, als auch im Vollduplexmodus notwendig, damit ein Empfänger die nachfolgende Präambel in jedem Falle erkennen kann [Sch07]. Treffen zwei Frames mit geringerem Abstand bei einem Verteiler ein, so kann der zweite Frame verworfen werden.

Die Slotzeit entspricht der maximalen Laufzeit eines Frames bis zum entferntesten Gerät der Kollisionsdomäne und der Rücklaufzeit eines möglichen Kollisionssignals mit $\frac{64Byte}{10Mbit/s} = 51.2\mu s$.

Im Vollduplexbetrieb verdoppelt sich die zur Verfügung stehende Bandbreite durch die Möglichkeit des gleichzeitigen Sendens und Empfangens, wodurch Kollisionen bei ausschließlicher Verwendung von Switches unmöglich werden. Das Einhalten der minimalen Framegröße und der Slotzeit ist dadurch hinfällig, da die Ausführung des CSMA/CD-Verfahrens obsolet wird. Jedoch verwerfen einige Verteiler zu kurze Frames als nicht standardkonform.

Addiert man zu einem kleinst möglichen Ethernet-Frame mit $64Byte$ die Präambel mit $8Byte$ sowie die Umrechnung der IFG in Byte ($10Mbit/s \cdot 9.6\mu s = 1.25MByte/s \cdot 9.6\mu s = 12Byte$), so werden insgesamt $84Byte$ übertragen, wozu $67.2\mu s$ benötigt werden.

2.2.4.2 Der 100 Mbit/s-Standard

Im aktuellen $100Mbit/s$-Standard des Ethernet nach IEEE 802.3u [SL07] wird auf eine Linientopologie vollständig verzichtet. Obwohl mit dem 100BaseFX auch Multimode-Glasfaser im 100MBit-Standard vorgesehen ist, ist eine Verkabelung nach 100BaseT auf Kupfer-Basis wesentlich weiter verbreitet. Da automatisierte Anlagen meist elektromagnetische Störungen verursachen, ist eine Glasfaserverkabelung durchaus empfehlenswert. Eine Analyse der Störanfälligkeit und Methoden zu deren Beseitigung ist jedoch nicht Gegenstand dieser Arbeit.

Die Framegrößen wurden aus Kompatibilitätsgründen zum 10Mbit/s Ethernet nicht verändert. Die Slotzeit beträgt aufgrund der 10-fachen Geschwindigkeit nur 1/10 des $10Mbit/s$-Ethernet Wertes. Die IFG wird beibehalten und ebenfalls an die neue Geschwindigkeit angepasst; sie verringert sich ebenso auf 1/10. Daraus resultieren die folgenden Kenngrößen für $100Mbit/s$-Ethernet:

- min. Framegröße (o. Präambel): $64Byte$
- max. Framegröße (o. Präambel): $1518Byte$
- Slotzeit: $5.12\mu s$
- Pause zwischen 2 Frames (IFG): $0.96\mu s$
- Zeit zur Übertragung eines min. Frames: $6.72\mu s$

Als Verteiler in $100Mbit/s$-Netzwerken haben sich Switches durchgesetzt. $100Mbit/s$-Hubs, die ausschließlich halbduplex arbeiten, sind zwar noch vereinzelt im Einsatz, spielen jedoch bei Neuanschaffungen keine Rolle mehr. Die *Bitzeit* beträgt im 100Mbit/s-Ethernet $t_{BK} = 0.01\mu s$.

Da Hubs den Frame nicht interpretieren, sondern lediglich eine Signalaufbereitung durchführen, ist deren Verzögerungszeit unabhängig vom übertragenen Frame. Der Standard erlaubt eine maximale Verzögerungszeit eines $100Mbit/s$-Hubs von 92 Bitzeiten [IDG03], also von $0.92\mu s$. Eine Messung [SZ03] hat eine durchschnittliche Verzögerung $< 0.4\mu s$ ergeben. Da keine einfacherern Verteiler als Hubs existieren, kann diese Verzögerungszeit als minimal angenommen werden.

Damit ein cut-through Switch den Zielport interpretieren kann, muss zunächst die Präambel und die Zieladresse ausgelesen werden. Für das Auslesen dieser $14Byte$ werden $1.12\mu s$ benötigt. Interpretiert ein Switch VLAN-Tags, so müssen weitere $10Byte$ ausgelesen werden, da diese Tags hinter der MAC-Adresse der Quelle übertragen werden, vgl. Abbildung 2.29. Für das Auslesen der ersten 24 Byte des Frames werden $1.92\mu s$ verwendet. Da die IFG bis zur Ankunft des nächsten Frames abgewartet werden muss, können diese $0.96\mu s$ ebenfalls zum Frame addiert werden, so dass sich als minimale Verzögerungszeit $2.08\mu s$ bzw. $2.88\mu s$ ergeben. Der Switch benötigt jedoch zusätzlich Zeit, um den Zielport in der Weiterleitungstabelle zu ermitteln, den Datenstrom zu diesem Zielport weiterzuleiten und ggf. die VLAN-Einstellungen zu verarbeiten. In der Literatur werden typische Verzögerungszeiten von cut-through Switches mit $10\mu s$ [Fel05] bis zu $40\mu s$ [BH06] pro Switch angegeben, Scheitin und Zuber [SZ03] ermittelten eine Verzögerung von $8.75\mu s$ für einen $102Byte$ großen Frame.

Da store-and-forward Switches gesamte Frames einlesen, zwischenspeichern und deren FCS prüfen, sind deren Verzögerungszeiten abhängig von der Größe der Frames. Ein minimal großer Ethernet-Frame incl. Präambel benötigt allein für seine Speicherung $5.76\mu s$, ein maximal großer Frame sogar $122.08\mu s$. Des Weiteren ist die IFG, die notwendige Zeit für die Prüfung der FCS sowie für das Ermitteln des Zielports zu addieren. Als Gegenüberstellung zum cut-through Switch ergibt die von Scheitlin und Zuber [SZ03] durchgeführte Messung eine Verzögerung zwischen $10.66\mu s$ und $15.40\mu s$ bei einem Frame mit einer Größe von $102Byte$ unter Verwendung von drei unterschiedlichen store-and-forward Switches. Ein Frame mit einer Größe von $1046Byte$ verzögerte sich um durchschnittlich $90\mu s$ ohne zusätzliche Netzlast. Aufgrund dieser Eigenschaften werden store-and-forward Switches als ungeeignet für die Übertragung von echtzeitkritischen Daten der Automatisierungstechnik eingestuft, da sie den Anforderungen der Automatisierungstechnik widersprechen.

2.2.4.3 Der Gigabit-Standard

Auch im Gigabit-Standard nach IEEE 802.3z ist ein Halbduplex- und ein Vollduplexmodus vorgesehen [SL07]. Der erste Modus wird als Shared Gigabit-Ethernet bezeichnet und wird höchst wahrscheinlich keine Verbreitung finden, da er lediglich einen einzelnen Repeater im Subnetz zulässt. Zur Abwärtskompatibilität wird wiederum der CSMA/CD-Algorithmus verwendet, der die minimale Framegröße und die Slotzeit vorgibt. Beim Gigabit-Ethernet hat man sich darauf geeinigt, die minimale Framegröße auf $512Byte$ zzgl. Präambel zu setzen; die Obergrenze von $1518Byte$ wird beibehalten. Werden die $512Byte$ unterschritten, so füllt die Bitübertragungsschicht den Frame durch eine Carrier Extention auf [Sta99]. Dieses Verfahren ähnelt dem Padding, welches auch innerhalb eines Ethernet-Frames eingesetzt wird. Um die Effizienz zu steigern, erlaubt man im Halbduplexmodus auch die direkte Hintereinanderreihung von Frames eines Senders ohne IFG. Mit diesem Burstmodus können bis zu $8172Byte$ incl. Präambeln der einzelnen Frames hintereinander gefügt werden. Dies gilt auch dann, wenn ein Sender an verschiedene Empfänger sendet. Der Nachteil besteht darin, dass die restlichen Geräte länger auf ihre Sendeberechtigung warten müssen, falls ein hohes Aufkommen an Bursttraffic auf dem gemeinsamen Medium existiert [Kau02].

Im Vollduplexmodus wird, wie bereits beim $100Mbit/s$-Standard üblich, mit Switches gearbeitet. Aufgrund der Kollisionsfreiheit wird auf das CSMA/CD-Verfahren verzichtet. In dieser Betriebsart verfügt das Gigabit-Ethernet auch über eine einfache Flusskontrolle, bei der ein Empfänger einen speziell definierten Pause-Frame an den Sender zurücksenden kann. Dieser Frame enthält eine Pausenzeit, für die der Sender seine Sendungen einstellt. Im Folgenden sind wesentliche Kenngrößen des Gigabit-Ethernets zusammen gefasst:

- min. Framegröße (o. Präambel): $512Byte$
- max. Framegröße (HD/VD): $8172/1518Byte$
- Slotzeit: $5.12\mu s$
- Pause zwischen 2 Frames (IFG): $0.096\mu s$
- Zeit zur Übertragung eines min. Frames: $4.26\mu s$

Das Gigabit-Ethernet ist bereits für die Kommunikation von Switches untereinander verbreitet, höherwertige Switches besitzen meist einen Gigabit Uplinkport. Zusätzlich werden Mutterplatinen herkömmlicher PCs heutzutage meist mit einer Gigabit Netzwerkkarte versehen. Im Home-Office Bereich sind jedoch noch $100Mbit/s$-Netzwerke vorherrschend.

Auffallend ist, dass die Zeit zur Übertragung eines minimal großen Frames gegenüber dem $100Mbit/s$-Ethernet keinen großen Vorteil bringt, was auf die Erhöhung der minimalen Framegröße zurückzuführen ist. Da der Nutzdatenanteil bei automatisierten Anlagen lediglich einige Byte beträgt, kann Gigabit-Ethernet die Vorteile der erhöhten Bandbreite in der Automatisierungstechnik nicht ausschöpfen.

2.2.5 Synchronisierung im Ethernet

Eine weitere Anforderung der Automatisierungstechnik besteht in einer genauen Synchronisierung der Sendungen, die als Isochronität bezeichnet wird. Die Geräte sind als unabhängige, verteilte Systeme anzusehen, welche eine hochpräzise, gemeinsame Zeitbasis besitzen müssen. Im Umfeld des Ethernets und insbesondere auch im Umfeld etablierter Ansätze zur Erreichung von Echtzeitfähigkeit im Ethernet sind bereits einige Vorgehensweisen zur Synchronisation verbreitet.

2.2.5.1 (Simple) Network Time Protocol

Ein gängiges Verfahren zur *Zeitsynchronisation* in LANs und auch über das Internet ist das nach RFC 1119 [Mil89] und RFC 1129 [Mil89b] genormte Network Time Protocol (NTP). Die erste Version wurde bereits 1985 als RFC 958 [Mil85] veröffentlicht. NTP steht als Dienst der Anwendungsschicht zur Verfügung und verwendet den UDP/IP-Protokollstapel. Bei dem Simple Network Time Protocol (SNTP) handelt es sich um eine vereinfachte Version des NTP, welches in RFC 4330 beschrieben ist und eine ähnliche Genauigkeit wie NTP aufweist [Mil06].

NTP nimmt an, dass ein Gerät eine exakte Zeit besitzt, die es beispielsweise durch den Empfang eines Global Positioning System (GPS) Signals oder DCF77-Signals (s. Kapitel 2.2.5.3) erhält. Dieses Gerät wird zum primären Zeitserver und erhält die Stratum-ID 1. Anhand dieser ID lassen sich Zeitserver im Netzwerk in einer Baumstruktur kaskadieren. Alle Server, die sich mit Stratum 1 synchronisieren, erhalten die Stratum-ID 2 usw. Je größer die ID, desto höher ist die Abweichung zum primären Zeitserver.

Ziel von NTP ist es nun, die Uhrzeiten aller Geräte über ein nicht-deterministisches Netzwerk zu synchronisieren. Dieses Verfahren wird in Abbildung 2.35 vereinfacht beschrieben:

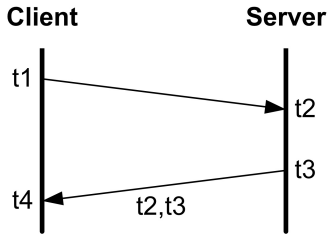


Abbildung 2.35: Zeitsynchronisation mit NTP

Das Gerät sendet als Client zunächst eine Nachricht zu einem Zeitpunkt t_1 an den Zeitserver, der diese zum Zeitpunkt t_2 empfängt. Er antwortet zur Zeit t_3 mit einem Frame, der die Zeitstempel t_2 und t_3 enthält. Dieser Frame trifft zur Zeit t_4 beim Gerät

ein. Die Laufzeit d der Frames lässt sich mit $d = (t_4 - t_1) - (t_3 - t_2)$ ebenso ermitteln wie die Zeitdifferenz $T = ((t_2 + t_3) - (t_1 + t_4))/2$. NTP geht von einer symmetrischen Verzögerung aus, d.h. dass der gesendete Frame eine identische Verzögerungszeit besitzt wie das empfangene. Dies ist aufgrund von Zwischenspeicherungen in Switches oder alternativen Routen im Internet nicht immer der Fall. Um Fehler durch unterschiedliche Laufzeiten zu minimieren, wird nicht einfach die letzte Zeitdifferenz zur Zeitkorrektur verwendet, sondern es werden die letzten acht Messungen betrachtet. Dabei wird ein gewichteter Mittelwert der letzten acht Zeitdifferenzen eines Zeitservers mit der aktuellen Differenz gebildet, wobei die Zeitdifferenzen mit geringerer Laufzeit höher gewichtet werden als Messungen mit hoher Laufzeit. Dieser Mittelwert wird als Dispersion bezeichnet. Zusätzlich kann die Genauigkeit durch die Verwendung von mehreren Zeitservern erhöht werden, da sich asymmetrische Laufzeitverzögerungen bei unterschiedlichen Servern ausgleichen. Das Gerät synchronisiert abschließend seine Uhr mit den Daten eines Servers, der sich als die beste Quelle erwiesen hat.

Unter günstigen Bedingungen, also bei annähernd gleicher Übertragungszeit der beiden Nachrichten auf dem Hin- und Rückweg, ist eine softwarebasierte Zeitsynchronisation auf $1ms$ möglich, im Internet wird mit einer Synchronisationsgenauigkeit im Bereich von $10ms$ gerechnet.

In LANs sind unter idealen Bedingungen sogar Genauigkeiten von ca. $200\mu s$ möglich. Dazu werden die Zeitstempel auf der Sicherungsschicht, ggf. unter dem Einsatz von Spezialhardware, ermittelt und den Frames hinzugefügt. Man spricht hier von einem *low-level timestamping* [Her05]. Durch die Bearbeitung auf unterer Protokollebene werden durch das Betriebssystem des Gerätes bedingte Schwankungen der Verarbeitungszeit weitgehend ausgeschlossen.

Für gängige Anwendungen ist dies absolut ausreichend, jedoch nicht für härtere Echtzeitklassen der Automatisierungsrechnik. Dort liegt die notwendige Genauigkeit im Bereich von Nanosekunden. Anhand von NTP kann jedoch das Prinzip der Zeitsynchronisation und dessen Schwierigkeiten leicht erläutert werden, weitere Informationen zu NTP sind Kostecke [Kos05] und Mills [Mil03] zu entnehmen.

2.2.5.2 Synchronisation nach IEEE 1588

Ein Verfahren zur Synchronisation, welches unter Verwendung von Hardwareunterstützung eine Zeitsynchronisation $< 1\mu s$ erreicht, ist nach *IEEE 1588* genormt [IEEE02]. Dieses Verfahren ist bei einer Vielzahl von Lösungen im Umfeld von echtzeitfähigem Ethernet der Automatisierungstechnik im Einsatz [Sch03c]. Die Verzögerungszeiten, die durch das Netzwerk und die Interpretation des Protokolls entstehen, können dabei mit einer Genauigkeit von $1\mu s$ ermittelt werden [And06]. Abbildung 2.36 zeigt die Vorgehensweise des nach IEEE 1588 genormten *Precision Time Protocol* (PTP). Die Uhr des Slaves besitzt zu Beginn des Protokollablaufs eine zeitliche Abweichung gegenüber der Uhr des Masters, die als Offset O gekennzeichnet ist. Diese Abweichung ist durch das Protokoll auszugleichen.

Dabei wird ein Sync-Frame zyklisch vom Master an alle Slaves gesendet und beinhaltet unter anderem den Zeitstempel t_{estim}, der den voraussichtlichen Sendezeitpunkt des Sync-Frames beinhaltet. Der exakte Sendezeitpunkt des Sync-Telegramms wird gemessen und in einem Follow_Up-Frame als t_0 vom Master nachgereicht. Die Slaves messen den exakten

Empfangszeitpunkt des Sync-Frames und können bereits mit der empfangenen Zeit t_0 die Abweichung zur Master-Uhr berechnen.

Die ermittelte Abweichung ist jedoch mit dem Fehler der Übertragungsdauer auf dem Netzwerk behaftet, die als Delay D bezeichnet wird. Zur Ermittlung dieser Verzögerung wird ein Delay_Req-Frame vom Slave an den Master unter Messung der exakten Sendezeit t_2 gesendet. Der Master ermittelt die exakte Empfangszeit dieses Fames und sendet diese Zeit t_3 an den Slave mit einem Delay_Resp-Frame zurück.

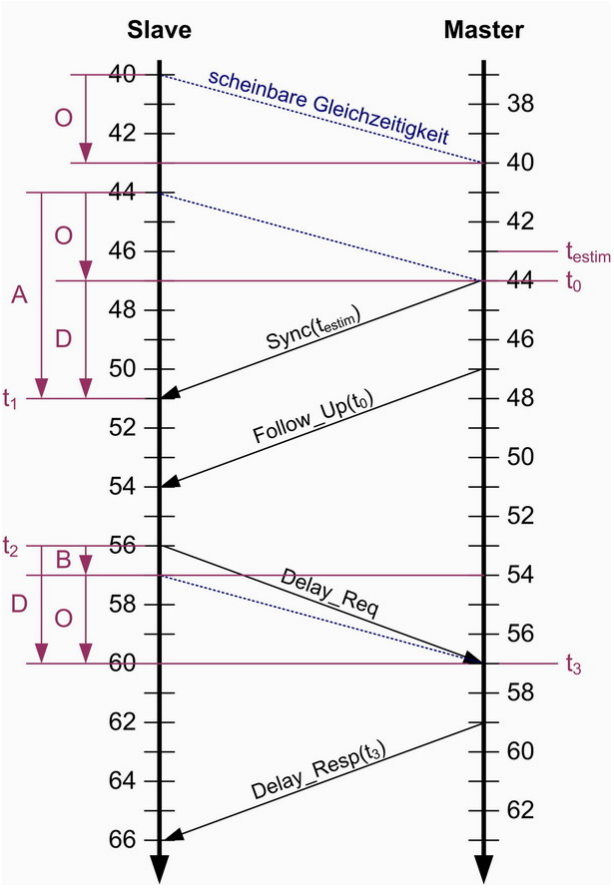


Abbildung 2.36: Zeitsynchronisation nach IEEE 1588 [Wei06]

Die so ermittelte Verzögerung fließt in die folgende Berechnung der Uhrenabweichung ein. Die Differenz der Zeiten t_1 und t_0 beinhalten das Delay und den Offset, so dass $A = D + O = t_1 - t_0$ gilt. Die Differenz der Zeiten t_3 und t_2 bilden das Delay abzüglich des Offsets, also $B = D - O = t_3 - t_2$. Da die beiden Zeitdifferenzen bekannt und in den Hilfsgrößen A und B abgelegt sind, können durch zwei Gleichungen mit zwei Unbekannten das Delay und der Offset berechnet werden mit

$$A + B = 2D \Leftrightarrow D = \frac{A+B}{2}$$

und

$$O = A - D = \frac{2A}{2} - \frac{A+B}{2} \Leftrightarrow O = \frac{A-B}{2}.$$

Seit einiger Zeit existieren Hardware-Bausteine wie der DP83640 von National Semiconductor [Zin07], welche eine Synchronisierung möglichst nah an der Übertragungsleitung durchführen. Diese hardwarenahen Zeitstempel werden auf der OSI-Schicht 2 implementiert und sind damit unabhängig von Verzögerungen und Jitter, die auf höheren Protokollebenen entstehen. Auf diese Weise kann eine Synchronisation im ns-Bereich erzielt werden. Weitere Informationen zum Protokoll können den Arbeiten von Eidson [Eid06], Sikora [Sik03], Gramann und Mohl [GM03] entnommen werden.

2.2.5.3 DCF77 und GPS

Um eine Synchronisation zu erreichen, kann auch jedes Gerät direkt mit einem Empfänger einer hochpräzisen Zeit ausgestattet werden. Die erste Möglichkeit besteht hier in der Verwendung eines *DCF77*-Empfängers [Lin06]. Der Langwellen-Sender mit $77,5kHz$ und einer Leistung von $50kW$ befindet sich in Mainflingen bei Frankfurt am Main und besitzt eine Reichweite von ca. $1900km$. Durch die Langwellenübertragung mit Amplitudenmodulation ist das Signal jedoch störanfällig. Bei einem Gewitter am Sendestandort wird die Abstrahlung für die Dauer des Gewitters eingestellt [Hop07]. Zusätzlich können Störungen am Empfangsort, hauptsächlich verursacht durch Antriebe, Monitore oder schaltende Schütze, die Synchronisation beeinträchtigen. Eine Möglichkeit zur Unterdrückung von Störungen besteht durch den Einsatz von schmalbandigen Empfängern, mit denen jedoch die Genauigkeit des Uhrenabgleiches sinkt. Selbst im optimalen Fall ist nur eine für harte Echtzeitklassen unzureichende Genauigkeit im Bereich von $5ms$ zu erreichen.

Eine Genauigkeit im μs-Bereich lässt sich über *GPS-Empfänger* erreichen, so dass ähnliche Eigenschaften wie bei einer Synchronisierung nach IEEE 1588 bestehen. Dieses Signal wird satellitenbasiert abgestraht, ist aufgrund der hohen Sendefrequenz von 1,57 GHz unempfindlich gegen Störungen und kann weltweit empfangen werden. Zum Empfang ist jedoch eine Außenantenne notwendig, während sich die Antennen von DCF77-Empfängern auch innerhalb einer Fertigungshalle befinden können. Die Satelliten senden ihre Bahnpositionen kontinuierlich zusammen mit der GPS-Weltzeit. Ein Empfänger wertet die Daten aus und bestimmt zunächst die Position der Empfangsantenne. Im Anschluss daran können die Laufzeiten der Sendeinformationen der einzelnen Satelliten ermittelt werden [Reg05]. Der Standard wurde in einem Dokument des Department of Defense der US-Regierung [DoD01] beschrieben. Die genaue Funktionsweise der GPS-Technologie hat Xu [Xu03] beschrieben.

Sowohl beim Einsatz einer DCF77-, als auch einer GPS-Synchronisation ist zu bedenken, dass der Anlagenbetreiber von einer externen Datenquelle abhängig ist, die keine Garantie für eine kontinuierliche und adäquate Ausstrahlung der Signale gibt. Bei einer fehlerhaften oder ausbleibenden Synchronisation ist damit die laufende Produktion gefährdet.

2.2.6 Zusammenfassung

Zusammenfassend lässt sich sagen, dass es sich bei Ethernet um ein altes, jedoch weltweit in hohem Maße verbreitetes Protokoll handelt. Nach heutigem Stand der Technik ist $100Mbit/s$-Ethernet mit einer Baumtopologie unter Einsatz von Switches im Vollduplexmodus vorherrschend. Es wurde jedoch bereits angedeutet, dass einige Protokolle, die standardmäßig im Ethernet angewendet werden, wie Prioritätenvergabe und VLANs, nur bedingt für den Echtzeitbetrieb im Umfeld der Automatisierungstechnik geeignet sind. Protokolle wie STP oder CSMA/CD sind sogar kontraproduktiv. Auf der anderen Seite existiert das vielversprechende Synchronisationsprotokoll nach IEEE 1588.

2.3 Scheduling und Graphenfärbung

Um die Anforderungen der Automatisierungstechnik an das Netzwerk zu erfüllen, wird im ersten Teil dieses Kapitels die Notwendigkeit einer Zeitablaufsteuerung (engl.: Scheduling) begründet, mit dessen Hilfe der notwendige Determinismus im Standard-Ethernet erreicht werden soll. Im Anschluss daran werden grundlegende Begriffe der Graphentheorie eingeführt, die im weiteren Verlauf der Arbeit auf zwei Arten von Bedeutung sind:

Das Netzwerk selbst lässt sich als Graph modellieren, wie es beispielsweise Erlebach [Erl98] in seiner Dissertation durchgeführt hat. Verteiler wie Hubs oder Switches werden dabei als Knoten, die Verdrahtung als Kanten im Graph modelliert.

Sendungen, die Kollisionen auf einer Leitung oder eine Pufferung in den Switches verursachen, sind durch die Erstellung einer Schedule zeitlich zu entzerren. Das Problem der Erzeugung einer Schedule, also eines Zeitablaufplans, lässt sich auf ein Problem der Graphenfärbung zurückführen. Aus einem gefärbten Graphen lässt sich dann unmittelbar eine Schedule erstellen. [UB05]

Im Anschluss an die graphentheoretischen Grundlagen werden gängige Algorithmen zur Färbung von Graphen diskutiert. Abschließend wird gezeigt, wie mit gefärbten Graphen eine Schedule entwickelt werden kann. Ein Großteil dieser Grundlagen wurde in Zusammenarbeit mit Nowak [Now06] erstellt unter Verwendung der Arbeiten von Jungnickel [Jun99], Volkmann [Vol91] und Diestel [Die06]; die Sätze werden von Nowak [Now06] formal bewiesen.

2.3.1 Deterministischer Medienzugang, Graphen und Scheduling

Wie bereits beschrieben wurde, ist der fehlende Determinismus des Ethernets bei einer Halbduplexübertragung auf das Auftreten von Kollisionen und anschließendem nicht-deterministischen Backoff (s. Kapitel 2.2.2.1) zurückzuführen. Bei einer Vollduplexübertragung existieren zwar keine Kollisionen, jedoch führt das Zwischenspeichern von Frames in Switches und die Verwerfung von Frames bei vollen Puffern (s. Kapitel 2.2.3.2) ebenfalls zu einem nicht-deterministischen Verhalten.

Auf der anderen Seite existieren hohe Anforderungen der Automatisierungstechnik an ein deterministisches Verhalten des Netzwerks, insbesondere in der harten IAONA-Echtzeitklasse (s. Kapitel 2.1.3.3). Die Anforderungen, welches echtzeitfähige Gerät in welchen Intervallen an andere Geräte senden will, ist aufgrund der Anlagenkonfiguration

im Vorfeld bekannt. Ein ereignisbasiertes deterministisches Verhalten, z. B. bei Betätigung eines Not-Aus Tasters, lässt sich durch die Reservierung eines Zeitslots (s. Abbildung 2.26) in jedem Produktionszyklus bzw. in Vielfachen von Produktionszyklen realisieren. Tritt das Ereignis nicht auf, so bleibt das Zeitfenster leer oder es wird ein anderes Ereignis übertragen, welches nicht gleichzeitig mit dem ersten Ereignis auftreten kann.

Bereits im Jahre 1985 wurde von Shimokawa und Shiobara [SS85] ein Verfahren vorgestellt, mit dem zeitkritische Daten in industriellen Anwendungen mittels Ethernet übertragen werden können. Statt dem im Ethernet typischen CSMA/CD-Protokoll und dem in dieser Zeit verbreiteten Token-Passing Verfahren, vgl. ProfiBus in Kapitel 2.1.3.6, erhält jedes Gerät für jede Sendung eine zuvor konfigurierte Sequenznummer, welche der Nummer eines Zeitslots entspricht. Nach der Hochlaufphase der industriellen Anlage leitet ein Gerät als Master die Sequenz durch das Senden einer Nachricht zur Synchronisation ein.

In auf Switches basierenden Netzwerken können Sendungen, die sich nicht überlappen, gleichzeitig ausgeführt werden, wie die Übertragungen U_1 bis U_4 in Abbildung 2.37. Sie können also die gleiche Sequenznummer Z_x erhalten. Sind alle Zeitslots abgearbeitet, so beginnt die Abarbeitung wieder bei dem Zeitslot Z_1. Dies entspricht dem zyklischen Betrieb einer automatisierten Anlage; bei der notwendigen Zeit zur Abarbeitung aller Zeitslots handelt es sich um die Zykluszeit T_Z der Anlage. Die Ausführung eines Zyklus lässt sich vergleichen mit einer Umdrehung einer zentralen mechanischen Welle, an der mehrere Kurvenscheiben angeschlossen sind (vgl. Kapitel 2.1.2). Bei einer Umdrehung führt der an der jeweiligen Kurvenscheibe angeschlossene Anlagenteil die Verfahrbewegung aus, welche durch die Kurvenscheibe vorgegeben wird.

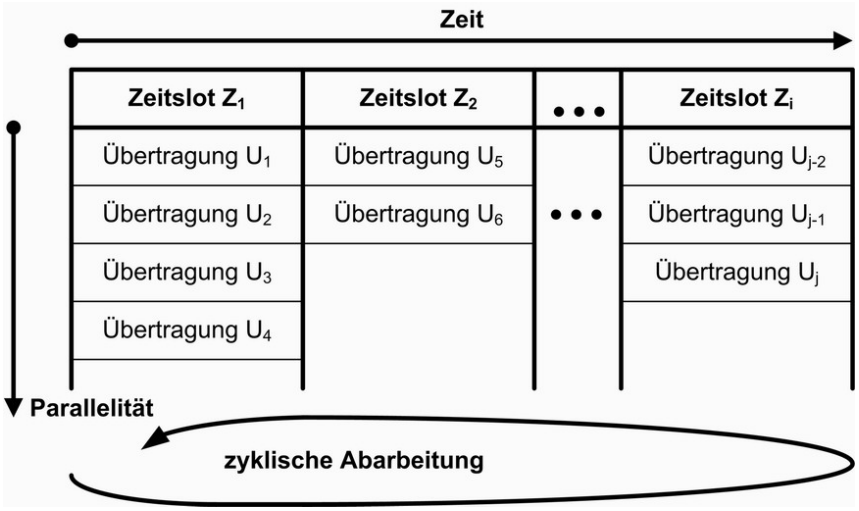


Abbildung 2.37: Schedule von Übertragungen

Durch die Einführung eines solchen Scheduling-Verfahrens im Ethernet nach IEEE 802.3 wird im folgenden Verlauf dieser Arbeit ein Determinismus eingeführt, der den Anforderungen der Automatisierungstechnik genügen soll. Da in der Automatisierungstechnik sowohl die Anforderungen der echtzeitfähigen Geräte, als auch die Infrastruktur des Netzwerkes im Vorfeld bekannt und unveränderlich sind, handelt es sich um ein Offline-Scheduling Verfahren.

2.3.2 Definitionen aus der Graphentheorie

Zur Modellierung der Netzwerkinfrastruktur sowie zur Erzeugung von Schedules werden in diesem Kapitel grundlegende Begriffe aus der Graphentheorie definiert, die im Wesentlichen von Nowak [Now06] übernommen werden.

Definition 2.3.1: Potenzmenge
Sei M eine Menge. Dann ist die Potenzmenge $P(M) := \{A : A \subseteq M\}$ die Menge aller Teilmengen von M.

Definition 2.3.2: Menge zu einer natürlichen Zahl
Zu einer natürlichen Zahl n ist $\underline{n} := \{0, ..., n-1\}$.

Definition 2.3.3: n-elementige Teilmenge
Sei M eine Menge. Dann ist die Menge aller höchstens n-elementigen Teilmengen $M^{\underline{n}} := \{A \mid A \subseteq M \wedge |A| \leq n\}$.

Definition 2.3.4: (ungerichteter) Graph
Seien V und E disjunkte Mengen und $g : E \mapsto V^{\underline{2}}$. Dann wird das Tripel $G := (V, E, g)$ als (ungerichteter) Graph bezeichnet. Die Elemente aus V nennt man Knoten, die Elemente aus E Kanten. Die Randabbildung g ordnet jeder Kante ihre Randknoten zu.

Definition 2.3.5: (ungerichteter) Hypergraph
Seien V und E disjunkte Mengen und $g : E \mapsto P(V)$. Dann wird das Tripel $G := (V, E, g)$ als (ungerichteter) Hypergraph bezeichnet. Die Elemente aus V nennt man Knoten, die Elemente aus E Kanten und die Elemente aus $g(e)$ Randknoten oder Knoten der Kante e.

Definition 2.3.6: Teilgraph, Untergraph
Sei $G = (V, E, g)$ ein Graph. Ist $V' \subset V$ und $E' \subset E$, so dass $G' := (V', E', g_{|E'})$ ein Graph ist, dann heißt G' Teilgraph von G. Man schreibt dann $G' \subset G$. Der Teilgraph G' heißt induziert von V' in G, wenn $E' = \{xy : x, y \in V'\} \cap E$ ist. Dann schreibt man $G' = G[V']$. Ein induzierter Teilgraph heißt auch Untergraph.

Definition 2.3.7: Randknoten, inzident, adjazent, benachbart
Sei $G = (V, E, g)$ ein Graph. Dann ist ein Knoten $v \in V$ inzident mit einer Kante $e \in E$, wenn $v \in g(e)$ ist. Zwei Kanten $e, f \in E$ sind adjazent, falls sie einen gemeinsamen Randknoten besitzen. Dies ist der Fall, wenn ein Knoten v existiert, der inzident mit e und f ist. Zwei Knoten $v, w \in V$ sind benachbart, wenn sie durch eine gemeinsame Kante miteinander verbunden werden. Dann muss also eine Kante e existieren mit $g(e) = \{v, w\}$.

Die mit einer Kante e inzidenten Knoten sind die Randknoten von e. Eine Kante mit den Randknoten v und w wird als vw bezeichnet. In einfachen Graphen ist $vw = \{v, w\} = wv$. In Multigraphen muss die Kante vw nicht eindeutig sein, dann sei vw eine beliebige Kante mit Randknoten v und w.

Ist $X, Y \subset V$ und $v \in X$ und $w \in Y$, so heißt vw eine X-Y-Kante. Die Menge aller X-Y-Kanten in E wird mit $E(X,Y)$ bezeichnet. Man schreibt auch $E(v,Y) := E(\{v\},Y), E(X,w) := E(X,\{w\})$ und $E(v,w) := E(\{v\},\{w\})$. Weiter wird $E(v) := E(v, V \setminus \{v\})$ gesetzt.

Definition 2.3.8: Grad eines Knotens, Minimal-/Maximalgrad
Sei $G = (V,E,g)$ ein Graph und $v \in V$. Dann ist der Grad des Knotens $d(v) := |E(v)|$ die Anzahl der mit v inzidenten Kanten. Der Minimalgrad des Graphen ist definiert als $\delta(G) := \min\{d(v) : v \in V\}$, der Maximalgrad als $\Delta(G) := \max\{d(v) : v \in V\}$.

Definition 2.3.9: Multiplizität
Sei $G = (V,E,g)$ ein Graph. Dann bezeichnet $\mu(G) := max\{|E(v,w)| : v,w \in V\}$ die Multiplizität des Graphen, also die maximale Anzahl paralleler Kanten.

Definition 2.3.10: einfacher Graph, Multigraph
Ein Graph $G = (V,E,g)$ ist einfach, wenn g injektiv ist, also $\mu(G) \leq 1$ gilt. In diesem Falle kann jede Kante $e \in E$ eindeutig mit ihren Randknoten $g(e) \in V^2$ identifiziert werden. Daher lässt sich $E' = \{g(e) : e \in E\}$ setzen, so dass man für $G = (V,E,g)$ auch $G := (V,E')$ schreiben kann. Einen nicht einfachen Graphen nennt man Multigraph.

Multigraphen $G = (V,E,g)$ können auf verschiedene Arten dargestellt werden. Bei einer Speicherung in einer *Adjazenzmatrix* werden die Knoten mit 0 bis $n-1$ durchnummeriert und in einer $n \times n$-Matrix $(a_{ij})_{1 \leq i,j \leq n}$ wird in der Komponente a_{ij} die Anzahl der Kanten zwischen i und j gespeichert.

Bei einer Speicherung in einer *Adjazenzliste* werden die Knoten in einer (doppelt) verketteten Liste oder einem Array gespeichert. Jeder Knoten enthält eine Liste aller mit ihm inzidenten Kanten, wobei jede Kante eine Referenz auf ihre Randknoten enthält.

Definition 2.3.11: bipartiter Graph
Sei $G = (V,E)$ ein einfacher Graph. Lassen sich alle Knoten in genau zwei disjunkte Mengen $A, B \subset V$ aufteilen, so dass für jede Kante $e = \{v,w\} \in E$ entweder $v \in A$ und $w \in B$ oder $v \in B$ und $w \in A$ gilt, so spricht man von einem bipartiten Graphen.

Definition 2.3.12: endlicher Graph
Ein Graph $G = (V,E,g)$ mit $|V| < \infty$ und $|E| < \infty$ bezeichnet man als endlichen Graphen.

Im weiteren Verlauf dieser Arbeit werden alle Graphen endlich sein, da stets eine Obergrenze der Knoten und Kanten angegeben werden kann.

Definition 2.3.13: Schlinge, schlingenfreier Graph
Sei $G = (V,E,g)$ ein Graph. Eine Kante $e \in E$ mit $g(e) = \{v\}$ heist Schlinge. Ein Graph ohne Schlinge ist schlingenfrei.

Definition 2.3.14: Kantengraph
Sei $G = (V, E)$ ein einfacher Graph. Der Kantengraph $L(G)$ ist definiert durch $L(G) := (V', E')$ mit $V' = E$ und $E' = \{\{e, f\} \in (V')^{\underline{2}} : e \cap f \neq 0\}$.

Die Knoten von $L(G)$ sind also die Kanten von G. Zwei Knoten von $L(G)$ sind genau dann benachbart, wenn sie als Kanten von G adjazent sind.

Definition 2.3.15: Clique, maximale Clique, größte Clique
Sei $G = (V, E)$ ein einfacher, schlingenfreier Graph und $X \subset V$. Man bezeichnet X als Clique von G, wenn für je zwei beliebige verschiedene Knoten $v, w \in X$ gilt, dass sie durch eine Kante miteinander verbunden sind. Eine Clique X in G ist maximal, wenn man keinen weiteren Knoten $u \in V$ zu X hinzufügen kann, so dass X zusammen mit u eine Clique ist. Existiert in G keine Clique, die mehr Elemente als X enthält, so nennt man X größte Clique.

Definition 2.3.16: (maximal) unabhängige Menge
Sei $G = (V, E, g)$ ein Graph. Eine Menge $X \subset V$ heißt unabhängig in G, wenn keine zwei Knoten aus X benachbart sind. Eine unabhängige Menge ist maximal, wenn man keinen weiteren Knoten $v \in V$ zu X hinzufügen kann, so dass X zusammen mit v eine unabhängige Menge bildet.

Definition 2.3.17: Pfad
Ein Pfad $P = (\bar{x}, \bar{e})$ in G besteht aus einer Folge $\bar{x} = (x_i)_{i=0,...,n}$ von Knoten aus V und einer Folge $\bar{e} = (e_i)_{i=1,...,n}$ von Kanten aus E, so dass für alle $i = 1, ...n$ gilt: $e_i = x_{i-1}x_i$.

Definition 2.3.18: Äquivalenzrelation
Sei $G = (V, E, g)$ ein Graph. Die Relation $\sim \subset V \mathrm{x} V$, $v \sim w \Leftrightarrow$ „Pfad von v nach w existiert“ ist eine Äquivalenzrelation. Das Lemma wurde von Nowak [Now06, Lemma 2.2.12] bewiesen.

Definition 2.3.19: Äquivalenzklasse
Ist R eine Äquivalenzrelation auf einer Menge M, so nennt man für ein $a \in M$ die Teilmenge $[\alpha]_R = \{x \in M | \mathrm{x} \sim_R \alpha\} \subseteq M$ die R-Äquivalenzklasse von α.

Definition 2.3.20: Weg, Länge eines Weges
Sei $G = (V, E, g)$ ein Graph. Ein Weg $P = (\bar{v}, \bar{e})$ in G ist ein Pfad mit paarweise verschiedenen Knoten. Die Knoten v_0 und v_n sind Randknoten. Man spricht von einem Weg von v_0 nach v_n. Die Menge der Knoten bzw. Kanten des Weges P wird mit $V_P = \{v_0, ..., v_n\}$ bzw. $E_P = \{e_1, ...e_n\}$ bezeichnet. Die Anzahl $l(P) := n$ der Kanten von P heißt Länge des Weges.

Für einfache Graphen wird $P = (v_0, ..., v_n)$ definiert[1]. Für einen Weg P und $0 \leq i \leq j \leq n$ wird $v_iPv_j := (v_i, ..., v_j)$, $v_iP := (v_i, ..., v_n)$ und $Pv_j := (v_0, ..., v_j)$ definiert.

Definition 2.3.21: inverser Weg, Teilweg
Sei $G = (V, E, g)$ ein Graph und $P = (V_P, E_P)$ ein Weg in G von v_0 nach v_n. Dann ist der inverse Weg $P^{-1} = ((v_n, ..., v_0), (e_n, ..., e_0))$ ein Weg von v_n nach v_0.
Für $0 \leq i \leq j \leq n$ ist $Q = ((v_i, ..., v_j), (e_{i+1}, ..., e_j))$ ein Teilweg in P. Man schreibt $Q \triangleleft P$ und sagt, dass P den Weg Q enthält.

Definition 2.3.22: Verkettung von Wegen
Seien $P = ((v_0, ..., v_n), (e_1, ..., e_n))$ und $Q = ((v_n, ..., v_{n+m}), (e_{n+1}, ..., e_{n+m}))$ Wege. Dann definiert man die Verkettung durch $P + Q = ((v_0, ..., v_{n+m}), (e_1, ..., e_{n+m}))$. Diese Verkettung ist genau dann ein Weg, wenn die Knoten beider Wege paarweise verschieden sind.

Definition 2.3.23: Bezeichnungen für Wege
Zu einem Weg $P = ((v_0, ..., v_n), (e_1, ..., e_n))$ wird folgende Notation vereinbart:
Für $i = 0, ..., n$: $e_i^{P-} := v_{i-1}$, $e_i^{P+} := v_i$
Für $i = 0, ..., n$: $v_i^{P-} := e_i$
Für $i = 0, ..., n-1$: $v_i^{P+} := e_{i+1}$
Des Weiteren wird

$$E_P(v_i) = \left\{(v_i)^{P-}, (v_i)^{P+}\right\} = \begin{cases} \{v_0v_1\} & i = 0 \\ \{v_{i-1}v_i, v_iv_{i+1}\} & \text{für } 1 \leq i \leq n-1 \\ \{v_{n-1}v_n\} & i = n \end{cases}$$

definiert.

Definition 2.3.24: Kreis
Sei $G = (V, E, g)$ ein Graph und $P = (\bar{v}, \bar{e})$ ein Pfad in G mit der Folge von Knoten $(v_0, ..., v_n)$. Ist $v_0 = v_n$, so spricht man von einem Kreis.

Definition 2.3.25: Baum, Blatt
Ein zusammenhängender schlaufenfreier Graph $G = (V, E)$ heißt Baum. Die Knoten $v_i \in V$ mit Grad $d(v_i) = 1$ heissen Blätter.

[1]Dies ist möglich, da bei einem einfachen Graphen der Weg $P = (\bar{v}, \bar{e})$ bereits durch die Folge seiner Knoten $(v_0, ..., v_n)$ und jede Kante e_i durch die Bezeichnung $e_i = v_{i-1}v_i$ bereits eindeutig festgelegt ist.

Satz 2.3.26: Eigenschaften eines Baumes
Sei $G = (V, E)$ ein Baum. Dann gilt:

1. Ist G nicht trivial, das heißt $|V| \geq 2$, so hat G mindestens zwei Blätter.
2. Zu je zwei verschiedenen Knoten $v, w \in V$ existiert ein eindeutiger Weg von v nach w in G.
3. Jeder zusammenhängende Untergraph von G ist ein Baum.
4. Seien $X, Y \subset V$ disjunkte Mengen und $G[X]$ und $G[Y]$ Bäume. Dann gibt es ein $x \in X$ und $y \in Y$, so dass jeder Weg von X nach Y in G den eindeutig bestimmten Weg P von x nach y und jeder Weg von Y nach X den inversen Weg P^{-1} enthält.

Definition 2.3.27: maximale Länge eines Weges im Baum
Sei $G = (V, E)$ ein Baum. Dann bezeichnet $l(G)$ die maximale Länge eines Weges in G.

2.3.3 Färbung von Graphen

Wie bereits in der Einleitung dargestellt wurde, soll der notwendige Determinismus im weiteren Verlauf dieser Arbeit durch die Einführung von Zeitslots im Rahmen eines TDMA-Zugriffsverfahrens erreicht werden. Unabhängige Übertragungen können dabei zeitgleich ablaufen, während IRT-Übertragungen, die zu einer Kollision oder einer Zwischenspeicherung in einem Switch führen, zu verschiedenen Zeiten ausgeführt werden müssen. Sowohl die IRT-Übertragungen, als auch die Informationen zu möglichen Kollisionen bzw. Zwischenspeicherungen werden im Verlauf dieser Arbeit durch Graphen repräsentiert. Durch eine Färbung dieser Graphen sollen unabhängige IRT-Übertragungen ermittelt werden, welche zeitgleich ausführbar sind. Das Ziel ist dabei die Erstellung einer Offline-Schedule. In diesem Kapitel wird nun die Graphenfärbung definiert sowie gängige Algorithmen vorgestellt.

Definition 2.3.28: Knotenfärbung, Kantenfärbung
Sei $G = (V, E, g)$ ein (Multi-)Graph oder (Multi-)Hypergraph. Eine Knotenfärbung von G ist eine Abbildung $c : V \mapsto S$ in eine endliche Menge S, so dass $c(v) \neq c(w)$ für benachbarte Knoten $v, w \in V$ gilt. Dabei wird G als schlingenfrei vorausgesetzt, da $c(v) \neq c(v)$ bei einer Schlinge um v nicht erfüllbar ist.
Eine Kantenfärbung von G ist eine Abbildung $c : E \mapsto S$ in eine endliche Menge S, so dass $c(e) \neq c(f)$ für adjazente Kanten $e, f \in E$ gilt.

Definition 2.3.29: knotenchromatischer/kantenchromatischer Index
Sei $G = (V, E, g)$ ein schlingenfreier Graph. Dann ist der knotenchromatische Index $\chi(G)$ bzw. der kantenchromatische Index $\chi'(G)$ wie folgt definiert:
$\chi(G) = \min \{k \in \mathbb{N} : \text{Es existiert eine Knotenfärbung } c : V \mapsto \underline{k}\}$
$\chi'(G) = \min \{k \in \mathbb{N} : \text{Es existiert eine Kantenfärbung } c : E \mapsto \underline{k}\}$

Der knotenchromatische Index wird oft lediglich als chromatische Zahl bezeichnet. Der resultierende Wert beider Indizes wird auch als benötigte Anzahl Farben für G bezeichnet.

Definition 2.3.30: k-färbbar, k-Färbung
Ein Graph $G = (V, E, g)$ ist k-knotenfärbbar, wenn $\chi(G) \leq k$ ist und k-kantenfärbbar, wenn $\chi'(G) \leq k$ ist. Eine Knoten- bzw. Kantenfärbung, welche maximal k Farben benötigt, heißt k-Knotenfärbung bzw. k-Kantenfärbung.

Satz 2.3.31: Kantenfärbung von G entspricht Knotenfärbung von $L(G)$
Sei $G = (V, E, g)$ ein Graph. Dann ist $c : E \mapsto S$ genau dann eine Kantenfärbung von G, wenn es eine Knotenfärbung des Kantengraphen $L(G)$ ist.

Satz 2.3.31 sagt aus, dass ein Graph, der knotengefärbt werden soll, stets in polynomieller Zeit in einen äquivalenten Graphen umgewandelt werden kann, der kantengefärbt werden soll, und umgekehrt. Im weiteren Verlauf der Arbeit wird jedoch deutlich, dass an manchen Stellen die Modellierung mit einem knotenzufärbenden Graphen anschaulicher ist als mit einem kantenzufärbenden Graphen und umgekehrt. Es werden also im Folgenden beide Arten der Färbung verwendet.

Satz 2.3.32: Graphenfärbung ist NP-vollständig
Das Entscheidungsproblem, ob ein gegebener Graph mit n Farben färbbar ist, ist NP-vollständig. Insbesondere gibt es, falls $P \neq NP$, keinen polynomiellen Algorithmus zur Bestimmung des chromatischen Index und somit auch keinen polynomiellen Algorithmus zur Bestimmung einer Färbung mit minimaler Farbanzahl. Dies gilt sowohl für die Knotenfärbung, als auch für die Kantenfärbung.

Auf die Definition der Klassen P und NP sowie auf die Beweisführung der NP-Vollständigkeit wird an dieser Stelle verzichtet. Die entsprechenden Definitionen und die Beweisführung sind in den Arbeiten von Nowak [Now06] sowie von Aho, Hopcroft und Ullman [AHU88] zu finden. Im weiteren Verlauf dieser Arbeit besteht die Zielsetzung darin, auf die Anwendung von nicht-polynomiellen Algorithmen zu verzichten. Statt dessen sollen polynomielle Heuristiken zum Einsatz kommen oder die Problemstellungen so eingeschränkt werden, dass auch exakte Färbungen in polynomieller Zeit durchgeführt werden können.

2.3.3.1 Abschätzungen zur Graphenfärbung

Sowohl für die knotenbasierte, als auch für die kantenbasierte Färbung von Graphen lassen sich Abschätzungen über die chromatischen Indizes angeben.

Satz 2.3.33: Satz von Brooks [Ste05]
Sei G ein schlingenfreier Graph, dann gilt für die Knotenfärbung $\chi(G) \leq \Delta(G) + 1$. Ist G zusammenhängend, nicht vollständig und beinhaltet keinen Kreis ungerader Länge, so gilt $\chi(G) \leq \Delta(G)$.

Nowak [Now06] merkt jedoch zurecht an, dass diese Abschätzungen äußerst grob sein können, da der Maximalgrad $\Delta(G)$ eines 4-färbbaren Graphen bereits beliebig hoch sein kann. Für die kantenbasierte Färbung können hingegen bessere Abschätzungen getroffen werden:

Satz 2.3.34: Satz von Vizing [Gör05]
Sei $G = (V, E, g)$ ein Multigraph. Dann gilt für die Kantenfärbung $\Delta(G) \leq \chi'(G) \leq \Delta(G) + \mu(G)$. Ist der Graph einfach, so gilt $\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$.

Die Bestimmung, ob für einen konkreten einfachen Graphen $\chi'(G) = \Delta(G)$ oder $\chi'(G) = \Delta(G) + 1$ gilt, ist jedoch bereits NP-vollständig. Lediglich für die Unterklasse der bipartiten Graphen ist eine exakte Aussage über den kantenchromatischen Index möglich, die in dem folgenden Satz formuliert wird. Für diese Klasse existieren auch effiziente Algorithmen, welche eine exakte Färbung in geringer Laufzeit ermöglichen.

Satz 2.3.35: Kantenchromatischer Index eines bipartiten Multigraphen [Vol91]
Sei $G = (V, E, g)$ ein bipartiter Multigraph. Dann gilt für die Kantenfärbung $\chi'(G) = \Delta(G)$.

2.3.3.2 Algorithmen zur Knotenfärbung

Der Fokus dieses Kapitels liegt auf der Strategie, nach der ein Graph gefärbt wird. Im Folgenden werden drei bedeutende Strategien skizziert, auf deren Basis jeweils sowohl exakte Algorithmen, als auch Heuristiken entwickelt werden können [Now06]. Eine Heuristik garantiert keine optimale Färbung mit $\chi(G)$ Farben, sondern liefert eine Näherungslösung bei polynomieller Laufzeit.

Als vierte Strategie werden Metaheuristiken skizziert, die ebenso wie Heuristiken Näherungslösungen finden. Im Gegensatz zu problemspezifischen Heuristiken können Metaheuristiken auf beliebige Problemstellungen angewendet werden. Im Umfeld dieser Arbeit wurden in dem Zusammenhang die Tabu-Suche und Genetische Algorithmen [Agb07] untersucht.

Im weiteren Verlauf dieser Arbeit wird ein Framework entwickelt, das aufgrund seiner Modularität beliebige Strategien und Algorithmen zur Färbung zulässt.

1. Strategie: Sequentielle Knotenfärbung

Bei der sequentiellen Färbung wird die Knotenmenge V eines Graphen in einer Reihenfolge $v_1, ...v_i, ..., v_n$ durchlaufen. Dem aktuell ausgewählten Knoten v_i wird dann die kleinst mögliche Farbe zugeordnet, die noch kein Nachbarknoten von v_i besitzt. Dabei wird ggf. eine neue Farbe vergeben. Die Färbung von v_i wird im Nachhinein nicht mehr verändert. Es handelt sich also um ein Greedy-Verfahren. Eine einfache Greedy-Heuristik, wie von Nowak [Now06] als *Greedy-Seq-Color* vorgestellt, erzeugt eine Färbung mit maximal $\Delta(G) + 1$ Farben. Bei einer Speicherung als Adjazenzliste besitzt dieser Algorithmus eine Laufzeitkomplexität von $O(|E| + |V|)$, bei einer Speicherung als Adjazenzmatrix eine Laufzeitkomplexität von $O(|V|^2)$. Bei einer ungeschickten Wahl der zu färbenden Knotenreihenfolge kann ein Greedy-Verfahren wesentlich viel mehr Farben als $\chi(G)$ benötigen. Des Weiteren existieren Graphenfamilien, bei denen eine sequentielle Färbung bei nahezu allen Knotenreihenfolgen schlechte Resultate liefert [Now06]. Jedoch besitzt die sequentielle Färbung eine interessante Eigenschaft:

Satz 2.3.36: Sequentielle Färbung liefert für eine Knotenreihenfolge optimale Färbung [Tur04]
Sei $G = (V, E, g)$ ein Graph. Dann existiert eine Knotenreihenfolge in der Art, dass die sequentielle Färbung eine optimale Färbung mit $\chi(G)$ Farben liefert.

Die Güte einer sequentiellen Färbung ist demnach abhängig von der Wahl der Reihenfolge, in der die Knoten gefärbt werden. Die *DSATUR*-Heuristik (Funktionsweise s. Kapitel 2.3.3.3) ist ein dynamisches Verfahren zur Bestimmung der Anordnung der Knoten, nimmt jedoch gleichzeitig eine sequentielle Färbung in $O(\min(|V|^2, |E|log|V|))$ vor. Aus diesem Grunde wird der Algorithmus sowohl in diesem Kapitel zur Färbung, als auch im folgenden Kapitel zur Knotenanordnung erwähnt. Auch wenn seine theoretische Güte nicht überzeugt [Now06], so hat sich dieser Algorithmus in der Praxis als sehr effizient erwiesen.

Viele gängige exakte Algorithmen, von denen im Folgenden vier vorgestellt werden, basieren auch auf der sequentiellen Färbung. Deren Ziel ist es, die Vielzahl an möglichen Färbungen möglichst schnell einzuschränken, so dass lediglich ein Pfad zur optimalen Färbung aus dem Lösungsbaum mit allen Möglichkeiten resultiert. Zur Laufzeitoptimierung kann dabei eine untere Grenze für $\chi(G)$ angegeben werden. Dies ist eine Anzahl an Farben, mit welcher der Anwender zufrieden ist. Ist eine solche Lösung gefunden, so kann der Algorithmus unverzüglich terminieren. Diese untere Grenze kann auch aufgrund einer bekannten großen Clique entstehen, da mindestens die Anzahl an Farben benötigt wird, wie diese Clique Knoten besitzt.

Die Färbung wird mittels sequentiellem *Backtracking* ermittelt, die erstmals von Brown [Bro72] veröffentlicht wurde. Dabei unterscheidet man Vorwärts- und Rückwärtsoperationen. In der Vorwärtsoperation wird sequentiell gefärbt. Besitzt eine Teilfärbung gleich viele oder mehr Farben wie eine bereits gefundene Lösung, so wird sie verworfen. Sind alle Knoten gefärbt, so wird im nächsten Durchlauf versucht, eine Färbung mit weniger Farben zu finden. In der Rückwärtsoperation wird der letzte Knoten gesucht, der mit einer anderen Farbe gefärbt werden kann. Von dort aus startet wiederum die Vorwärtsoperation. Dies geschieht so lange, bis alle möglichen Färbungen betrachtet wurden oder eine zuvor eingegebene untere Grenze für $\chi(G)$ erreicht worden ist.

Auf Basis dieser Vorgehensweise wurden im Laufe der Zeit weitere Verbesserungen an dem Algorithmus vorgenommen, die Laufzeitkomplexität von $O((|V| + |E|) \cdot V!)$ bleibt jedoch bestehen und ist damit wie zu erwarten wesentlich höher als bei den untersuchten Heuristiken. Statt die theoretische Laufzeitkomplexität zu verbessern, liegen die Bestrebungen in einer Verbesserung der Geschwindigkeit des Algorithmus in der Praxis.

Brown [Bro72] beschreibt bereits selbst einen Look-Ahead Mechanismus. Dabei wird in der Vorwärtsoperation bei der Färbung des Knotens v_i untersucht, ob dieser einen benachbarten Knoten v_j mit $j > i$ besitzt, der nur mit genau einer bislang verwendeten Farbe s gefärbt werden kann. Ist dies der Fall, so ist s keine zulässige Farbe für v_i und es wird versucht, v_i mit einer anderen bislang verwendeten Farbe zu färben, möglichst ohne eine neue Farbe zu vergeben. Damit wird die Anzahl der Rückwärts-Operationen minimiert.

Der Algorithmus von Christofides [Chr75], der von Kubale und Jackowski [KJ85] korrigiert wurde, verbessert die Rückwärtsoperation. Die Rückwärtsoperation wurde nochmals

von Brélaz [Bre79] verbessert und anschließend von Peemöler [Pee83] korrigiert. Zusätzlich dazu optimiert Brélaz die Initialisierung seines exakten Algorithmus, indem er zunächst eine DSATUR-Heuristik anwendet und die resultierende Knotenanordnung als statische Anordnung (s. Kapitel 2.3.3.3) für den exakten Algorithmus verwendet.

2. Strategie: Sequentielle Färbung mit Umfärben

Der sequentiellen Färbung können zusätzlich Maßnahmen zum Umfärben hinzugefügt werden. Dabei wird zunächst eine sequentielle Färbung durchgeführt. Ist jedoch das Einführen einer neuen Farbe für einen Knoten v_i notwendig, so wird versucht, die bisher bestehende Färbung so zu verändern, dass die bislang verwendete Anzahl an Farben auch mit dem gefärbten Knoten v_i ausreichend ist. Mit einer Laufzeitkomplexität von $O(|V|\cdot|E|)$ führt die Heuristik *Greedy-SeqI-Color* [Now06] eine solche Färbung durch, deren Güte in Verbindung mit einer statischen Anordnung der Knoten (s. Kapitel 2.3.3.3) weiter verbessert werden kann.

Eine Umfärbung kann ebenso in den DSATUR-Algorithmus implementiert werden. Der Algorithmus *DSATURI* wurde von Kubale und Jackowski [KJ85] vorgestellt und färbt ebenfalls in $O(|V| \cdot |E|)$.

3. Strategie: Färbung mittels unabhängiger Mengen

Das Ziel einer Färbung mittels unabhängiger Mengen ist es, die Knotenmenge des Graphen in unabhängige Mengen bzw. Klassen aufzuteilen. Alle Knoten, die sich innerhalb einer solchen Menge befinden, sind nicht benachbart und können einheitlich gefärbt werden. Die Anzahl der Mengen entspricht demnach der Anzahl der benötigten Farben. Liegt ein Knoten in mehreren unabhängigen Mengen, so kann er mit einer beliebigen Farbe gefärbt werden, die seinen Mengen zugeordnet wird.

Der von Nowak [Now06] vorgestellte exemplarische Algorithmus zur Färbung mit unabhängigen Mengen kann in drei Teile geteilt werden. Der erste Teil, *Greedy-IS-Color*, färbt eine gegebene Menge von Knoten mit einer einheitlichen Farbe und inkrementiert den Zähler der verwendeten Farben. Die maximal unabhängigen Mengen der Knoten, die noch nicht gefärbt sind, werden zuvor von *Greedy-MaxIS* [Now06] [Joh74] ermittelt. Hier wiederum ist die Frage, in welcher Reihenfolge die Knoten ausgewählt werden. Nach Aussage von Johnson soll dies jeweils der Knoten sein, der den geringsten Grad im ungefärbten Teil des Graphen besitzt. Der Algorithmus zur Ermittlung dieses Knotens wird als *Johnson-Color* [Joh74] bezeichnet, der nach [Wig83] in $O(|V|^2)$ implementiert werden kann.

Der von Eppstein [Epp03] vorgestellte auf unabhängigen Mengen basierte exakte Algorithmus garantiert mit $O(2.4150^{|V|})$ die bisher beste asymptotische Laufzeit. Diese Laufzeit wird insbesondere darauf verwendet, die benötigte Anzahl der Farben, also $\chi(G)$, zu ermitteln. Im Anschluss daran wird die Färbung durchgeführt. Diese beste Worst-Case-Laufzeit tritt jedoch in fast jedem Falle ein, so dass in der Praxis häufiger auf sequentieller Färbung basierte Algorithmen zum Einsatz kommen.

Abschließend sei noch auf eine weitere effiziente Methode zur Färbung mittels unabhängiger Mengen hingewiesen. Hier wird das Modell der unabhängigen Mengen in ein *lineares ganzzahliges Optimierungsproblem* (integer linear program, ILP) in der Art umgewandelt,

dass das ILP bei einer exakten Färbung eine optimale Lösung liefert. Eine Übersicht zur linearer Programmierung liefert Nowak [Now06], eine ausführliche Abhandlung der Thematik wurde von Ignizio und Cavalier [IC94] durchgeführt.

4. Strategie: Anwendung von Metaheuristiken

Eine große Anzahl von Metaheuristiken basiert auf der lokalen Suche. Im Umfeld dieser Arbeit wurde von Agbanzo [Agb07] der Algorithmus der *Tabu-Suche* von Dorne und Hao [DH98] untersucht. Dabei wird zunächst eine beliebige gültige Startlösung gewählt, die im Falle der Graphenfärbung eine gültige Färbung eines Graphen darstellt. Diese Färbung ist meist das Resultat einer schnellen heuristischen Färbung, wie Greedy oder DSATUR. Die Aufgabe der Metaheuristik liegt darin, die Güte der Startlösung zu verbessern. Dazu wird im nächsten Schritt die Nachbarschaft definiert, also eine Umgebung der bisherigen Lösung. Bei der Graphenfärbung handelt es sich dabei um Färbungen, die sehr ähnlich zur Ausgangsfärbung sind. Die Nachbarschaft wird nach einer besseren Lösung abgesucht und die eventuell gefundene bessere Lösung wird als neue Basis verwendet. Um lokale Extrema zu vermeiden, wird die Tabu-Suche mit verschiedenen Startlösungen initialisiert. Der Begriff „Tabu“ ist erklärbar durch die Existenz einer Tabu-Liste. Diese Liste verhindert, dass bereits untersuchte Lösungen mehrfach betrachtet werden. Offensichtlich ist das Resultat der Tabu-Suche stark anhängig von der Güte der Startlösung.

Des Weiteren skizziert Agbanzo [Agb07] die Färbung von Graphen anhand von genetischen Algorithmen. Dabei wird eine Menge von bereits gefundene Lösungen als Chromosomen codiert und zu einer neuen Lösungen kombiniert bzw. einzelne Lösungen zufällig verändert. Dabei beschreibt Agbanzo eine konkrete Vorsortierung der Knoten als ein Chromosom, dessen Güte polynomiell durch eine Greedy-Färbung als Bewertungsfunktion ermittelt werden kann. Ziel ist es, eine optimale Vorsortierung zu finden. Die initiale Population wird gebildet durch eine Anzahl von - möglichst guten - Färbungen. Auf diese Menge von Chromosomen können dann Reproduktionsoperatoren wie der Crossover[2] zur Kombination von Lösungen und die Mutation[3] zur zufälligen Veränderung angewendet werden. Bei den genetischen Algorithmen sorgt die Mutation dafür, dass lokale Extrema des Lösungsraums überwunden werden können. Bevor eine neue Generation beginnt, werden Selektionsoperatoren angewendet. Dabei wird eine Anzahl von Chromosomen für die nächste Generation ausgewählt. Dies kann z. B. in einer Turnierauswahl [Kup06] geschehen.

In Abbildung 2.38 werden die vorgestellten Algorithmen nochmals zusammengefasst und nach exakten Algorithmen, Heuristiken und Metaheuristiken klassifiziert.

2.3.3.3 Heuristiken zur Knotenanordnung

Wie bereits erwähnt wurde, liefert die sequentielle Färbung, beispielsweise ein Greedy-Algorithmus, für eine bestimmte Knotenreihenfolge eine optimale Färbung. Das Problem der optimalen Färbung kann also verlagert werden auf die optimale Wahl der Knotenreihen-

[2]Die Kombination von zwei Eltern-Chromosomen zu einem neuen, besseren Chromosom.
[3]Die zufällige Veränderung eines Chromosoms.

Algorithmus **exakte Algorithmen**	Beschreibung zu finden in	Laufzeitkompexität
Brown Recursive-Seq-Color	[Bro72], [Now06]	$O((\|V\|+\|E\|)\cdot\|V\|!)$
Brown mit Look-Ahead	[Chr75], [KJ85]	
Christofides	[Bre79], [Pee83]	
Brélaz	[Bro72]	
Heuristiken		
Greedy Greedy-Seq-Color	[CH94]	$O(\|E\|+\|V\|)$ bzw. $O(\|V\|^2)$
Greedy mit Umfärbung Greedy-SeqI-Color	[SDK83], [Joh74]	$O(\|V\|\|E\|)$
Saturation Largest First DSATUR	[Bre79], [KJ85]	$O(\min(\|V\|^2, \|E\|\log\|V\|))$
Saturation Largest First mit Umfärbung DSATURI	[KJ85]	$O(\|V\|\|E\|)$
Färbung mittels unabhängiger Mengen Greedy-IS-Color + Greedy-MaxIS + Johnson-Color	[Joh74], [Wig83], [Now06]	$O(\|V\|^2)$ bzw. $O(\|V\|^3)$ $O(\|V\|)$ bzw. $O(\|V\|^2)$ $O(\|V\|^2)$
Metaheuristiken		
Tabu-Suche	[Agb07]	
genetische Algorithmen	[Agb07]	

Abbildung 2.38: Algorithmen zur Knotenfärbung

folge, die zu färben ist. Bei n Knoten existieren dabei $n!$ Möglichkeiten. Die Vorsortierung der Knoten ist optional, kann jedoch die Güte der entstehenden Färbung verbessern für den Fall, dass eine heuristische Färbung eingesetzt wird und erhöht die Geschwindigkeit der Lösungsfindung beim Einsatz eines exakten Algorithmus.

Im letzten Abschnitt wurde gezeigt, dass die Metaheuristiken den Suchraum aus diesen Möglichkeiten durch das Prinzip der Nachbarschaftsrelation einschränken. In diesem Kapitel werden nun gängige Heuristiken zur Knotenanordnung vorgestellt, die in direktem Zusammenhang mit der Färbung stehen. Neben einer *zufälligen Anordnung*, die in $O(|V|)$ vorgenommen werden kann, existieren statische und dynamische Heuristiken zur Wahl der Reihenfolge der zu färbenden Knoten. Eine statische Heuristik kann beispielsweise einem Greedy-Algorithmus vorgeschaltet werden, um dessen Güte zu verbessern.

Eine dynamische Heuristik besteht darin, dass sich die Reihenfolge der restlichen zu färbenden Knoten im Verlauf der Färbung ändern kann. Dies geschieht aufgrund der Annahme, dass man anhand der bereits vorgenommenen Färbung neue Informationen über die Schwierigkeit des restlichen zu färbenden Graphen erhält.

Bei der statischen *Largest First* (LF) Anordnung werden die Knoten nach ihrem Grad monoton fallend sortiert [WP67], so dass der Knoten mit dem höchsten Grad zuerst gefärbt wird. Man geht davon aus, dass Knoten mit hohem Grad generell schwieriger zu färben sind und daher zu einem Zeitpunkt gefärbt werden sollten, an dem nur wenige Farben bereits

verwendet wurden. Es ist wahrscheinlich, dass für die restlichen Knoten mit niedrigem Grad keine neuen Farben mehr verwendet werden müssen.

Der von Matula, Marble und Isaacson [MMI72] vorgestellte Algorithmus zur statischen *Smallest Last* (SL) Anordnung wählt zunächst einen Knoten v für die Knotenreihenfolge aus, der über den minimalen Grad des Graphen verfügt. Im Anschluss daran wird aus dem restlichen Graphen $G' = G \setminus \{v\}$ wieder ein Knoten mit minimalen Grad der Knotenreihenfolge hinzugefügt. Die der Reihenfolge bereits hinzugefügten Knoten werden demnach bei der Bestimmung des Grads nicht mehr betrachtet. Auf diese Weise werden zunächst die leicht zu färbenden Teile des Graphen der Knotenreihenfolge hinzugefügt und aus der Betrachtung der weiteren Reihenfolge entfernt. Die Knotenreihenfolge wird abschließend umgedreht, so dass wiederum die schwierig zu färbenden Knoten zuerst gefärbt werden.

In dem von Brélaz [Bre79] vorgestellten dynamischen *Saturation Largest First* Algorithmus (DSATUR) wird zunächst ein Knoten v mit maximalem Grad mit der ersten Farbe gefärbt. Im Anschluss daran wird für jeden noch nicht gefärbten benachbarten Knoten von v ein *Sättigungsgrad* [Now06] ermittelt. Dieser Sättigungsgrad beinhaltet die Anzahl der unterschiedlichen Farben der bereits gefärbten Nachbarknoten. Mit dieser Anzahl von Farben kann der betreffende Knoten also nicht mehr gefärbt werden. Als nächstes wird der Knoten mit dem höchsten Sättigungsgrad mit der nächsten freien Farbe gefärbt und es werden wiederum neue Sättigungsgrade ermittelt. Die Reihenfolge der noch nicht gefärbten Knoten ergibt sich demnach erst während der laufenden Färbung, so dass in diesem Fall der Algorithmus zur Ermittlung der Sortierreihenfolge mit der Färbung verschränkt ist. Abbildung 2.39 fasst die Algorithmen zur Vorsortierung der zu färbenden Knoten nochmals zusammen.

Strategie zur Sortierreihenfolge	**Algorithmus**	
	Beschreibung	**Laufzeitkomplexität**
zufällig	---	$O(\|V\|)$
Largest First LF	[WP67]	$O(\|V\| \log \|V\|)$
Smallest Last SL	[MMI72]	$O(\|E\| + \|V\|)$
Saturation Largest First DSATUR	[Bre79], [KJ85]	$O(\min(\|V\|^2, \|E\| \log \|V\|))$

Abbildung 2.39: Algorithmen zur Sortierreihenfolge

2.3.3.4 Algorithmen zur Kantenfärbung und -anordnung

Zusätzlich zur Knotenfärbung wird im weiteren Verlauf dieser Arbeit die Kantenfärbung von Multigraphen von Bedeutung sein. Wie schon bei der Knotenfärbung skizziert, sind auch für die Kantenfärbung sequentielle Färbungsalgorithmen weit verbreitet. Einige Heuristiken verfügen sogar über eine sehr gute Approximationsgüte und sind damit in der Praxis verwendbar. Da die Knotenfärbung in die Kantenfärbung übertragbar ist und um-

gekehrt, können viele Algorithmen übertragen werden, wie beispielsweise die Greedy-Färbung. Die Färbungen stellen sie zwei verschiedene Sichtweisen dar, die ein Problem auf unterschiedliche Weise darstellen.

Heuristiken zur Kantenfärbung

Im Rahmen dieser Arbeit hat Nowak [Now06] den zu Greedy-Seq-Color analogen Algorithmus *Greedy-Seq-Edgecolor* untersucht. Der Algorithmus Greedy-Seq-Edgecolor erzeugt zu einem Multigraphen $G = (V, E, g)$ eine Kantenfärbung mit maximal $2 \cdot \Delta(G) - 1$ Farben. Wird die nächste zu färbende Kante in konstanter Zeit ausgewählt, so beträgt die Laufzeit $O(|E| \cdot \Delta(G))$. Dies deutet bereits an, dass die Auswahl der nächsten zu färbenden Kante auf unterschiedliche Weise erfolgen kann. Ebenso wie bei der Knotenfärbung können sowohl statische, als auch dynamische Algorithmen zur Vorsortierung der Reihenfolge der zu färbenden Kanten zum Einsatz kommen, um die Güte der Heuristik zu verbessern. Aus dem Satz von Vizing lässt sich ableiten, dass neben dem Grad der Knoten $\Delta(G)$ der Multigraphen auch die Betrachtung der Multiplizität $\mu(G)$ für eine sinnvolle Vorsortierung der Reihenfolge der Kanten sinnvoll ist.

Eine polynomielle Heuristik zur sequentiellen Kantenfärbung von Multigraphen, welche eine Umfärbung vornimmt, wurde von Nishizeki und Kashiwagi in [NK90] vorgestellt. Diese Heuristik mit sehr guter Approximationsgüte nimmt eine Färbung in $O(|E| \cdot (\Delta(G) + |V|))$ vor. Auch zur Färbung mittels unabhängiger Mengen existiert ein Äquivalent der Kantenfärbung. Dazu wird der Begriff des Matchings benötigt:

Definition 2.3.37: Matching
Sei $G = (V, E, g)$ ein Multigraph. Eine Menge $F \subset E$ heißt Matching von G, wenn keine zwei Kanten aus F adjazent sind.

Nowak [Now06] stellt den zu *Greedy-IS-Color* analogen Algorithmus *Greedy-Match-Edgecolor* vor, der eine heuristische Kantenfärbung in $O(|E^2|)$ vornimmt, sofern das Matching in $O(|V|)$ bestimmt wird.

Exakte Algorithmen zur Kantenfärbung

Wie bereits erwähnt, ist die Klasse der bipartiten Graphen mit $\chi'(G) = \Delta(G)$ kantenfärbbar. Dazu präsentierten Cole, Ost und Schirra [COS01] einen exakten Algorithmus mit einer Laufzeit von $O(|E| \cdot log(\Delta(G)))$. Alon [Alo03] stellte im Jahre 2003 einen einfachereren Algorithmus vor mit einer Laufzeit von $O(|E| \cdot log(|E|))$.

Auf den ersten Blick ungewöhnlich ist der Beweis von Erlebach [Erl98], dass eine exakte Kantenfärbung beliebiger Multigraphen mit beschränkter Knotenzahl in polynomieller Zeit möglich ist. Der Algorithmus *Bounded-Edgecolor* überführt dabei das Kantenfärbungsproblem in ein ILP. Die polynomielle Laufzeit wäre für diese Arbeit von großer Bedeutung, da genau dieser Fall eintritt. Nowak [Now06] beweist die Verallgemeinerung für Multihypergraphen. Praktisch ist dieser Algorithmus jedoch nicht verwendbar aufgrund einer multiplikativen Konstante, die von der Knotenzahl abhängig ist. Bereits bei

32 Knoten - welche im weiteren Verlauf dieser Arbeit der Anzahl der Ports eines Swiches entsprechen - besitzt diese Konstante bereits eine Größe von $8.7869 \cdot 10^{158}$.

2.3.3.5 Generierung von Schedules aus Konfliktgraphen

Abschließend wird in diesem Kapitel am Beispiel der Knotenfärbung gezeigt, wie aus einem gefärbten Graphen eine Schedule ermittelt werden kann. Diese Methodik ist in der Literatur als Konfliktgraph-basiertes Verfahren zur Erstellung einer Schedule bekannt [Kön03]. Dabei seien die Knoten des Graphen Übertragungen, die z. B. im Ansatz von Erlebach [Erl98] als „Calls“ bezeichnet werden, vgl. Kapitel 3.5. In Abbildung 2.40 sind die IRT-Übertragungen mit 1 bis 9 nummeriert und die Farben von F1 bis F4.

Zwei IRT-Übertragungen können nicht gleichzeitig ausgeführt werden, wenn sie beide eine gemeinsame Ressource verwenden. Dies kann ein gemeinsames Kabel in einer Halbduplexverbindung sein, wodurch bei gleichzeitiger Verwendung eine Kollision entsteht. Eine gemeinsame Ressource im Rahmen einer Vollduplexverbindung ist ein Ausgangsport eines Switches, falls beide Übertragungen in die gleiche Richtung weisen. Eine zeitgleiche Verwendung führt hier zu einer Zwischenspeicherung und ggf. einer Verwerfung des Frames. Für den Fall, dass zwei IRT-Übertragungen nicht gleichzeitig ausgeführt werden können, werden ihre Knoten durch eine Kante miteinander verbunden.

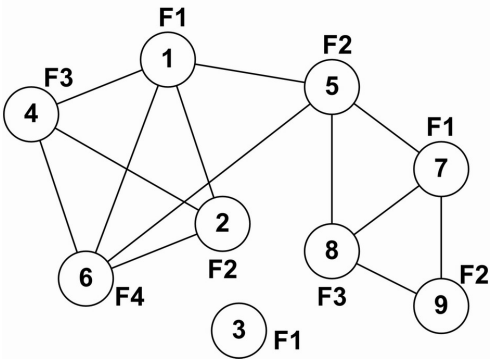


Abbildung 2.40: Gefärbter Konfliktgraph

Ein solcher Graph kann nun mit einem Algorithmus zur Knotenfärbung gefärbt werden, so dass das Problem der Kollisionen bzw. der Pufferung in ein Problem der Graphenfärbung überführt wird. Dies hat zur Folge, dass Knoten, welche durch eine Kante miteinander verbunden sind, in einer gültigen Färbung nicht die gleiche Farbe erhalten. Andererseits sind Knoten mit gleicher Farbe nicht durch eine Kante miteinander verbunden, können also gleichzeitig ausgeführt werden.

Ist der Graph gefärbt, so kann er leicht in eine Schedule überführt werden. Die Nummern der Farben - im Beispiel F1 bis F4 - werden hintereinander gereiht und bilden die Zeitslots. Im Anschluss wird jede IRT-Übertragung dem Zeitslot mit der entsprechenden Farbe zugeordnet. Abbildung 2.41 zeigt die entstandene Schedule aus dem Beispiel.

Aufgrund der Definition einer gültigen Färbung befinden sich in den Zeitslots Fi keine Übertragungen, die miteinander in Konflikt stehen.

Zeitslot 1	Zeitslot 2	Zeitslot 3	Zeitslot 4
Übertragung 1	Übertragung 2	Übertragung 4	Übertragung 6
Übertragung 3	Übertragung 5	Übertragung 8	
Übertragung 7	Übertragung 9		

Abbildung 2.41: Resultierende Schedule

2.3.4 Zusammenfassung

Zu Beginn dieses Kapitels wurde dargelegt, dass der Determinismus in einem Netzwerk durch die Einführung von Zeitslots erreicht werden kann. Die im Bereich der Automatisierungstechnik im Vorfeld bekannten IRT-Übertragungen werden diesen Zeitslots zugeordnet, so dass eine Offline-Schedule entsteht. Im Bereich der Netzwerktechnik wird dieses Verfahren als *TDMA* bezeichnet. Im Anschluss daran wurden Grundlagen und Definitionen aus dem Bereich der Graphentheorie vorgestellt, da sowohl das Netzwerk, als auch die Konflikte der IRT-Übertragungen im folgenden Verlauf der Arbeit als Graphen modelliert werden. Die Idee besteht darin, die Konflikte der IRT-Übertragungen durch die Bildung der Offline-Schedule zeitlich zu entzerren. Zu diesem Zweck werden Konflikt-Graphen erstellt, aus deren Knoten- bzw. Kantenfärbung eine Schedule erzeugt werden kann.

Obwohl die Graphenfärbung NP-hart ist, sind eine Vielzahl von Heuristiken und Abschätzungen bekannt. In diesem Kapitel wurden Algorithmen zur Färbung vorgestellt, die in

- exakte Algorithmen,
- Heuristiken und
- Metaheuristiken

klassifizierbar sind. Zusätzlich existiert eine Klassifizierung nach

- sequentiellen Algorithmen,
- statische und dynamische Algorithmen zur Erstellung einer günstigen Vorsortierung von Kanten oder Knoten für die entsprechende Färbung,
- Algorithmen zur Färbung mittels unabhängiger Mengen sowie
- genetische Algorithmen.

Zum Abschluß dieses Kapitels wurde anhand eines Beispiels skizziert, wie aus einem Graphen durch dessen Färbung eine Schedule erstellt werden kann.

Kapitel 3

Ansätze für echtzeitfähiges Ethernet

In diesem Kapitel werden existierende Ansätze vorgestellt, mit denen Ethernet Echtzeitfähigkeit erreicht. Die Vorstellung der Technologien erfolgt nach einer Klassifizierung von Felser [Fel05b], wobei die Historie und die Idee der jeweiligen Ansätze im Vordergrund stehen mit dem Ziel, Strategien für die Einführung des Echtzeitverhaltens zu ermitteln und zu bewerten. Der Fokus liegt hier auf der Diskussion der Einhaltung des Standards des Ethernet-Protokolls nach IEEE 802.3 einerseits und andererseits dem Erreichen der Kritierien für automatisierte Anlagen. Gerade in den höheren Echtzeitklassen der IAONA ist die Minimierung der Laufzeitverzögerung ebenso von Bedeutung wie die Minimierung des Jitters dieser Verzögerung.

Ein zweiter Fokus liegt darin, die Breite der Ansätze zur Erreichung von Echtzeitfähigkeit zu zeigen und diese zu klassifizieren. Generell werden dabei zwei Zielsetzungen verfolgt. In den härtesten Echtzeitklassen wie der Antriebsregelung wird die Integration der Feldbusse mit Ethernet gefördert, während im Falle von weicheren Echtzeitbedingungen die Feldbusse durch Ethernet ersetzt werden können.

Nach dem Vergleich der Technologien wird die Notwendigkeit eines neuen Ansatzes begründet, dessen Ausarbeitung den Kern dieser Arbeit darstellt.

3.1 Vorstellung existierender Ansätze

Felser [Fel05b] kategorisiert die Einführung der Echtzeitfähigkeit in das 4-Schichten-Modell der Internet-Architektur und in das OSI-Modell in 3 Klassen:

- Klasse 1: Einführung der Echtzeitfähigkeit oberhalb der Transportschicht
- Klasse 2: Einführung der Echtzeitfähigkeit oberhalb der Ethernet-Schicht
- Klasse 3: Einführung der Echtzeitfähigkeit durch Modifikation der Ethernet-Schicht

In der ersten Klasse bleibt der gesamte herkömmliche Protokollstapel erhalten, wodurch die volle Kompatibilität zum herkömmlichen Ethernet bis auf Anwendungsebene gewahrt bleibt. Bekannte Realisierungen der ersten Klasse sind Modbus/TCP [Acr05], P-NET [Pne99], JetSync [Jet07], EtherNet/IP mit CIPsync und der Foundation Fieldbus HSE. Da der Fokus dieser Arbeit auf der Ethernet-Schicht - also auf die von Felser

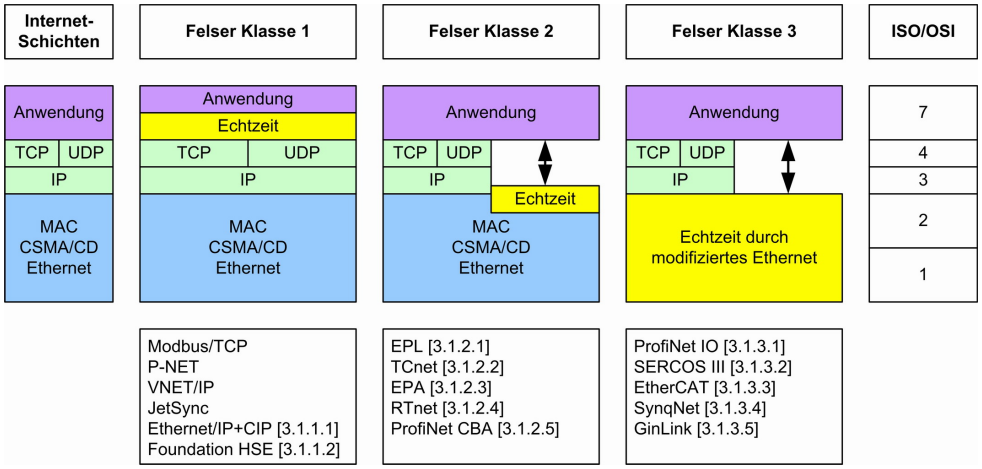


Abbildung 3.1: Übersicht über die Klassen nach Felser [Fel05b]

definierten Klassen 2 und 3 - liegt, werden aus der ersten Klasse nur exemplarisch Ether-Net/IP mit CIPsync sowie der Foundation Fieldbus HSE in den Kapiteln 3.1.1.1 und 3.1.1.2 vorgestellt. Im ersten Fall wird Ethernet unter Beibehaltung der Kompatibilität echtzeitfähig, im zweiten Fall wird die Integration eines vorhandenen Feldbusses in eine Ethernetinfrastruktur vorgestellt.

Die zweite Klasse der echtzeitfähigen Protokolle werden direkt über der Ethernet-Schicht eingesetzt, wodurch die Kompatibilität bis zu OSI-Schicht 2 gewahrt bleibt. Der gewünschte Determinismus wird durch einen Eingriff in die MAC-Schicht erreicht. Die physikalische Schicht bleibt dabei unverändert bestehen. Gleichzeitig bleibt oberhalb der Ethernet-Schicht die Transparenz der Protokolle gewahrt, so dass neben echtzeitkritischen Daten auch TCP/UDP/IP-Daten übermittelt werden können. Zu der zweiten Klasse gehören Ethernet PowerLink EPL, TCnet, EPA, RTnet und ProfiNet SRT (seit Version 2).

Der Ansatz der dritten Klasse liegt in der Ersetzung oder Ergänzung der Ethernet-Schicht inclusive der Medienzugriffssteuerung der Sicherungsschicht. Die Ersetzung hat den Zweck, für die härtesten Echtzeitklassen optimierte Hardware bereit zu stellen. Die Transparenz oberhalb der Sicherungsschicht existiert auch hier. Als Vertreter dieser Klasse werden ProfiNet IRT (seit Version 3), SERCOS III, EtherCAT, SynqNet und GinLink vorgestellt.

Die Liste der vorgestellten Lösungen besitzt nicht den Anspruch auf Vollständigkeit. Ziel ist es vielmehr, die Ansätze, Vorteile und Nachteile der einzelnen Lösungen zu extrahieren. Des Weiteren sollen etablierte Strategien zur Einführung von Determinismus in das bestehende Ethernet analysiert und deren Nähe zum Ethernet-Standard im Gegensatz zu der erreichbaren Echtzeitklasse diskutiert werden. Eine Übersicht über existierende Ansätze können dem IAONA Handbook [LL05], der Veröffentlichung von Larsson [Lar05], von Decotignie [Dec05] sowie der Veröffentlichung von Schnell und Wiedemann [SW06] entnommen werden.

3.1.1 Echtzeitfähigkeit über der Transportschicht

3.1.1.1 EtherNet/IP mit CIPsync

Historie

Ethernet Industrial Protocol (EtherNet/IP) [ODVA07] wurde 1988 ursprünglich von Allen-Bradley als Teil der Rockwell-Automation Inc. entwickelt. Der Ausgangspunkt war die Aufgabe, das bereits auf Anwendungsebene existierende Common Industrial Protocol (CIP) in den TCP/IP-Protokollstapel zu integrieren. Bei CIP handelt es sich um eine objektorientierte Bibliothek für Dienste der Automatisierungstechnik, die als Application Object Library bezeichnet wird [LL05]. Auf dieser Basis können Gerätehersteller, z. B. von Antrieben oder Ventilen, ihre eigenen Objekte und Profile definieren, die dann als abstrakte Repräsentation einer Anlagen-Komponente dient. Diese Komponenten können dann parametriert werden. Im Anschluss daran kann die Kommunikation zwischen den CIP-Objekten modelliert werden. Aufgrund der Standardisierung und der hohen Verbreitung von CIP können Feldbusse wie ControlNet und DeviceNet [SW06] über Bridges in das Netzwerk integriert werden [LL05].

Im Jahre 2000 wurde EtherNet/IP als offener Standard an die Open DeviceNet Vendor Association (ODVA) übergeben. In diesem Jahr wurde dem Protokoll auch das nach IEC 61588 genormte CIP-Sync hinzugefügt, um eine härtere Echtzeitfähigkeit zu ermöglichen. EtherNet/IP ist standardisiert nach IEC 61784-1 in Kommunikationsprofil 2/2.

Infrastruktur

EtherNet/IP ohne CIPsync setzt auf einer herkömmlichen Ethernet-Hardware nach IEEE 802.3 auf und verwendet zur Übertragung von Echtzeitdaten den UDP/IP-Protokollstapel. Zur Übertragung von Konfigurations- und Diagnosedaten wird TCP/IP verwendet. Der Ethernet-Frame besitzt als EtherType 0x0800, da es sich bei dem darüberliegenden Protokoll um IP handelt. In dieser ursprünglichen Version stellt EtherNet/IP also eine reine Software-Lösung bereit, die ein $100 Mbit/s$-Ethernet unter Verwendung von Switches im Vollduplexbetrieb und Baumtopologie verwendet. Die Klasse von Lösungen ohne Verwendung eigener Hardware wird als *Commercial off the Shelf* (COTS) bezeichnet. Auch der Einsatz von Hubs mit CSMA/CD-Verhalten ist prinzipiell möglich, wird jedoch nicht empfohlen, da in diesem Falle keine Echtzeitfähigkeit erreicht werden kann.

Jedes Gerät meldet den Bedarf für den Empfang von bestimmten Daten an und stellt seine Daten, beispielsweise die Istposition eines Motors, zum Senden bereit. EtherNet/IP arbeitet also nach dem Publisher/Subscriber-Modell und unterstützt neben Unicast- und Broadcast-Nachrichten auch Multicasting über IP [LL05].

Erreichen der Echtzeitfähigkeit

Die Switches können jedoch Pakete im Falle der Überlast verwerfen. Um dies zu vermeiden, verwendet EtherNet/IP eine Priorisierung durch das Einfügen von VLAN-Tags nach IEEE 802.1p. Dabei entstehen acht Prioritäts-Stufen, bei denen EtherNet/IP-Frames die höchste Priorität zugewiesen wird. Die korrekte Interpretation der Tags obliegt dabei

den Switches. Dies wird dann problematisch, wenn andere Applikationen ebenfalls die Priorisierung mittels VLAN-Tagging verwenden, beispielsweise eine Videokonferenz über einen Teil des Netzwerkes übertragen wird.

Aufgrund des fehlenden Determinismus und der Reaktionszeiten, die im besten Fall im Millisekunden-Bereich liegen, ist EtherNet/IP nicht für Aufgaben der Antriebstechnik geeignet. Die Reaktionszeiten sind zum einen begründet mit dem großen Protokoll-Overhead, der über das Schichtenmodell interpretiert werden muss. Die Abarbeitung eines UDP/IP-Protokollstapels benötigt bei einem Intel Pentium $166MHz$ - dessen Rechenleistung sich in jedem Gerät befinden muss - zwischen $400\mu s$ und $500\mu s$ [SW06].

Zusätzlich dazu sorgt die Verzögerung in Standard-Switches, die ggf. mit der store-and-forward Technik arbeiten, für eine erhöhte Verzögerungszeit, die durch die Priorisierung der Pakete verringert werden soll. Andererseits ist die hohe Kompatibilität zum Ethernet-Standard zu nennen. Die CIP-Dienste der Automatisierungstechnik befinden sich ausschließlich auf der Anwendungsschicht.

Durch das Hinzufügen von CIP-Sync, welches auf der Synchronisation nach IEEE 1588 basiert, soll das Echtzeitverhalten verbessert werden. Dazu muss jedes Gerät mit einem 1588-Synchronisationsbaustein [Zin07] ausgerüstet sein, um exakte Abtastzeitpunkte einzuhalten. Aufgrund der hohen Verbreitung dieses standardisierten Synchronisationsverfahrens können die Bausteine in hohen Stückzahlen preisgünstig gefertigt werden. Sie messen den tatsächlichen Sendezeitpunkt eines Frames kurz vor seiner Ausgabe auf das Übertragungsmedium, so dass sie, wie in Abbildung 3.2 skizziert, idealerweise auf der Sicherungsschicht implementiert werden. Dieser Sendezeitpunkt wird als Zeitstempel in den Frame integriert. EtherNet/IP nennt dieses Verfahren „Time Synchronized Distributed Control“ [LL05]. Der Jitter liegt dabei im Bereich von $100ns$ [ODVA07].

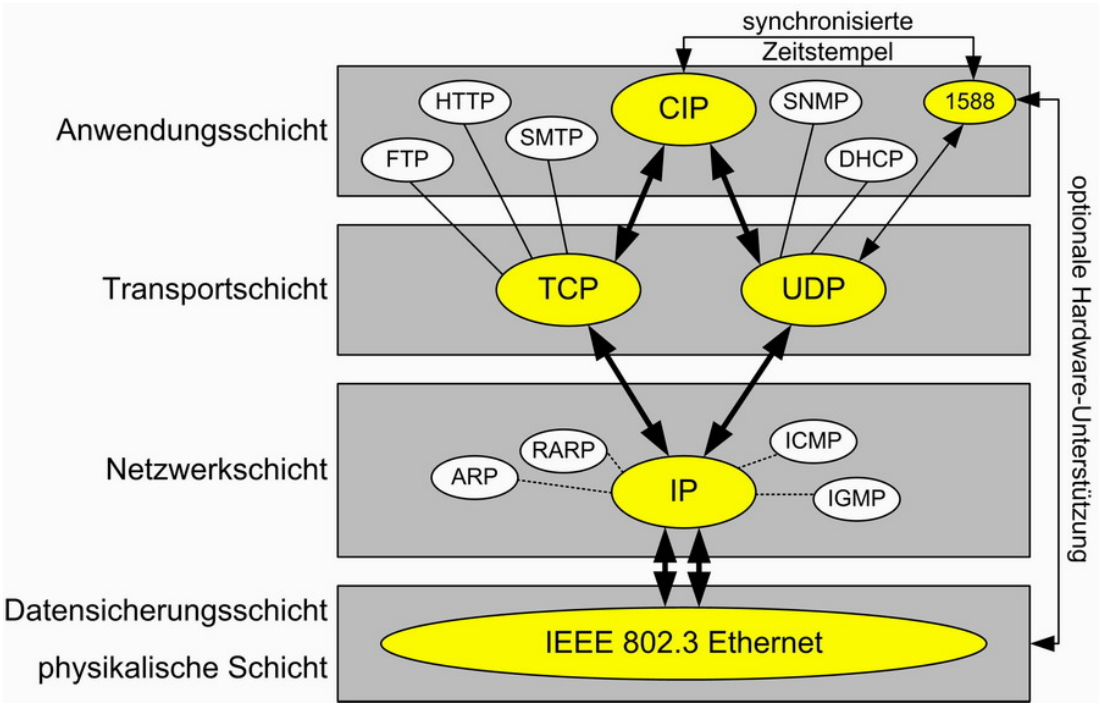


Abbildung 3.2: Protokollhierarchie von EtherNet/IP [LL05]

Auch wenn der exakte Zeitpunkt des Versendens mit einem sehr geringen Jitter bekannt ist, ist die Verzögerungszeit des Frames von den Switches und deren aktueller Last abhängig. Des Weiteren wird durch die Einführung der Synchronisation nach IEEE 1588 der Protokoll-Overhead durch die Verwendung von UDP/IP für die Übertragung von Echtzeitdaten nicht verringert. Das UDP-Protokoll besitzt einen Overhead von $8Byte$ im Header, das IP-Protokoll zusätzlich $20Byte$. Der Ethernet-Header erreicht mit Präambel und VLAN-Tagging einen Overhead von $30Byte$, so dass jeder EtherNet/IP-Frame einen Overhead von $58Byte$ besitzt. Die ist für kurze Zykluszeiten der Antriebstechnik nicht vertretbar. Durch das Auslesen der Zeitstempel kann ein Empfänger zwar den exakten Sendezeitpunkt ermitteln. Trifft der Frame aufgrund von Zwischenspeicherung in mehreren überlasteten Switches jedoch zu spät ein, ist die Information zum Empfangszeitpunkt bereits wertlos.

Über das IP-Multicasting wird andererseits ein Querverkehr ermöglicht, so dass die folgende Topologie zur Ansteuerung von 4 Achsen bei einer Master-Achse empfehlenswert ist:

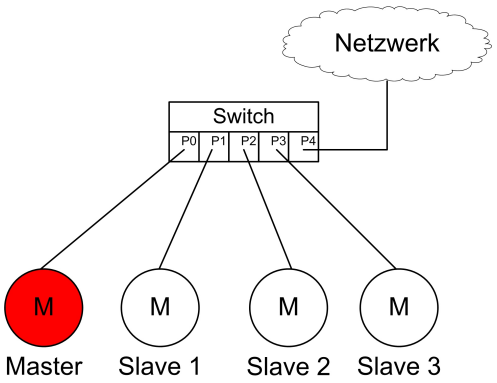


Abbildung 3.3: Antriebsregelung mit EtherNet/IP

Der Switch soll dabei lediglich mit Frames der Master-Achse belastet werden, die über einen kurzen Weg - also nicht über eine variable Zahl an weiteren Switches - mit den anderen Achsen verbunden ist. Der Uplink-Port des Switches ist mit anderen Anlagenteilen oder mit einem Büro-Netzwerk verbunden. Auf diese Weise wird die maximale Echtzeitfähigkeit von EtherNet/IP erreicht unter Verwendung von Standard-Hardware.

3.1.1.2 Foundation Fieldbus HSE

Historie

Im Gegensatz zu EtherNet/IP verfolgt die Fieldbus Foundation [Fie07] [VP06] einen völlig anderen Weg der Integration von Ethernet in die Feldebene. Bei der Foundation handelt es sich um einen Zusammenschluss von insgesamt 350 Firmen der WorldFIP North America und dem Interoperable Systems Project (ISP) zu einer non-profit Organisation. Ausgangspunkt ist der 1995 vorgestellte *Foundation Fieldbus H1* mit einer Übertragungsrate von 31.25 kBit/s und identischer Busphysik wie der ProfiBus PA (Prozess-Automation) gemäß IEC 61158-2 [Mah03]. Da die Übertragungsrate sehr gering ist, wurde für die

Kommunikation auf der Leitebene zunächst ein schnellerer Bus H2 in Erwägung gezogen. Aufgrund der hohen Verbreitung von Ethernet in diesem Bereich wurde die Entwicklung jedoch frühzeitig eingestellt und mit der Entwicklung des *Foundation Fieldbus High Speed Ethernet* (HSE) [SW06] begonnen. Nach dem Konzept der Fieldbus Foundation bleibt die Trennung zwischen Feldebene und Prozessleitebene in der Pyramide der Automatisierungstechnik erhalten.

Infrastruktur

Bei dem H1-Feldbus kann eine Mischung aus Baum- und Bustopologie zum Einsatz kommen [Gru01]. Die Kommunikation erfolgt mittels eines Master/Slave-Zugriff oder durch die Verwendung eines deterministischen Token-Passing Verfahrens. In der Spezifikation 1.2 können sich maximal 32 Geräte in einem H1-Subnetz befinden, in dem keine nicht-deterministische Kommunikation erlaubt ist.

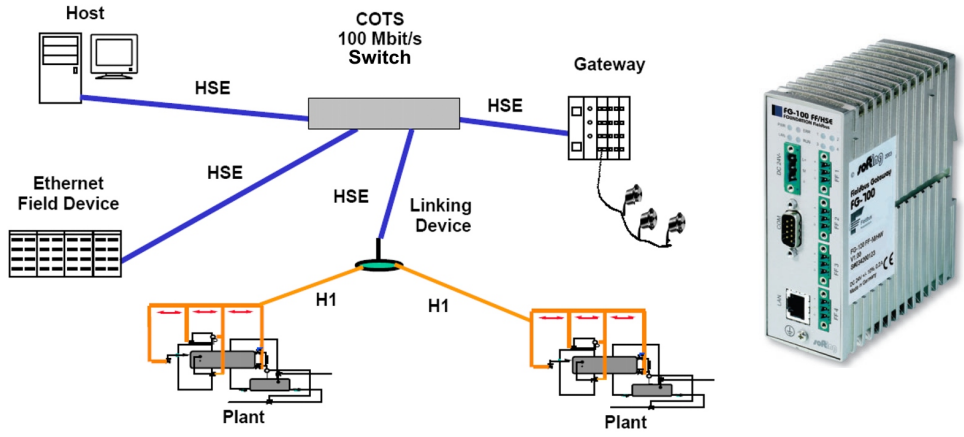


Abbildung 3.4: Foundation Netzwerk und Linking Device[Mey04]

Über eine Bridge bindet ein Linking-Device mehrere H1-Subnetze zu einem HSE-Netzwerk zusammen, bei dem herkömmliche $100Mbit/s$-Switches eingesetzt werden. Das FG-100 FF/HSE von Softing [Sof05] verbindet beispielsweise bis zu vier H1-Subnetze mit einem HSE-Uplink.

Erreichen der Echtzeitfähigkeit

HSE ist selbst nicht echtzeitfähig, da es weiterhin auf Standard-Ethernet mit übergelagerten TCP/UDP/IP-Protokollstapels aufsetzt [Par02]. Die Entwicklung eines weiteren echtzeitfähigen Netzwerks war jedoch auch nicht im Fokus der Fieldbus Foundation.

Auf der Anwendungsebene des H1-Feldbusses existiere bereits, ähnlich wie bei EtherNet/IP, ein Funktionsblock-Modell zur Verwaltung von wiederverwendbaren Hardware- und Sowftware-Komponenten der automatisierten Anlage. Der Funktionsblock ist nach IEC 61131 genormt und interagiert mit anderen Funktionsblöcken über Ein-/Ausgangsvariablen. Auch hier existieren Gateways zu anderen Feldbussen, die als Foreign I/O-Gateways bezeichnet werden.

Das Ziel der Fieldbus Foundation war es, dieses Funktionsblock-Modell auf die Ebene des HSE zu übertragen und das gleiche Objektmodell zu verwenden. Auf diese Weise erscheint die Bridge zwischen den beiden Bussen transparent. Der Anwender auf der Ethernet-Seite erhält also den Eindruck, direkt und gleichermaßen auf alle H1-Geräte zugreifen zu können. Mehrere H1-Busse können über die HSE-Bridges zeitunkritische Management-, Diagnose- und Konfigurationsdaten untereinander austauschen.

3.1.2 Echtzeitfähigkeit oberhalb der Ethernet-Schicht

3.1.2.1 Ethernet PowerLink (EPL)

Historie

Ethernet PowerLink (EPL) wurde 2001 von der österreichischen Firma Bernecker und Rainer [BR07] vorgestellt, befindet sich aktuell in der zweiten Version und wird von der Nutzergruppe Ethernet PowerLink Standardization Group (EPSG) mit Sitz in Winterthur in der Schweiz verwaltet. Der EPL-Standard wurde in die IEC zur Normung eingebracht und als Teil der IEC 61158 akzeptiert. EPL setzt direkt über der Medienzugriffssteuerung der Sicherungsschicht auf. EPL ist die erste ethernet-basierte Technologie, die bereits über einen langen Zeitraum im Feldbereich erprobt ist. Nach Angaben von Bernecker und Rainer waren im Jahr 2004 über 15.000 EPL-Geräte bei mehr als 100 Kunden im Einsatz [PH06].

Infrastruktur

Die Infrastruktur eines EPL-Netzes [SV07] besteht ausschließlich aus preisgünstigen Hubs, die in einer Baumtopologie angeordnet sind. Eine Linientopologie kann unter Verwendung besonderer 3-Port-Hubs, welche in die Geräte integriert sind, emuliert werden. Maximal zehn Hubs können in einer Linie hintereinander verschaltet werden. Aufgrund der Verwendung von Hubs wird die Latenzzeit bei weitem nicht so stark kaskadiert wie beim Einsatz von Switches. Das EPL-Protokoll basiert auf dem IEEE 802.3 Standard und verwendet auf der Bitübertragungsschicht 100 BASE-TX mit Standard-Ethernet NICs. Es existiert zu jedem Zeitpunkt nur genau ein Sender im Netzwerk.

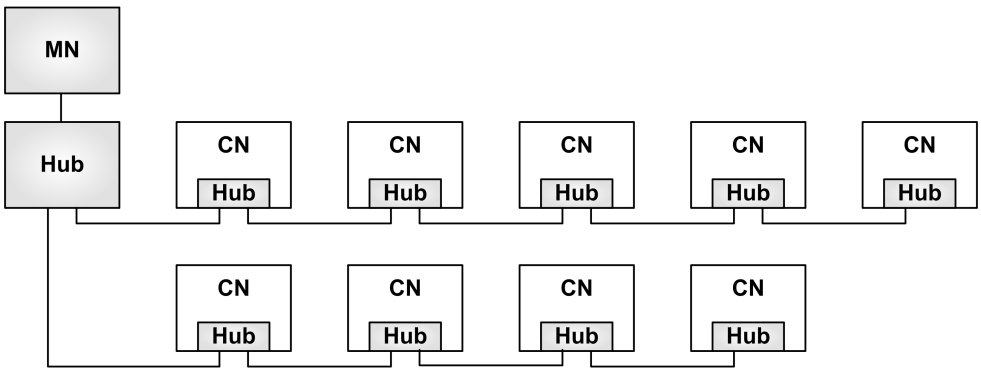


Abbildung 3.5: EPL Infrastruktur [LL05]

Jedes Gerät verfügt über eine MAC-Adresse sowie über eine $8Bit$ Node-ID. Die Node-ID adressiert die Geräte nochmals in den EPL-Headern und ist gleichzeitig der Host-Anteil der IP-Adresse, falls oberhalb des OSI-Schicht 2 das IP-Protokoll ausgeführt wird.

EPL arbeitet nach einem strengen Master/Slave-Zugriffsverfahren, bei dem ein zentrales Master-Gerät jeweils ein Subnetz verwaltet. Dieses Gerät wird als *Managing Node* (MN) bezeichnet und besitzt auch eine Gateway-Funktionalität, um das Echtzeit-Subnetz vom nicht-deterministischen Ethernet abzugrenzen. Aufgrund der Verwendung von Hubs, die ein eintreffendes Signal stets an alle Ausgangsports weiterleiten, muss jedes Gerät im Subnetz dem EPL-Protokoll und damit dem Master folgen. Ein asynchron sendendes Gerät würde die Echtzeitfähigkeit unmittelbar zerstören. Ein EPL-Slave wird auch als *Controlled Node* (CN) bezeichnet.

Erreichen der Echtzeitfähigkeit

Das Ziel bei der Entwicklung von EPL lag darin, ein deterministisches Netzwerk auf Basis von IEEE 802.3 zu schaffen mit einem maximalen Jitter $< 1\mu s$ und Netzwerkzykluszeiten unter $500\mu s$ [SW06], so dass die härteste Echtzeitklasse nach IAONA erfüllt wird. Der Modus dieser harten Echtzeit wird als Protected Mode bezeichnet, der ausschließlich EPL-Geräte im Subnetz zulässt. Das CSMA/CD-Verfahren der Sicherungsschicht bleibt auch im Protected Mode erhalten und wird durch ein übergeordnetes Scheduling ergänzt, das zentral vom MN verwaltet wird. Da sich genau ein MN im Subnetz befindet, entfällt die Synchronisation mehrerer Schedules. Das Slot Communication Network Management (SCNM) unterteilt die Kommunikation in eine Start-, isochrone und asynchrone Phase, wie in Abbildung 3.6 dargestellt ist. Die Länge der isochronen und asynchronen Phase ist ebenso konfigurierbar wie die Framelängen.

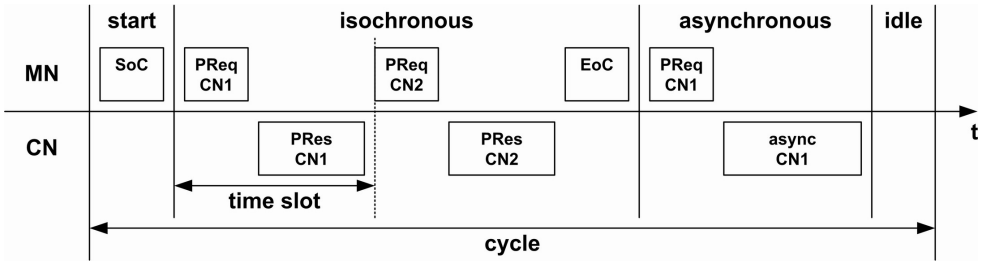


Abbildung 3.6: EPL Zyklus, vgl. [LL05]

In der Start-Phase zu Beginn jedes Zyklus synchronisieren sich die CN mit der Uhr des MN, indem dieser eine *Start-of-Cycle* (SoC) Nachricht als Broadcast versendet. Alle Geräte müssen in der Lage sein, diese Nachricht zu empfangen.

In der isochronen Phase teilt der MN den CN nacheinander durch einzelnes Polling mit Anforderungstelegrammen Zeit zu, in denen der jeweilige CN einen Frame mit der in der Konfigurationsphase definierten Größe senden kann. Der Poll Request zum nächsten CN wird dann mit etwas Sicherheitsabstand gesendet. Die isochrone Phase wird durch eine *End-of-Cycle* Nachricht (EoC) vom MN beendet. Es ist jedoch nicht notwendig, dass jeder CN in jedem Zyklus angesprochen wird, da nicht jedes Gerät gleich wichtig ist und die gleiche Bandbreite benötigt. Zu diesem Zweck wurde eine *Multiplex-Betriebsart*

eingeführt, die zwischen obligatorischen Sendungen in jedem Slot - z. B. zur Ansteuerung von Master-Achsen - und gemeinsamen Slots für mehrere Geräte - z. B. für Slave-Achsen - unterschiedet. Im zweiten Fall fordert der MN die Geräte nur in jedem n-ten Zyklus zum Senden auf. Im Beispiel der Abbildung 3.7 sind die Sendungen der Geräte 1 bis 3 obligatorisch; sie gehören zur Geräteklasse 1, die als *cyclic* bezeichnet wird. Die Geräte 4 bis 11 gehören zur Geräteklasse 2, die als *prescaled* bezeichnet werden. Sie werden nur in jedem vierten Slot angesprochen. Auf diese Weise werden die Slots aufgefüllt, die zur Verfügung stehende Bandbreite effizienter genutzt und letztlich auch die Zykluszeit minimiert.

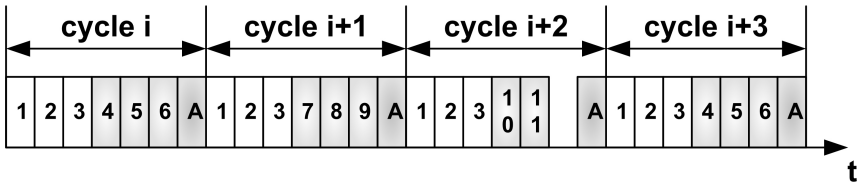


Abbildung 3.7: EPL Zyklen [LL05]

Der angesprochene CN antwortet auf den Poll Request mit einem Poll Response Frame. Da alle anderen Geräte dieses Subnetzes alle Daten mithören können, bietet EPL eine Producer/Consumer-Beziehung. Die anderen CN können durch den Querverkehr die beispielsweise von der Master-Achse gesendeten Istwerte unmittelbar verarbeiten.

Die asynchrone Übertragung wird „*im Anschluss an den isochronen Teil im so genannten asynchronen Zeitslot behandelt. Wünscht ein CN den asynchronen Kanal zu nutzen, teilt er dies dem Managing Node im isochronen Telegramm mit. Der Managing Node ruft dann dieses Gerät im asynchronen Zeitslots mit einem «Invite»-Telegramm ein weiteres Mal auf. Falls nach dem asynchronen Kanal noch etwas Zeit übrig bleibt, wird bis zum Beginn des nächsten Zyklus gewartet.*“ [Sch04] Diese Restzeit bis zum Beginn des nächsten Zyklus wird als Idle-Time bezeichnet.

Auf diese Weise wird sicher gestellt, dass die nächsten SoC-Nachrichten wieder zu periodisch exakten Zeitpunkten gesendet und die Anforderungen an eine Antriebsregelung erfüllt werden. Mit der Konfiguration des Netzwerks wird die *Maximum Transfer Unit* (MTU) für den asynchronen Slot festgelegt. Wird oberhalb der Ethernet-Schicht IP verwendet, so fragmentiert es die Frames falls notwendig selbst. In EPL Version 2 kann ein CN dem MN auch die Anzahl der anstehenden Frames mitteilen, wodurch ein effizienteres Online-Scheduling des asynchronen Slots ermöglicht wird.

Initiiert wird die asynchrone Übertragung durch den Master, indem er die EoC-Nachricht mit der Invite-Nachricht zu einer einzigen *Start-of-Asynchronous* Nachricht (SoA) bündelt und somit die Sendeberechtigung für einen CN erteilt. Normalerweise wird die Invite-Nachricht in einer der folgenden asynchronen Phasen an den anfragenden CN gesendet. Ist noch ausreichend Restzeit in dieser asynchronen Phase vorhanden, so kann das Invite auch direkt im Anschluss an die Anmeldung erfolgen. Der CN kann nach Empfang der Invite-Nachricht seine Daten unter Verwendung von Standard-Protokollen wie IP und TCP/UDP versenden. Seit der Version 2.0 sind alle EPL-Geräte TCP/IP-fähig. Da sich alle Geräte dem EPL-Protokoll unterwerfen, wird sichergestellt, dass die asynchronen Sendungen rechtzeitig vor Beginn des neuen Zyklus beendet sind.

Das EPL-Protokoll

Das EPL-Protokoll ist gekapselt in einen Ethernet-Frame mit $20Byte$ Overhead. Allen EPL-Frames gemeinsam ist ein $3Byte$ großer EPL-Header, der den Typ der EPL-Nachricht sowie die jeweils ein Byte großen Node-IDs des Empfängers und des Senders enthält. Je nach Typ der EPL-Nachricht können Kontroll- oder Status-Bits sowie Service-IDs zum EPL-Header hinzukommen. Der geamte EPL-Header ist damit maximal $6Byte$ lang.

Im SoC-Frame wird zusätlich die Net Time, also die Zeit des Masters, in einem $8Byte$ großen Feld übertragen. Der Datenteil des Poll Response besteht aus *Process Data Objects* (PDOs), die auf der Anwendungsschicht von EPL interpretiert werden. Der Header der gesamten PDOs ist $4Byte$ groß, die PDOs selbst besitzen keinen zusätzlichen Overhead [LL05].

Der Austausch von Parametern oder weniger echtzeitkritischen Daten erfolgt in der asynchronen Phase über *Service Data Objects* (SDOs), die unter Verwendung des UDP/IP-Protokollstapels versendet werden. Dadurch werden die Geräte über das Gateway des MN erreichbar, der die SDOs als Antworten auf Anfragen aus anderen Netzwerk-Segmenten nach außen weiter gibt. Bei dem Kommunikationsmodell handelt es sich hier um ein Client/Server-Verfahren, bei dem jeder CN als Server für Anfragen von außen fungiert. Der MN kann als Proxy gesehen werden.

Berechnung der EPL-Schedule

Nach der Konfiguration der vorgesehenen Zeit für die asynchrone Phase werden die anfallenden zyklischen Übertragungen der isochronen Phase hintereinander sortiert. Im Anschluss daran können die Übertragungen, welche sich Zeitslots teilen, eingefügt werden. Da nur eine Paketgröße existiert und keine überlappenden Übertragungen im Subnetz erlaubt sind, ist dies algorithmisch einfach. Die Zeit für den SoC-Frame addiert mit den Zeiten der isochronen Phase - also der Übertragungsdauer der jeweiligen Poll Requests und Poll Responses - , dem SoA-Frame sowie der asynchronen Phase ergibt die Zykluszeit. Schnell und Wiedemann [SW06] geben die folgende Abhängigkeit der Zykluszeit von der Anzahl der Geräte an, wobei stets $80Byte$ Nutzdaten bidirektional übertragen werden und jeweils 50 m Kabel zwischen den Geräten liegt, bei denen es sich ausschließlich um Geräte der Klasse 1 handelt:

2	Geräte:	$200\mu s$
12	Geräte:	$500\mu s$
30	Geräte:	$1000\mu s$
66	Geräte:	$2000\mu s$
102	Geräte:	$3000\mu s$

Ein Testaufbau mit 50 I/O-Geräten, die jeweils $< 46Byte$ Nutzdaten versenden bzw. empfangen sowie 50 Antriebe mit jeweils ca. $80Byte$ bidirektionalen Nutzdaten hat eine reale Zykluszeit von $2.4ms$ ergeben bei einem Jitter $< 1\mu s$ [SW06].

Protected Mode und Open Mode

Die bislang beschriebene Architektur beschreibt den *Protected Mode* von EPL, der die härteste IAONA-Echtzeitklasse erfüllt und bereits seit der ersten Version von EPL verfügbar ist. Der Protected Mode ist dann aktiv, wenn ausschließlich EPL-Geräte im Subnetz vorhanden sind. Wird im Protected Mode ein nicht PowerLink-fähiges Gerät angeschlossen, so soll die Umschaltung in den Open Mode automatisch erfolgen. Aus diesem Grunde wird kritisiert, dass EPL im Protected Mode nicht standardkonform arbeitet.

Im *Open Mode* werden sowohl Standard Ethernet-Frames, als auch Echtzeit-Frames versendet. Im Gegensatz zu ProfiNet und EtherCAT wird jedoch bei EPL auf den Einsatz von modifizierter Hardware verzichtet und lediglich ein IEEE 1588 Zeitstempel unter Verwendung herkömmlicher Hubs eingeführt. Dadurch fällt EPL im Open Mode in die Echtzeitklasse 3 zurück [Doy04]. In realen Anlagen führt diese Veränderung im laufenden Betrieb zu einem Verlust der Echtzeitfähigkeit, so dass eine dynamische Umschaltung zwischen Open und Protected Mode nicht vorgesehen ist. Die IRT-Übertragungen werden im Protected Mode in Subnetzen gekapselt, während diese Subnetze untereinander im Open Mode kommunizieren können. Der Open Mode soll in der dritten Version von EPL vollständig spezifiziert sein [PH06].

3.1.2.2 Time-critical Control Network (TCnet)

Historie

Bei *TCnet* handelt es sich um einen Ansatz von Toshiba [Tos06], der sich stark an EPL anlehnt und bei dem ähnliche technische Daten erwartet werden. Es wird von der japanischen Regierung gefördert und befindet sich zur Zeit in der Standardisierungsphase der IEC in der Arbeitsgruppe IEC/TC65/SC65C. Das Kommunikationsprofil von TCnet ist in der *Communications Profile Family* 11 (CPF11) der IEC 61784-2 hinterlegt. Das System ist noch wenig verbreitet, obwohl erste Geräte bereits verfügbar sind.

Infrastruktur

TCnet basiert ebenso wie EPL im Protected Mode auf einem geschlossenen Subnetz mit Baumtopologie. TCnet bietet die Möglichkeit der Redundanz, indem auf physikalisch getrennten Lichtwellenleitern übertragen wird. Dazu werden spezielle optische Hubs verwendet, die an den Endpunkten des Netzwerkes über herkömmliche Twisted-Pair Verkabelung an Standard-Ethernet angebunden werden können. Die Verbindung zu einem Büro-Netzwerk wird über eine Bridge Unit hergestellt, die gleichzeitig der zentrale Master des Netzwerkes ist.

Erreichen der Echtzeitfähigkeit

Wie auch EPL erreicht TCnet die Echtzeitfähigkeit durch Scheduling der Übertragungen, dessen Berechnungsvorschrift bei der Recherche im Rahmen dieser Arbeit nicht ermittelt werden konnte. Die Online-Schedule wird wie bei EPL von dem zentralen Master verwaltet. Toshiba nennt seinen Weg zur Vermeidung von Kollisionen innerhalb des Stan-

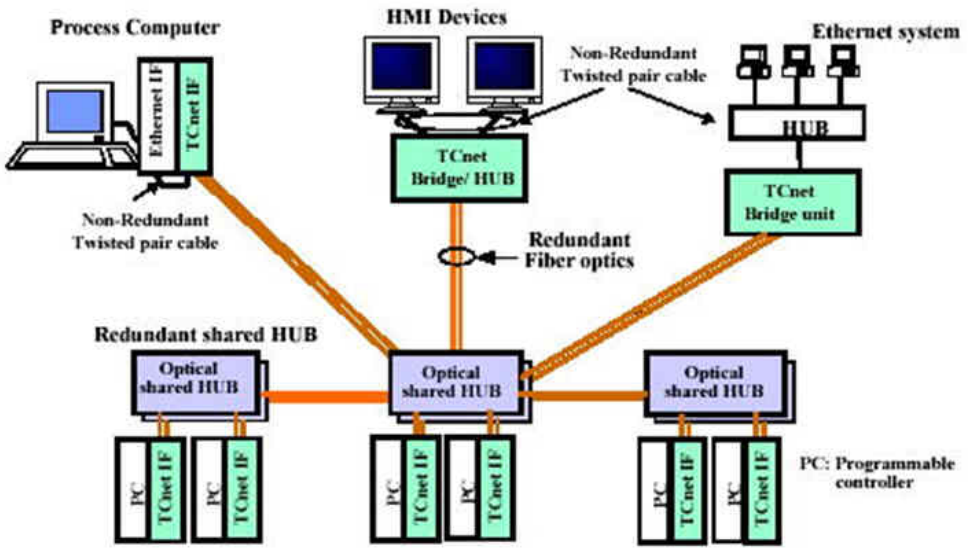


Abbildung 3.8: TCnet Infrastruktur [Tos06]

dard-Ethernets *Deterministic Ordered Multiple Access* (DOMA) [Tos06]. Frames, die aus dem geschützten Subnetz nach außen übertragen werden, passieren die Bridge Unit unmittelbar. Von außen eintreffende Frames in das Subnetz werden über die Schedule weiter geleitet.

Als Unterschied zu EPL ist zu nennen, dass der Datenverkehr in vier Prioritätsklassen unterteilt ist. Liegen Daten unterschiedlicher Priorität zur Sendung an, so werden hochpriore Daten von DOMA bevorzugt.

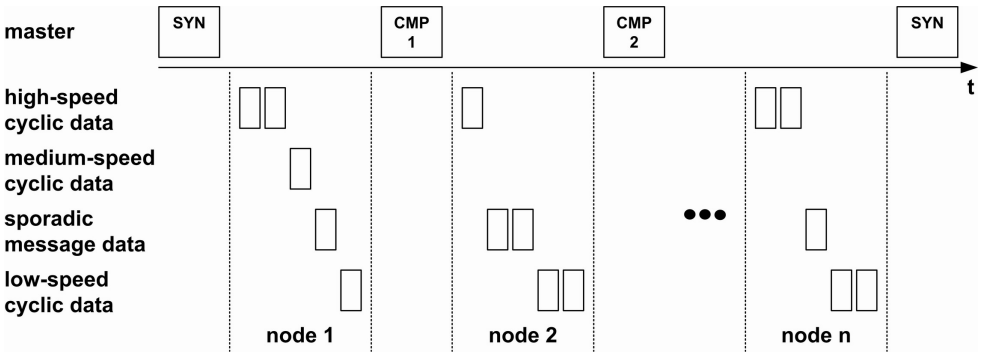


Abbildung 3.9: TCnet-Zyklen [Tos06]

Zur Synchronisation und als Hinweis an alle Geräte, dass ein neuer Zyklus startet, versendet der Master bei TCnet einen SYN-Frame. Nach dem Empfang des SYN-Frames beginnt das erste Gerät mit dem Senden seiner Daten, wie es während der Systemkonfiguration geplant war [Fel05b]. Nach der Beendigung der Sendung wird mittels Broadcast eine *Completed Message* Nachricht (CMP) versendet, wodurch das zweite Gerät mit seiner Sendung beginnt. Ist ein Gerät an der Reihe, so hat es aufgrund der Systemkonfiguration auf jeden Fall das Recht, mindestens eine Nachricht höchster Priorität zu senden. Dies

entspricht einem Token-Passing Verfahren.

Das TCnet-Protokoll

Das Protokoll eines TCnet-Frames ist noch einfacher gestaltet als bei EPL. Die Gesamtframegröße inclusive Ethernet-Overhead liegt zwischen $72Byte$ und $160Byte$, wozu noch der Interframe-Gap addiert werden muss. Der TCnet-Header umfasst lediglich $2Byte$. Daneben können Standard-Frames des TCP/IP- oder UDP/IP-Protokollstapels mit einer Größe zwischen $72Byte$ und $1526Byte$[1] mit niedrigster Priorität übertragen werden.

Das Modell des gemeinsamen Speichers

Ein weiterer Unterschied zu EPL liegt in der Idee des gemeinsamen Speichers. Die Daten aller Geräte liegen in einem virtuellen gemeinsamen Speicher, der sich über das Netzwerk verteilt. Der verteilte Speicher ist in Blöcke unterschiedlicher Größe aufgeteilt, wobei stets genau ein Gerät das Schreibrecht auf einen Block besitzt. Die lokale Kopie des Speichers auf einem Gerät wird durch die Broadcastsendungen aufgrund der Verwendung von Hubs zyklisch aktualisiert; ein einzelnes Polling entfällt. Die Aktualisierungsrate reicht für die ersten drei Prioritätsklassen von $1ms$ aufwärts in $0.1ms$-Schritten. Jedes Gerät muss jedoch nicht den vollständigen Datenstamm halten. Bei der Systemkonfiguration wird eingestellt, welches Gerät welche Daten durch Verwendung des Publisher/Subscriber-Modells in seinem lokalen Speicher hält. Danach werden Punkt-zu-Mehrpunkt Verbindungen zur Datenaktualisierung etabliert. Physikalisch arbeiten die Publisher dennoch aufgrund der Verwendung von Hubs mit Broadcastsendungen.

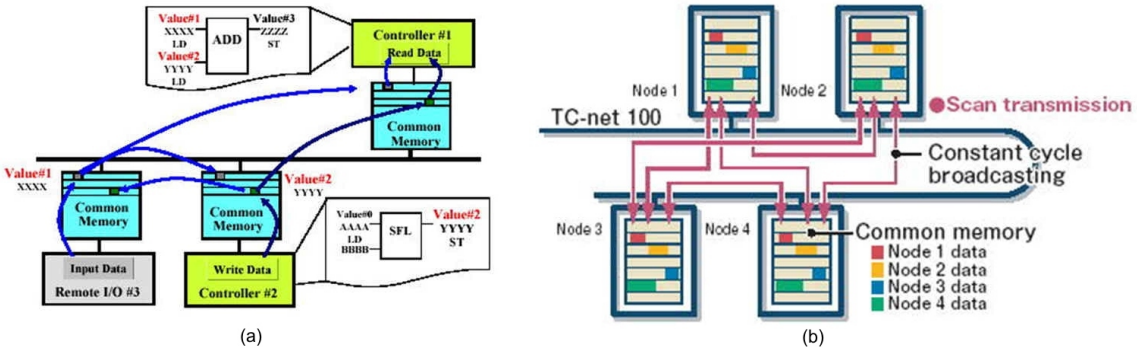


Abbildung 3.10: Konzept des verteilten Speichers bei TCnet [Tos06]

Abbildung 3.10b illustriert das zyklische Update der Variablen im gemeinsamen Speicher durch das Netzwerk. Hat ein Gerät die Daten abonniert, so verwendet es die übertragenen Daten, die im verteilten Speicher abgelegt sind. Abbildung 3.10a zeigt die Datenverteilung anhand eines Beispiels. Controller 2 schreibt den Wert Value 2 exklusiv in den gemeinsamen Speicher. Bei dem nächsten Update wird dieser Wert im Speicher von Controller 1 und Remote I/O 3 aktualisiert. Controller 1 liest diesen Wert und noch einen

[1] Sofern das Scheduling der DOMA dies mit der Zykluszeit der konkreten echtzeitkritischen Anwendung zulässt.

weiteren Wert, der von Remote I/O 3 stammt, um eine Addition durchzuführen, die als Value 3 im lokalen Speicher von Controller 1 abgelegt wird. Die zyklischen Aktualisierungen sind aufgrund des Modells des gemeinsamen Speichers des gesamten Systems für den Programmierer der Anlage transparent.

3.1.2.3 Ethernet for Plant Automation (EPA)

Historie

EPA ist ein weiterer Ansatz, der Ethernet PowerLink sehr ähnlich ist und von der chinesischen Firma Zhejiang Supcon zum Zwecke der deterministischen Ethernet-Kommunikation in der Automatisierungstechnik entwickelt wurde [Fel05b]. Bislang existiert keine Nutzergruppe um die bislang nur in China verbreitete Technologie, die in der CPF14 der IEC 61784-2 beschrieben ist.

Infrastruktur

Die Netzwerktopologie besteht hier aus einem Hub, der eine große Anzahl von Ports aufweist und nicht mit anderen Geräten kaskadiert werden darf. Aufgrund der Hubs kann auch hier nur eine Kommunikation pro Zeiteinheit statt finden. Bislang wurden im Gegensatz zu EPL nur kleine Subnetze realisiert, die sich durch die Verwendung von Bridges an das herkömmliche Ethernet ankoppeln lassen. Die Verwaltung der Zeitscheibe wird als *EPA Communication Scheduling Management Entity* (ECSME) bezeichnet, die sich in der Bridge befindet. Diese Bridge agiert somit als zentraler Master für das Subnetz.

Erreichen der Echtzeitfähigkeit

Während in den ersten Versionen ohne Synchronisation Zykluszeiten von $10ms$ bis $100ms$ möglich waren, wird dies durch die Verwendung des IEEE 1588 Standards verbessert. Die Zykluszeit wird nun in einer festen Anzahl von Millisekunden vorgegeben, während der Jitter durch die Art der Implementierung des IEEE 1588 Protokolls - durch Software oder durch Hardware - bestimmt wird [Dec06].

Auch hier wird ein Zeitscheibenverfahren oberhalb der Medienzugriffskontrolle der Sicherungsschicht eingesetzt und die Kommunikation in Zyklen, die als Makrozyklen T_i bezeichnet werden, aufgeteilt. Auf Applikationsebene werden Funktionsblöcke eingesetzt, wobei ein Block einen Algorithmus mit eigenem statischen Speicher beinhaltet und ausschließlich über Ein-/Ausgabevariablen angesprochen wird. Die Zeitscheibe ist aufgeteilt in einen periodischen Teil T_p und einen nicht-periodischen Teil T_n der Datenübertragung, in dem Standard-Protokolle des TCP/IP-Protokollstapels zum Einsatz kommen können. Im Unterschied zu den anderen Protokollen melden die Geräte direkt im Anschluss an ihre periodische Sendung den Bedarf der Übermittlung von nicht-periodischen Frames an. Alle Geräte dürfen bei Beginn der nächsten T_n-Phase ihre Daten senden, sofern sie von dem Management dazu befugt worden sind. Die Kommunikation erfolgt nach dem Client/Server-Modell. Die Reihenfolge, in der die Geräte senden dürfen, wird in der Konfigurationsphase festgehalten. Es handelt sich somit auch hier um ein Token-Passing Verfahren.

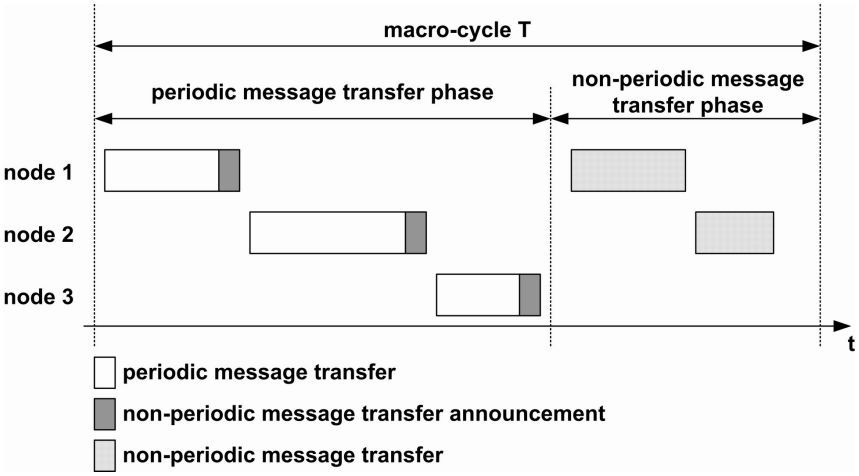


Abbildung 3.11: EPA-Zyklus [Dec06]

Da nahezu die gesamte Dokumentation ausschließlich in chinesischer Sprache verfasst ist, lassen sich weitere Informationen zu der Erstellung der Schedule und des Protokolls nicht ermitteln. Decotignie [Dec06] merkt jedoch in seiner Bewertung an, dass der garantierte Determinismus nur sehr aufwendig zu berechnen ist und einige Teile des Vorschlages zur Standardisierung „quite obscure“ seien.

3.1.2.4 RTnet

Historie

Anders als die bislang vorgestellten Ansätze ist *RTnet* [Kis06] ein offenes Projekt unter GNU *General Public License* (GPL) der Universität Hannover, die seit 2001 mit der Weiterentwicklung beschäftigt ist. Neben der Anbindung an Ethernet wird auch ein echtzeitfähiges FireWire-Protokoll beschrieben. RTnet soll neben echtzeitkritischen Prozessen der Automatisierungstechnik auch für die Übertragung kritischer Multimedia-Daten [SJHH02] geeignet sein.

Infrastruktur

Bei RTnet handelt sich um eine reine Software-Lösung, welche direkt auf das Treiber-Interface von Netzwerkkarten aufsetzt. Dazu existiert eine Liste von Netzwerkkarten, die bereits erfolgreich mit RTnet getestet wurden [Kis06]. Die Implementierung von RTnet erfolgt unter Linux ab Kernel-Version 2.6.x, welches um das *Real Time Application Interface* (RTAI) [Pol06] erweitert werden muss. RTnet empfiehlt aufgrund der kürzeren Verzögerungszeiten die Verwendung von $10Mbit/s$- oder $100Mbit/s$-Hubs, jedoch ist es auch mit $100Mbit/s$- und $1000Mbit/s$-Switches funktionsfähig. Es handelt sich also um eine baumförmige Topologie unter Verwendung von Standard-Ethernet Hardware.

Erreichen der Echtzeitfähigkeit

Die Beschreibung des Verfahrens zum Erreichen der Echtzeitfähigkeit ist im Wesentlichen dem Fachbeitrag von Kiszka und Schwebel [KS04] entnommen. Das Ziel von RTnet ist es nicht, unter der Einbuße der Ethernet-Kompatibilität oder sogar unter der Verwendung von „*grundlegend überarbeiteten Ethernet-Controllern*“ Zykluszeiten im Mikrosekundenbereich zu erhalten. Statt dessen liegt die „*Breite der industriellen Anwendungen*“ im Fokus der Entwickler. Auch bei RTnet müssen sich alle Geräte im Subnetz an das verwendete Protokoll halten, da die Daten sonst auf der physikalischen Schicht zu einer nicht-deterministischen Kollisionen bei der Verwendung von Hubs und zu einer Pufferung in Switches führen würde.

„*RTnet erreicht die Echtzeitfähigkeit zum einen durch die genaue Kontrolle der Absendezeitpunkte von Nachrichten. Dadurch werden sowohl Kollisionen auf dem Netzwerk verhindert als auch Überlastungen einzelner Teilnehmer oder der Infrastrukturkomponenten unterbunden, die bei alleiniger Verwendung von Standard-Switches nicht auszuschließen sind. Nach welchem Protokoll der Medienzugriff gewährt wird, entscheidet ein modular eingebundenes Zugriffsverfahren. [...] Neben der Kontrolle des Sendezugriffs stellt die streng deterministische Implementierung des Protokollstapels die zweite Hauptsäule dar, auf denen die harten Echtzeiteigenschaften von RTnet beruhen. So weisen herkömmliche Protokollstapel eine erhebliche Anzahl konkurrierend genutzter Ressourcen auf, etwa Datenpuffer, CPU-Zeit, logische Adressen oder Eingangswarteschlangen. Um eine lastunabhängige Reaktivität des Protokollstapels sicherstellen zu können, wurden kritische Komponenten eingehend hinsichtlich der maximalen Ausführungsdauer optimiert. [...] So ist es in jedem Fall sichergestellt, dass sich ein Paket, welches von der Applikation zum Senden übergeben wird, in deterministischer Zeit auch physikalisch auf dem Kabel befindet.*“ [KS04]

Der Medienzugriff bei RTnet erfolgt über ein Master/Slave-Protokoll, welches auch für die Synchronisation innerhalb eines Netzwerk-Segmentes verantwortlich ist und die MTU der Netzwerkkarte überschreiben kann. Neben einem TDMA-Zugriffsverfahren kann bei der Konfiguration des Masters auch ein Token-Passing Verfahren gewählt werden; die Art des Zugriffsverfahrens ist als Softwarekomponente in RTnet integriert. Die Konfiguration erfolgt während der Hochlaufphase entweder statisch oder zentral im Netzwerk durch die Verwendung des RTcfg-Protokolls. In seiner ersten Sendung sendet jedes Gerät einen Frame mit einer Anfrage zur Kalibrierung an den Master, der dann mit einer Nachricht antwortet, welche die Ankunftszeit des Requestes und das Absetzen der Antwort in das Netzwerk enthält. Dadurch kann jeder Slave seine Round-Trip Zeit schätzen. Dieser Vorgang wird einige Male wiederholt. Nach der Konfigurationsphase wechselt das Netzwerk in den Echtzeit-Betrieb, der vom Master kontrolliert wird.

Dabei synchronisiert der Master die Geräte, wie bereits bei anderen Verfahren vorgestellt, durch das periodische Broadcast-Senden einer Startnachricht zu Beginn jedes Zyklus. Gleichzeitig wird die Uhrensynchronisation vorgenommen. Interessant ist bei RTnet die Möglichkeit eines Backup-Masters für den Fall, dass der Master ausfallen sollte. In diesem Fall sendet dieser etwas später einen Backup-Synchronisations-Frame, wie in Abbildung 3.12 zu erkennen ist. Hier lässt sich erkennen, dass jede Hardware-Redundanz in der Automatisierungstechnik zusätzlich Zeit in Anspruch nimmt. Anstatt auf die Zeitstem-

pel innerhalb der NICs zuzugreifen, die nicht einheitlich oder je nach Netzwerkkarte gar nicht existieren, wird im RTnet-Treiber ein Zeitstempel direkt nach Aufruf des Interrupt-Handlers gesetzt. Dabei verlässt sich RTnet im Unterschied zu EPL ausschließlich auf das Echtzeitbetriebssystem, so dass neben dem Hardware-Jitter ein Jitter des Betriebssystems zu addieren ist.

Während der Konfigurationsphase kann die Reihenfolge der sendenden Geräte incl. der Interframe-Gaps eingestellt werden. Während die Konfiguration in Abbildung 3.12a mit einheitlicher Paketgröße arbeitet, ist dies in der Konfiguration in Abbildung 3.12b nicht der Fall. Dort ist auch dargestellt, dass ein Gerät auch mehrmals hintereinander die Chance zum Senden erhalten kann. Abbildung 3.12c zeigt, dass sich außerdem mehrere Geräte einen Zeitslot teilen können.

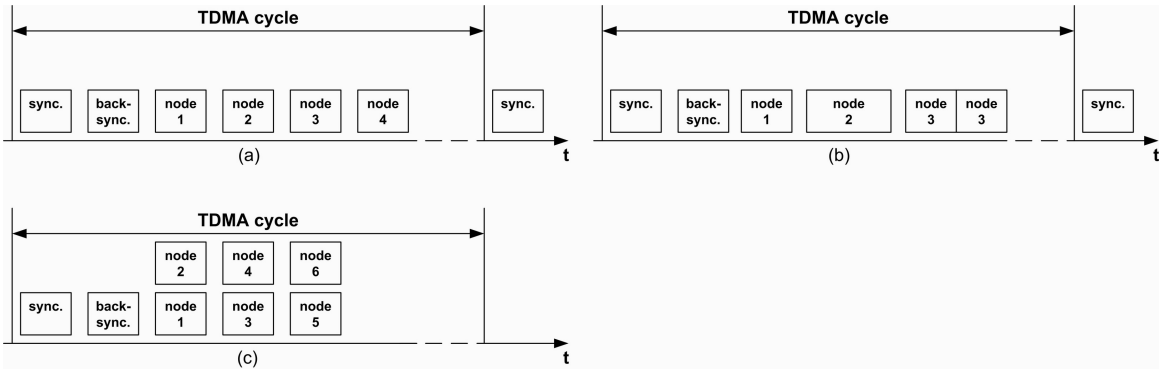


Abbildung 3.12: Zyklen von drei RTnet-Konfigurationen [KWZB05]

In dem Fall, dass sich n Geräte einen Slot teilen, wird jedoch nicht bestimmt, dass jedes Gerät in nur jedem n-ten Zyklus sendet. Statt dessen erfolgt hier eine Priorisierung der Frames. Dabei können 32 Stufen verwendet werden, wobei die letzte Stufe für zeitunkritische Daten vorgesehen ist. Es kann nicht vorkommen, dass ein Gerät mehr Daten versendet als in der Schedule festgelegt wurde, da die MTU von der über der Sicherungsschicht implementierten Zugriffskontrolle überwacht wird. Die Sendeberechtigung wird entweder zentral durch den Master über die Token-Vergabe oder dezentral nach der Festlegung der Zeitslots im TDMA-Zugriffsverfahren vergeben.

Das RTnet-Schichtenmodell

RTnet setzt, wie bereits beschrieben, im Rahmen des echtzeitfähigen Betriebssystems über dem Treiber der Netzwerkkarte auf. Der RTnet-Kern schirmt den Zugriff aus höheren Schichten auf die Netzwerkkarte vollständig ab. Das Modul, welches die Schedule enthält, wird als RTmac bezeichnet. Vor dem Erreichen des Netzwerkkarten-Treibers (NIC-Treiber) werden die ggf. asynchron gesendeten Pakete durch RTmac abgefangen und dynamisch in die TDMA-Schedule eingefügt.

Direkt in den RTnet-Kern wurde RTcap eingefügt, bei dem es sich um ein Packet Capture Interface zu der weit verbreiteten Open-Source-Software Etherreal/Wireshark [Eth07] zu Analysezwecken des Netzwerkverkehrs handelt. Durch die Optimierung des Protokollstapels bietet RTnet eine echtzeitfähige Realisierung von UDP/IP, ICMP und ARP an [KS04].

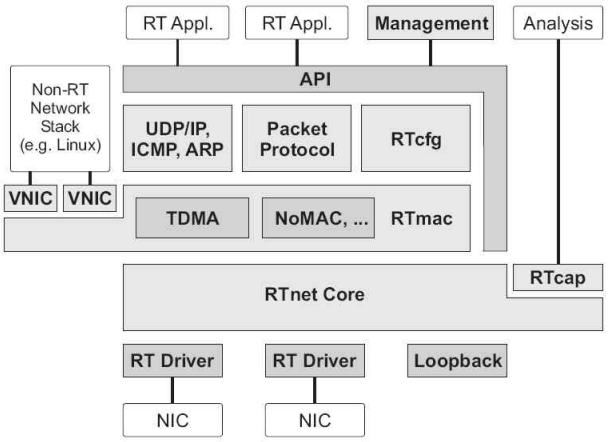


Abbildung 3.13: Schichtenmodell von RTnet [KWZB05]

Eine interessante Eigenschaft von RTnet liegt darin, dass die gesamte RTmac-Schicht und damit das Zugriffsverfahren austauschbar ist. Neben *TDMA* wird ein Token-Passing Verfahren vorgestellt [HJSM05], bei der ein Token zwischen den Geräten weitergereicht wird. Die Idee, ein deterministisches Ethernet durch die Einführung eines Token-basierten Verfahrens einzusetzen, wurde bereits 1994 von Chiueh und Venkatramani [CV94] beschrieben. RTnet wurde auf Grundlage des daraus entstandenen Protokolls *Real-Time Ethernet* (RETHER) [Chi99] entwickelt. Ziel ist dabei die Kollisionsvermeidung und die Gewährleistung von QoS-Garantien. Der Eigentümer des Token ist berechtigt, Echtzeit- oder Nicht-Echtzeit-Frames zu senden. Die Zeitdauer, die das Token behalten werden darf, wird als *Token Holding Time* (THT) bezeichnet. Im Gegensatz zu einfachen Token-Verfahren ist bei RTnet die THT nicht für alle Geräte identisch, sondern wird in der Konfigurationsphase des Netzwerkes im Rahmen der Schedule-Berechnung festgelegt. Um den Verlust des Tokens zu vermeiden, übernimmt jeweils der vorletzte Besitzer des Tokens die Rolle eines Monitors, der die Weitergabe des Tokens vom nächsten zum übernächsten Gerät überwacht. Hansen, Jansen, Scholten und Hattink [HJSM05] beschreiben den Token-Ansatz für die Übertragung von Multimedia-Daten, der jedoch auch für die Automatisierungstechnik übertragbar ist.

Meßergebnisse

Obwohl noch kein industrieller Einsatz durchgeführt wurde, existieren bereits Messungen mit RTnet, die sich auf die Auslastung verschiedener CPUs (in %) in Abhängigkeit der Framegröße und der Gesamtzykluszeit konzentrieren. Bei einem dreistündigen Test, der wahrscheinlich ohne eine kaskadierte Netzwerkinfrastruktur erfolgte, wurden folgende Ergebnisse gemessen:

Es wurden Messungen mit Standard-PCs Intel Pentium $90MHz$, $150MHz$ und $266MHz$ durchgeführt mit den Zykluszeiten von $2.5ms$ (a), $5.0ms$ (b) und $10.0ms$ (c). Deutlich zu erkennen ist die Abnahme der auf der y-Achse dargestellten CPU-Belastung bei größeren Zykluszeiten. Auf der x-Achse werden verschiedene Framegrößen abgebildet. Die Zykluszeit sowie der Jitter bis zu $38\mu s$ [KHW03] kann die härteste vierte IAONA-Echtzeitklasse nicht erreichen.

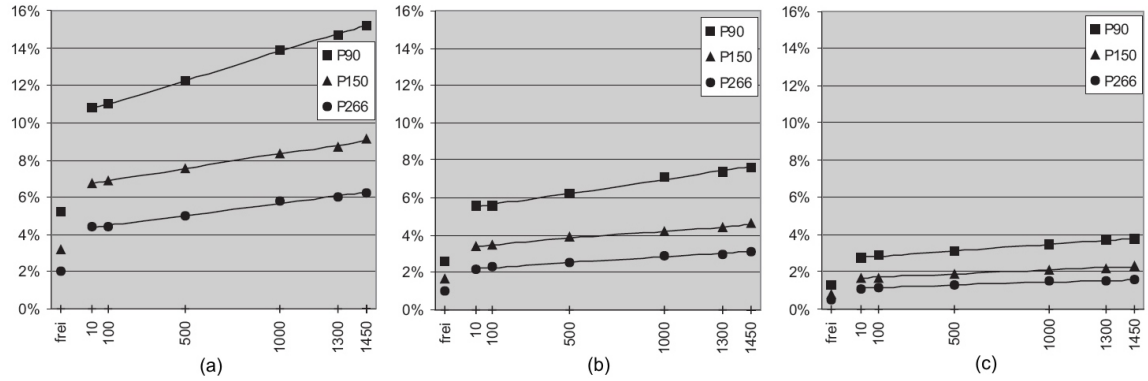


Abbildung 3.14: Performance-Messungen mit RTnet [KHW03]

3.1.2.5 ProfiNet Soft Realtime (SRT)

Historie

Im August 2000 stellte die Profibus Nutzerorganisation e.V. erstmals das Automatisierungsmodell ProfiNet vor, dass sich historisch aus den Feldbussen Profibus (s. Kapitel 2.1.3.6) und Interbus [BM94] mit dem Trend zum echtzeitfähigen Ethernet entwickelt hat. In der PNO sind mehr als 260 Hersteller und Anwender eingetragen, wobei die Siemens AG als Weltmarktführer der Automatisierungstechnik zentralen Anteil an der Weiterentwicklung von ProfiNet besitzt. Seit 2003 ist ProfiNet gemäß IEC 61158/61784 und IEC SC 65C genormt.

In der ersten Version aus dem Jahre 2002 verwendet ProfiNet Standard-Ethernet unter Verwendung des TCP/IP-Protokollstapels für die zeitunkritische Kommunikation übergeordneter Geräte. Dabei wird das Distributed Component Object Model (DCOM) von Microsoft als gemeinsames Applikationsprotokoll zwischen ProfiNet-Geräten verwendet. Neben dem Zugriff vom Planungswerkzeug Step7, welches z.B. für das Verwalten der Anlagen-Konfiguration, das Lesen von Diagnosedaten und die Geräteparametrierung verantwortlich ist, erfolgt auch die Etablierung der Änderungen und die Kommunikation zwischen den Automatisierungsgeräten - also der Nutzdatenaustausch zwischen den Komponenten - über DCOM. Echtzeitkritische Geräte wurden in der ersten Version von ProfiNet in konventionellen Profibus-Subnetzen gekapselt, die über Proxies vom Ethernet aus angesprochen werden können. In dieser ersten Version, die als ProfiNet nRT (non-realtime) bezeichnet wird, besteht die Infrastruktur also aus zwei getrennten Netzen, die über Proxies verbunden sind. Die Trennung der Ebenen der Pyramide der Automatisierungstechnik bleibt also in der ersten Version noch bestehen. „*Die komponentenbasierte Kommunikation [...] ermöglicht Zykluszeiten in der Größenordnung von* $100ms$“ [PW04].

Mit ProfiNet in der Version 2.0, das seit 2003 verfügbar ist, werden die Echtzeit-Subnetze mit dem Ethernet über OSI-Schicht 2 auf einer Leitung vereinigt. Dieser Ansatz, der unter dem Namen *ProfiNet Soft Realtime* (SRT) bzw. *Component Based Automation* (CBA) bekannt ist, wird in diesem Kapitel vorgestellt.

Das Ziel von ProfiNet liegt nicht nur in der Einführung eines weiteren echtzeitfähigen Ethernet Busses, sondern es soll ein „*innovativer Automatisierungsstandard der Profibus-Nutzerorganisation für die Realisierung einer ganzheitlichen und durchgängigen Automati-*

sierungslösung auf Basis von Industrial Ethernet“ [Hei04] geschaffen werden. Der Lebenslauf einer Anlage soll vom Engineering bis zum Betrieb und Wartung mit Entwicklungs- und Diagnosewerkzeugen unterstützt werden, um eine durchgängige Datenintegration in den Unternehmen zu erreichen.

Infrastruktur

ProfiNet setzt in allen Versionen auf $100Mbit/s$-Ethernet mit einer Baumtopologie, wobei zur Verkabelung auch industrietaugliche Stecker verwendet werden können. Es können für ProfiNet SRT herkömmliche Switches verwendet werden, die VLAN-fähig sind. In der industriellen Praxis werden jedoch nahezu aussschließlich Siemens-eigene Geräte verwendet, die in Step7 mit ihren Eigenschaften und technischen Daten registriert sind. So bietet Siemens beispielsweise eigene Switches und Gateways mit industrie-tauglichen Ports an sowie die Verwendung von Lichtwellenleitern für Umgebungen mit elektromagnetischen Störungen. Viele Geräte beinhalten integrierte 3-Port Switches, von denen ein Port direkt vom Gerät verwendet wird. Auf diese Weise wird eine Linientopologie emuliert.

Erreichen der Echtzeitfähigkeit

Als Verzögerungszeiten geben Pigan und Metter [PM06] unter Kenntnisnahme der Siemens AG die folgenden Werte an:

Kabel:	$5ns/m$
Protokollstapel im Sender/Empfänger:	$100 - 300ns$
Switch:	$10\mu s/Switch$

Da die Netzwerktopologie in dem Planungswerkzeug abgespeichert wird, lassen sich anhand dieser Zeiten die Verzögerungen im Netzwerk kalkulieren. Werden Siemens-eigene Switches eingesetzt, so kann deren Verzögerungszeit genauer angegeben werden, da diese Geräte bereits als Objekte im Planungswerkzeug hinterlegt sind. Ansonsten müssen ggf. die Parameter der Standard-Switches als Eingaben vorgegeben werden, für deren Korrektheit dann der Anwender verantwortlich ist.

Um die Echtzeitfähigkeit unter Verwendung von Standard-Switches zu erreichen, verwendet Profinet SRT ebenso wie EtherNet/IP die Priorisierung mit VLAN-Headern [Pop05]. Dabei wird für ProfiNet-Frames die zweithöchste oder höchste Priorität [Joh03] im TCI-Feld des VLAN-Headers gesetzt. „*Die Real-Time Kommunikation ermöglicht Zykluszeiten in der Größenordnung von* $10ms$ *und ist für den Einsatz im Bereich dezentraler Peripherie sehr gut geeignet.*“ [PW04] Zur Anwendung kommt dabei das Soft Realtime Protokoll von Siemens, welches nach dem Publisher/Subscriber-Modell arbeitet und sich auch für langsame I/O-Anwendungen auf der Feldebene eignet.

Nach der Parametrierung der Anlage können in einem Testbetrieb die gerätespezifischen Performance-Parameter, die in dem Planungswerkzeug hinterlegt sind, mit den aktuellen Daten verglichen werden. Auf diese Weise kann die Lastgrenze des Netzwerkes erkannt werden. Denn eine Überlastung des Netzwerkes oder das ausschließliche Vorhandensein einer großen Anzahl von hochprioren Frames würde den Effekt der VLAN-Priorisierung aufheben und wiederum in einem nicht-deterministischen Verhalten resultieren.

Andererseits bedeutet dies, dass Netzwerkverkehr durch asynchron sendende Geräte, die ggf. erst im laufenden Betrieb der Anlage hinzugefügt werden, nicht berücksichtigt wird. Daher bleibt auch ProfiNet SRT in gewisser Weise ein geschlossenes Netzwerk, da zusätzliche Geräte im Subnetz die Echtzeitfähigkeit beeinträchtigen können.

ProfiNet SRT bietet einerseits eine zyklische Übertragung mit festem Raster, in dem die Publisher ihre Daten für die Consumer aktualisieren. Die Projektierung und „Verschaltung“ der Daten wird offline in dem Planungswerkzeug durchgeführt, so dass die notwendigen Schedules offline berechnet werden können. Diese Verschaltung ist in Abbildung 3.15 skizziert.

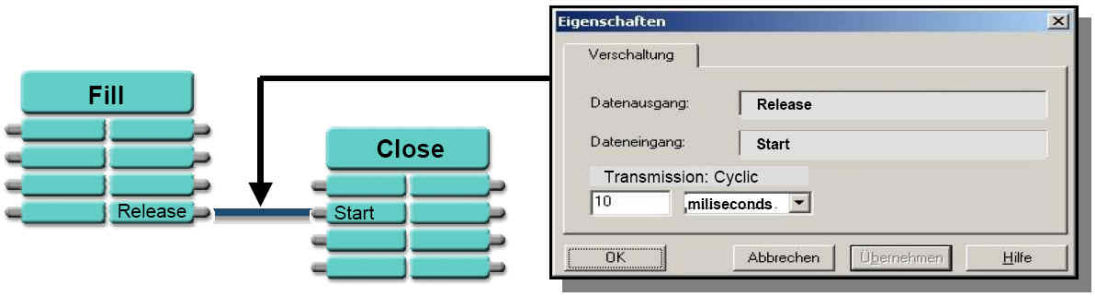


Abbildung 3.15: Komponentenbasierte Verschaltung von Daten mit ProfiNet SRT [Joh03]

Die Daten können mit einem Zeitstempel nach IEEE 1588 versehen werden, der softwareseitig oberhalb der Ethernet-Schicht verwaltet wird. Im Gegensatz zu EtherNet/IP und ProfiNet IRT setzt ProfiNet SRT keine zusätzliche Hardware zur Synchronisation ein.

Um die Betriebsfähigkeit zu gewährleisten, definiert ProfiNet Regeln [Pop05], die für eine stabile Datenübertragung sorgen. Dabei sollen nicht mehr Frames übertragen werden als 60 % der Bandbreite entsprechen und es soll nicht mehr als 10 % der Bandbreite für die Übertragung von Alarmen vorgesehen werden. Unter Alarmen versteht man hochpriore Ereignisse wie das Betätigen eines Not-Aus Schalters oder kritische Fehler- und Statusmeldungen, die nicht verworfen oder verzögert werden sollten. Neben den zyklischen RT-Frames unterscheidet ProfiNet aRT-Frames, in denen azyklisch zeitunkritische Ereignisse wie Statusmeldungen übertragen werden. Weitere Regeln bestehen darin, dass zyklische Frames der RT-Klasse nicht mehr als 50 % der Bandbreite ausmachen dürfen.

In der aktuellen dritten Version von ProfiNet können für die aRT-Frames, Zykluszeiten zwischen $31.25\mu s$ und $4ms$ definiert werden [Pop05]. Bei solch geringen Zykluszeiten im μs-Bereich ist jedoch die Verwendung von modifizierter Hardware erforderlich, wie sie in Kapitel 3.1.3.1 beschrieben wird. Diese Version beinhaltet jedoch das SRT Protokoll unverändert.

Abbildung 3.16 beschreibt die Aufteilung der zyklischen Sendungen im Rahmen der aktuellen dritten Version, bei der modifizierte Hardware obligatorisch ist. Die notwendige Zeit für einen Zyklus bezeichnet ProfiNet mit T_{SC} (SendClock). Das SRT Protokoll kann jedoch auch ohne Modifikation der Hardware zum Einsatz kommen, garantiert in diesem Falle jedoch keine Zykluszeiten im μs-Bereich.

Eine Überlastung kann dann auftreten, wenn gegen Ende des geplanten Bus-Zyklus neue nRT-Frames versendet werden sollen. Bei der Verwendung von Standard-Hardware verschiebt sich dadurch der Beginn des folgenden Zyklus, wodurch auf Dauer die Anla-

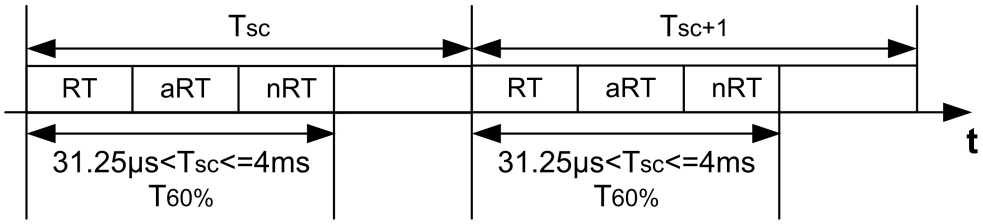


Abbildung 3.16: Aufteilung der Bandbreite von ProfiNet SRT [Pop05]

ge abgeschaltet werden müsste. Werden die beschriebenen Regeln zur Vermeidung von Überlastungen verletzt, so muss zunächst die bereits begonnene Sendung zu Ende geführt werden. Da die Echtzeit bei ProfiNet SRT nicht isochron ist, wird der Versand der RT-Frames verschoben, so dass ein erhöhter Jitter entsteht. Im Beispiel der Abbildung 3.17 ist die Synchronität wieder bei $T_{SC} + 2$ erreicht. Die Verwendung von VLAN-Tags zur Priorisierung und die Reservierung von Bandbreite für RT-Daten kann also keinen Determinismus und keine isochrone Kommunikation im Ethernet in der Weise ermöglichen, wie sie von der Antriebstechnik gefordert wird.

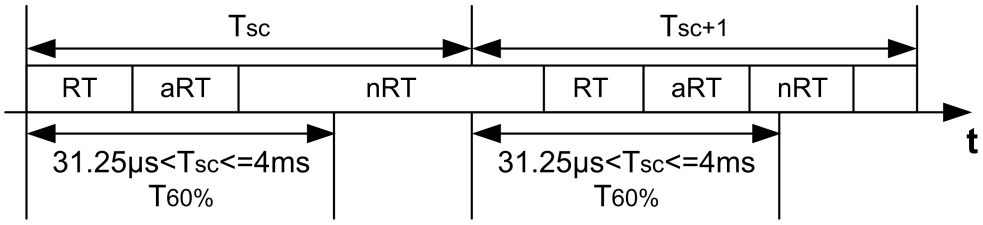


Abbildung 3.17: Verletzung der Echtzeit durch Überlastung [Pop05]

Das Schichtenmodell von ProfiNet CBA

Das SRT Protokoll umgeht den Overhead des TCP/IP-Protokollstapels und reicht die Daten direkt von der Sicherungsschicht auf die Anwendungsebene weiter, wie es in Abbildung 3.18 dargestellt ist. Zur Anlagenkonfiguration und Geräteparametrierung bleibt die Protokoll-Hierarchie der Internet-Protokolle mit DCOM in der Anwendungsschicht aus der ersten Version von ProfiNet bestehen und ermöglicht eine komponentenbasierte Parametrierung.

Das Echtzeitverhalten des Netzwerks resultiert aus der VLAN-Priorisierung, der Einführung von Regeln und aus der Reduktion des Protokollstapels im Vergleich zu TCP/IP-basierten Anwendungen. Die härteste Echtzeitklasse der IAONA kann auf diese Weise jedoch nicht erreicht werden.

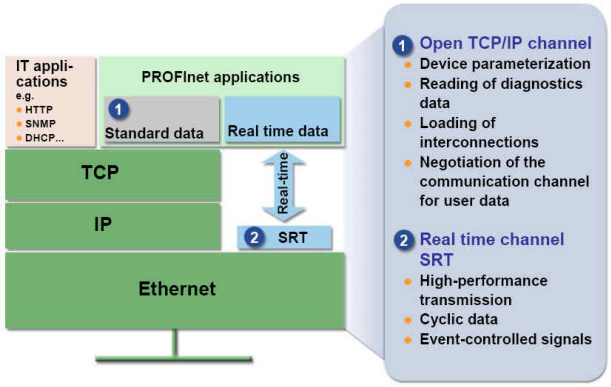


Abbildung 3.18: Schichtenmodell von ProfiNet RT [Fel05c]

3.1.3 Echtzeitfähigkeit durch Modifizierung der Ethernet-Schicht

3.1.3.1 ProfiNet Isochronous Realtime (IRT)

Historie

Einen weiteren Schritt zur Erreichung harter Echtzeitfähigkeit und zur Synchronisierung von Antrieben tätigt *ProfiNet IO* (Input/Output) mit seinem Ansatz des *Isochronous Real Time* (IRT) bei, das im Jahre 2004 zentraler Bestandteil der dritten Version von ProfiNet ist [PN04]. Dabei bleibt die SRT-Kommunikation ebenso erhalten wie die Fähigkeit zur asynchronen Kommunikation über den TCP/IP-Protokollstapel. Abbildung 3.19 stellt die drei Entwicklungsstufen von ProfiNet gegenüber und trifft Aussagen über die Verzögerungszeiten sowie über die Jitter in Prozent.

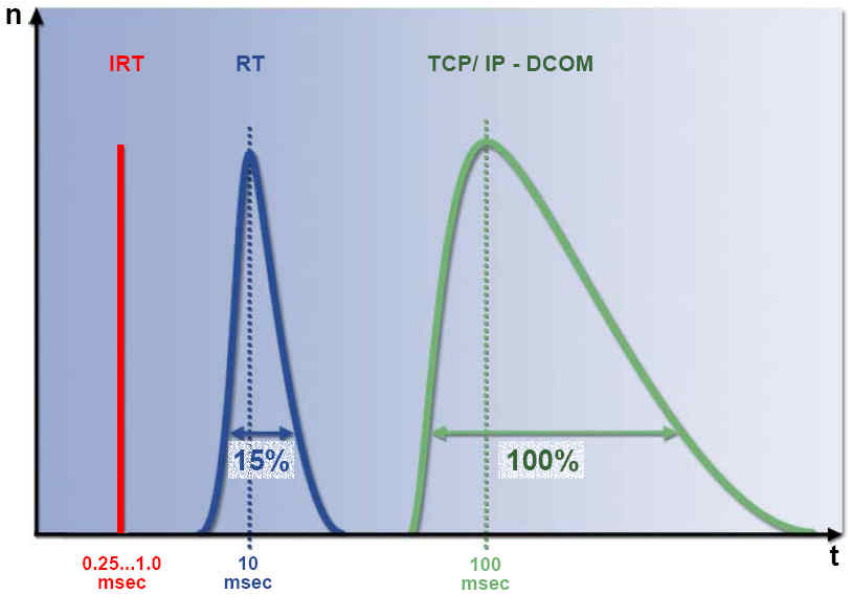


Abbildung 3.19: Verzögerungszeiten und Jitter der ProfiNet-Versionen [Fel05c]

Infrastruktur

Bereits die Ansätze von EtherNet/IP und ProfiNet SRT haben gezeigt, dass auf der Basis eines gängigen $100Mbit/s$ Ethernet-Netzwerkes mit COTS-Switches und Vollduplexbetrieb die härteste Echtzeitklasse der IAONA nicht erreicht werden kann. Aus diesem Grunde sind modifizierte Ethernet-Switches wie der *Enhanced Real Time Ethernet Controller* (ERTEC) 200 und 400 Ausgangspunkt für ProfiNet IRT [Pop05]. Ein Blockschaltbild dieses Switches ist in Abbildung 3.20 dargestellt.

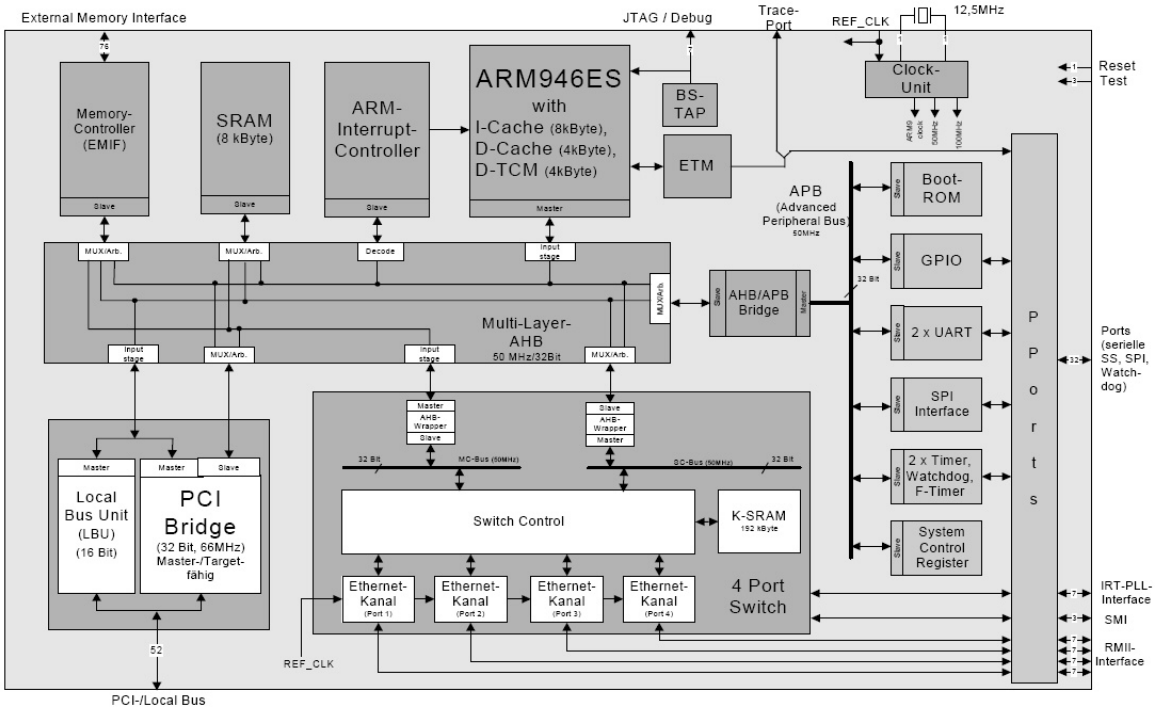


Abbildung 3.20: Blockschaltbild des ERTEC 400 ASIC [PM06]

In beiden Fällen stellt Siemens einen Application Specific Integrated Circuit (ASIC) zur Verfügung, den andere Hersteller in ihre Geräte integrieren können. Der ERTEC 200 eignet sich dabei insbesondere zur Integration in Feldgeräte und bietet zwei Ports nach außen an, während es sich bei dem ERTEC 400 um einen 4-Port Switch handelt, der auch als PCI-Version in herkömmliche PCs eingebaut werden kann [FFMAV07]. Während der 2-Port Switch ERTEC 200 zwei *Physical-Layer Bausteine* (PHY) integriert, sind diese beim ERTEC 400 ASIC separat hinzuzufügen. Der ERTEC 400 bietet statt dessen 4 vollduplexfähige *Reduced Media Independent Interfaces* (RMII) zur Kommunikation mit der physikalischen Schicht (s. Kapitel 5.3.2). Diese Interfaces sind im Ethernet-Umfeld zwischen der physikalischen Schicht und der Datensicherungsschicht etabliert, so dass PHYs beliebiger Hersteller an den ProfiNet-ASIC angekoppelt werden können. Für den IRT-Modus dürfen die Jitter dieser Bausteine jedoch gewisse Grenzen nicht überschreiten. ProfiNet hat zu diesem Zweck eine Liste mit geeigneten Bausteinen veröffentlicht [Pop05].

Obwohl Siemens diesen Chip im Rahmen der Profibus Nutzerorganisation der Branche als offene Lösung zur Verfügung stellt, bleibt ProfiNet ein proprietärer Ansatz. Der ASIC ist durch seine modifizierte Switching-Engine in der Lage, im Gegensatz zu herkömmlichen

Switches isochrone Frames unabhängig von aRT- und SRT-Frames zu behandeln. Der interne Aufbau der Switch Control (vgl. Abbildung 3.20) ist jedoch nicht veröffentlicht. In der Hochlaufphase werden die Switches mit ihren lokalen Schedules geladen und statische Adresstabellen für die echtzeitfähigen Geräte aufgebaut. Auf diese Weise sind die Routen für isochrone Daten beim Start der echtzeitkritischen Anwendung bereits bekannt.

ProfiNet bietet in den aktuellen Datenblättern der Switches [Pro07] [Pro07b] lediglich einen Hinweis auf die Taktanbindung der PHYs an den ASIC. Diese Anbindung ist in Abbildung 3.21 skizziert. Während die PHYs in einer RMII-Ankopplung[2] mit einem externen Taktsignal betrieben werden können, müssen bei einzelnen PHYs mit *Media Independent Interface* (MII) Schnittstelle (s. Kapitel 5.3.1) die Taktausgänge der PHYs selbst verwendet werden. Es existiert für jeden PHY-Baustein jeweils ein separates Signal für den Sendetakt und für den Empfangstakt.

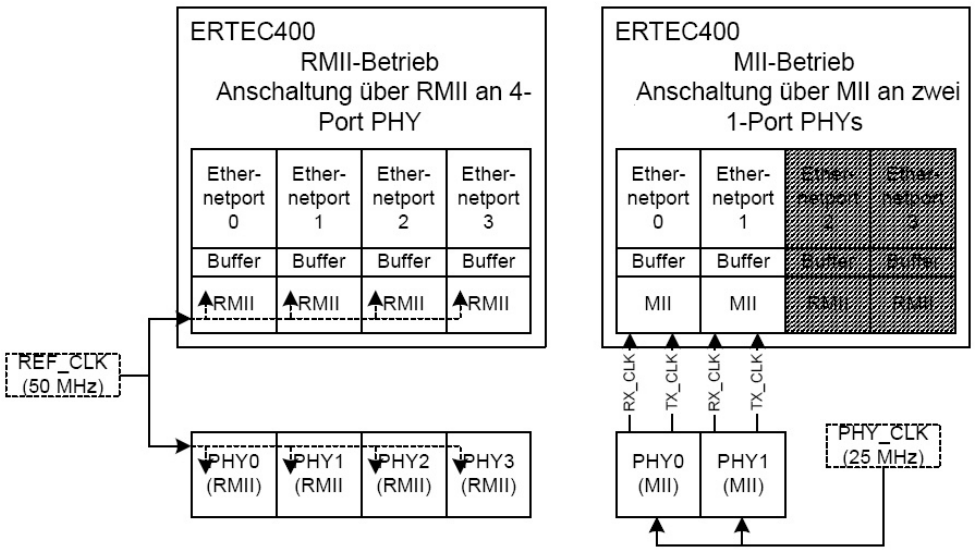


Abbildung 3.21: Taktung des ERTEC 400 ASIC am MII [Pro07b]

Des Weiteren erwähnen Popp und Weber in ihrer Veröffentlichung im Rahmen der ProfiBus Nutzerorganisation e.V., dass der ERTEC 200 „*über zwei integrierte PHYs mit Auto-Crossover-Funktionalität*“ [PM06] verfügt. Es ist also anzunehmen, dass die ERTEC-Switches die Ports direkt über der physikalischen Schicht verschalten und auf diese Weise eine geringere Verzögerungszeit als herkömmliche Switches erreichen können. „*Neben der als Dual-Port Random Access Memory (DPRAM) ausgeführten Schnittstelle für Realtime-Daten ermöglicht der ERTEC 200 über seine Standard-Schnittstelle das Aufsetzen von TCP/IP-Anwendungen und IT-Funktionen*“ [PM06]. Eine genauere Betrachtung der technischen Realisierbarkeit dieses Ansatzes wird am Rande dieser Arbeit in Kapitel 5 erläutert.

[2]Es sind einzelne Bausteine erhältlich, die bereits 4 PHY-Einheiten enthalten.

In der dritten Version setzt ProfiNet also auf modifizierte Switches und einer baumförmigen Verkabelung. Eine Linientopologie kann durch Kaskadierung der Switches emuliert werden. Zusätzlich können die Switches zum Zwecke der Redundanzerhöhung auch in Ringtopologien angeordnet werden. Die Kommunikation erfolgt weiterhin nach dem Publisher/Subscriber-Modell.

Erreichen der Echtzeitfähigkeit

Die Idee von ProfiNet IRT besteht also darin, das Prinzip der Verwendung von Switches auszunutzen. In einem Netzwerk, in dem ausschließlich Switches eingesetzt werden, finden keine Kollisionen statt. Das Zwischenspeichern von Frames kann verhindert werden, indem zu jedem Zeitpunkt genau ein Frame am Ausgang eines Ports anliegt, der dann unverzüglich gesendet wird. Ausgenutzt werden kann die Fähigkeit dieses Netzwerkes zum Vollduplexbetrieb, in dem zwei Frames gleichzeitig in entgegengesetzter Richtung auf einer Leitung versendet werden können, ohne dass eine zusätzliche Verzögerung oder ein nichtdeterministisches Verhalten resultiert.

Da sowohl die Topologie, als auch die Programmierung der automatisierten Anlage dem Planungs-Werkzeug bekannt ist, kann ein Planungsalgorithmus angewendet werden, dessen Resultat eine Schedule der Übertragungen ist. Dieser Algorithmus wurde von Siemens nicht veröffentlicht. Diese Schedule wird in jeden Switch geladen und die Switches werden zueinander synchronisiert. Der Ablauf wird im Folgenden genauer erläutert, da er eine wichtige Grundlage für die weitere Arbeit darstellt. Abbildung 3.22 zeigt das Problem, dass die Switches 4 und 5 mehrere IRT-Übertragungen über ein Kabel leiten müssen.

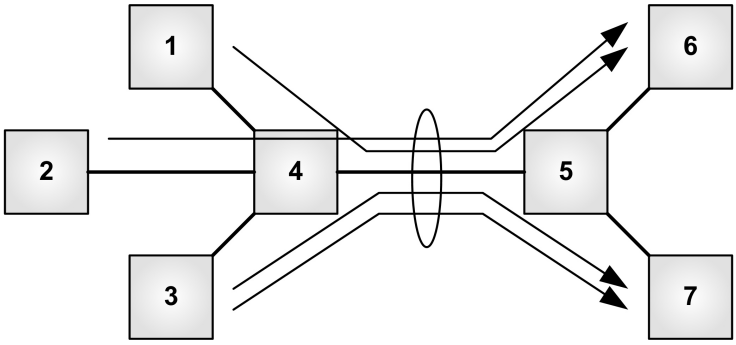


Abbildung 3.22: Übertragungen über zwei ProfiNet-Switches [Pop05]

Da diese Übertragungen in die gleiche Richtung verlaufen, müssen sie nacheinander ausgeführt werden. Da der Nutzdatenanteil eines IRT-Frames gering und die Übertragungsgeschwindigkeit des Ethernets mit $100 Mbit/s$ hoch ist, können die Frames unter Berücksichtigung der Interframe-Gap hintereinander gereiht werden. Die resultierenden Schedules werden als Teil der Netzwerk-Konfiguration gesehen und zusammen mit der Anlagenkonfiguration in der Hochlaufphase der Anlage geladen.

ProfiNet Version 3 unterscheidet - wie auch Ethernet PowerLink - in jedem Zyklus prinzipiell zwei Phasen. In der ersten Phase wird der isochrone Echtzeitverkehr weiter geleitet und im Anschluss daran eine Zeit für asynchronen Datenverkehr reserviert. Die erste Phase wird von ProfiNet als *rotes Intervall* bezeichnet. Die Länge des Intervalls ergibt sich

aus der Anzahl der isochronen Sendungen, die nacheinander ausgeführt werden müssen, sowie deren Übertragungsdauer. Die Länge der zweiten Phase kann bei der Projektierung der Anlage festgelegt werden. Sie wird als *grünes Intervall* bezeichnet und beinhaltet die Zeit, die man für zusätzlichen Verkehr erübrigen kann ohne die Echtzeitfähigkeit der Anlage zu stören. Die untereinander synchronisierten Switches schalten nun in jedem Zyklus zwischen diesen beiden Phasen um.

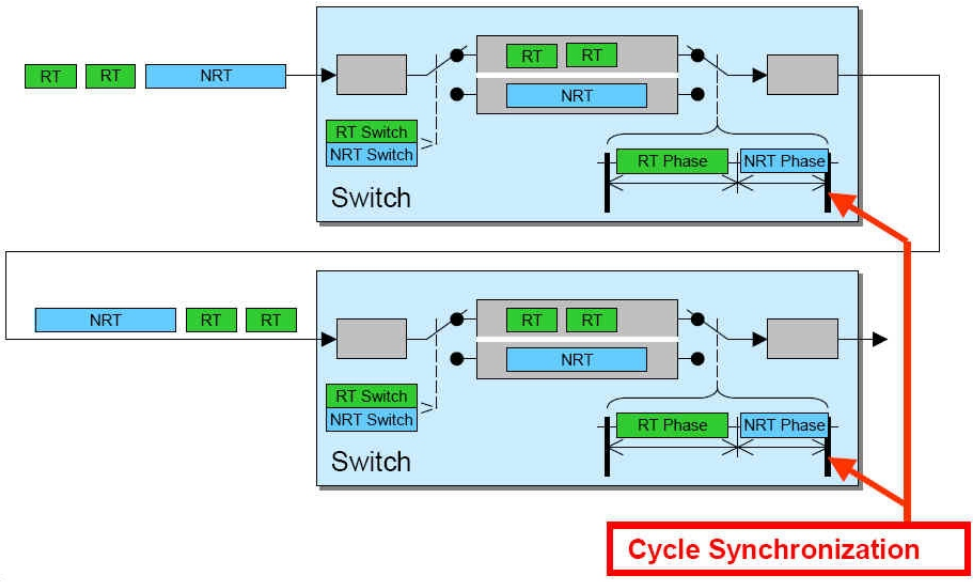


Abbildung 3.23: Schematischer Ablauf der Kommunikation über zwei Switches [Fel05c]

Auch wenn der isochrone Datenverkehr im Vorfeld bekannt ist, schalten die Switches innerhalb des roten Intervalls die Eingangsports nicht automatisch an die Ausgangsports durch. „*Eine zeitgesteuerte Bearbeitung der Aufträge innerhalb des roten Intervalls hilft, die letzten Quellen von Ungenauigkeiten zu beseitigen. Auch in Switches muss die Reihenfolge der Bearbeitung einzelner Aufträge genau eingehalten werden, weil sonst bei komplexeren, vermaschten Netzen keine exakte Kalkulation der Durchlaufzeit möglich ist. Der Fahrplan (Schedule) orientiert sich nur an der Sequenz der eintreffenden Frames, die durch ihre Frame-ID und die Länge bestimmt wird.*“ [Pop05] Die Verschaltung der PHY-Bausteine erfolgt demnach erst im Anschluss an die Interpretation der Frame-ID. Der Frame wird also bis zur Frame-ID eingelesen und zwischengespeichert. Abschließend erfolgt die direkte Durchschaltung an den Ausgangsport.

Die Frame-ID ist ein 2 Byte großes Feld, das bei IRT-Frames statt des VLAN-Tags eingefügt wird. Anhand dessen werden auch IRT-Frames von asynchronen Frames unterschieden, welche zwischengespeichert und erst im grünen Intervall weiter verarbeitet werden. Es handelt sich bei ProfiNet also nicht um eine reine TDMA-Schedule. Kann ein IRT-Frame in seinem Zeitslot nicht mehr gesendet werden oder trifft ein IRT-Frame außerhalb des roten Intervalls ein, so wird er verworfen, da seine Information bei einem späteren Empfang wertlos ist. Dieser Fall kann jedoch bei exakter Synchronisation nicht eintreten. Trifft im roten Intervall ein nRT-Frame ein, so wird er zwischengespeichert und nach Beendigung des Intervalls weiter geleitet.

Die Weiterleitung von Frames innerhalb des grünen Intervalls erfolgt nach IEEE802.3-Standard, wobei die VLAN-Priorisierung berücksichtigt wird. Dies bedeutet, dass sowohl die Profinet SRT Kommunikation, als auch die Sendung von TCP/IP-Daten im grünen Intervall gekapselt werden. In dem grünen Intervall können die Switches im cut-through oder im store-and-forward Modus arbeiten [Pop05].

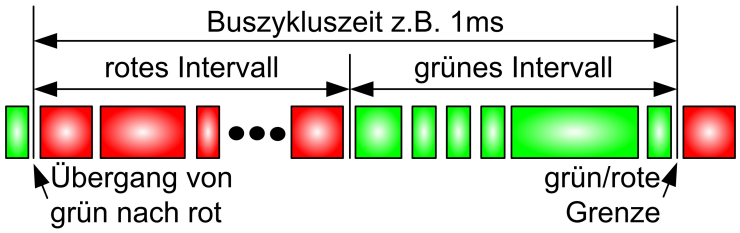


Abbildung 3.24: Rotes und grünes Intervall bei ProfiNet IRT [Pop05]

Die Aufteilung zwischen roten und grünen Intervallen ist flexibel. So muss nicht in jedem Zyklus ein rotes IRT-Intervall existieren. Genauso können mehrere rote Intervalle hintereinander definiert werden. Die Randbedingungen bestehen darin, dass das rote Intervall aufgrund der minimalen Framelänge mindestens $31.25\mu s$ und ein grünes Intervall aufgrund der maximalen Framelänge mindestens $125\mu s$ lang sein muss. Ein Buszyklus muss mindestens aus einem grünen Intervall bestehen. Werden rote und grüne Intervalle gemischt, so ergibt sich eine Mindestzeit von $156.25\mu s$. Die zyklischen Gesamtsendungen, auch als „send cycles“ bezeichnet, ergeben sich aus mehreren Buszyklen konstanter Länge.

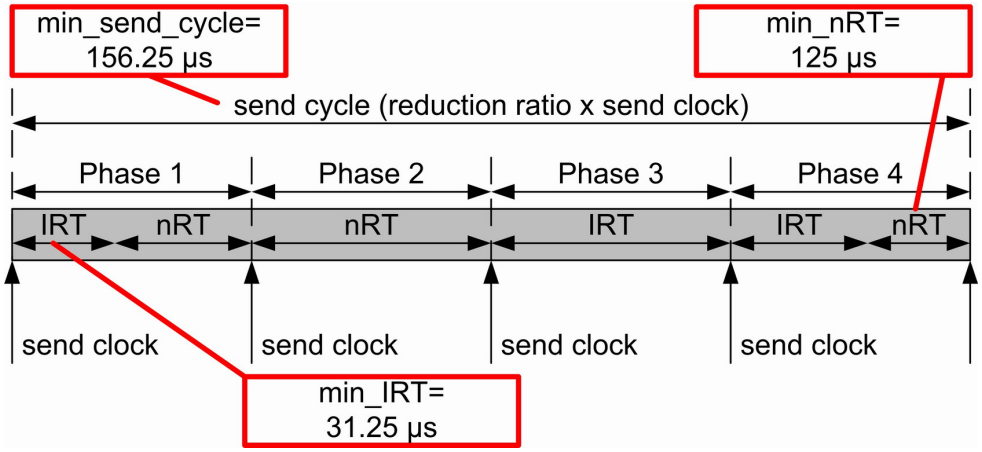


Abbildung 3.25: Zeiten der ProfiNet IRT Zyklen [Pop05]

Siemens kalkuliert den auftretenden Jitter zwischen zwei Ethernet-PHYs mit maximal $40ns$ [Pop05]. Dadurch können maximal 25 Geräte hintereinander geschaltet werden, um die Obergrenze des Jitters der härtesten IAONA-Echtzeitklasse $< 1\mu s$ einzuhalten. Siemens empfiehlt aus Sicherheitsgründen, die Ausdehnung des Gesamtnetzes auf 20 hintereinander geschaltete Geräte zu begrenzen. Die Durchlaufzeit eines Frames durch einen ERTEC-Switch wird aufgrund der verbesserten Weiterleitung durch die Schedule mit $3\mu s$ angegeben [Pop05] und ist damit kürzer als bei handelsüblichen Switches, kommt jedoch an die schnelle Weiterleitung eines Hubs nicht heran.

Popp [Pop05] hat zwei Berechnungsformeln mit Beispiel-Rechnungen angegeben, mit deren Hilfe sich typische Zeiten kalkulieren lassen. Diese Formeln sind so strukturiert, dass sie bei einer entsprechenden Parametrierung allgemein gültig sind:

1. Zeit $T_u(x)$, bis ein Frame bei Gerät x ankommt:
   $T_u(x) = A_t \cdot (Lz_b + DLZ_{Switch}) + T_u$
   A_t: Anzahl der ERTEC-Switches, die der Frame durchläuft
   Lz_b: Bitlaufzeit ($0.5\mu s$ bei $100Mbit/s$ und einer $100m$ langen Leitung)
   DLZ_{Switch}: $3\mu s$ (ERTEC-Switch)
   T_u: Übertragungszeit eines minimal großen Frames (gerundet auf $7\mu s$ bei $100Mbit/s$)

   So ergibt sich beispielsweise bei einem Durchlauf von 4 Switches eine Verzögerungszeit von $21\mu s$.

2. Anzahl der Geräte A_t, die in einer vorgegebenen Zeit aktualisiert werden können:
   $A_t = (IRT - Lz_b - DLZ_{Switch})/T_u$
   IRT : Gewählte Länge für den IRT-Zyklus (rotes Intervall)
   Lz_b: Bitlaufzeit ($0.5\mu s$ bei $100Mbit/s$ und einer $100m$ langen Leitung)
   DLZ_{Switch}: $3\mu s$ (ERTEC-Switch)
   T_u: Übertragungszeit eines minimal großen Frames (gerundet auf $7\mu s$ bei $100Mbit/s$)

   Bei einer gewählten Buszykluszeit von $1ms$, die sich aufteilt in $500\mu s$ für das rote und $500\mu s$ für das grüne Intervall ergibt sich eine Anzahl von 70 Geräten.

Die Einbindung von SRT- und Standard-Verkehr

Wie bereits in Kapitel 3.1.2.5 erwähnt, wird im grünen Intervall der im vorherigen Kapitel diskutierte SRT-Datenverkehr mit VLAN-Priorisierung ebenso abgewickelt wie asynchron versendete Frames. In der dritten Version können für die aRT-Frames, in denen azyklisch Alarminformationen übertragen werden, Zykluszeiten zwischen $31.25\mu s$ und $4ms$ definiert werden [Pop05]. Folgen mehrere grüne Intervalle nacheinander, so werden die IRT-Frames im Falle der Überlastung durch zu lange nRT-Frames verworfen. Dadurch wird die Synchronität der IRT-Frames bei dem übernächsten Zyklus wieder gewährleistet.

Diese Vorgehensweise ist jedoch bei einer Grenze zwischen einem grünen und roten Intervall nicht mehr möglich, da die kritischen Frames des roten Intervalls nicht verworfen werden dürfen aufgrund einer nRT-Übertragung. Zu diesem Zweck wurde zwischen einem grün-roten Intervallwechsel ein virtuelles oranges Intervall eingefügt. Dort werden nur Sendungen weitergeleitet, die vor Beginn des folgenden roten Intervalls abgearbeitet werden können. Auf diese Weise wird der feste Beginn des nächsten roten Intervalls garantiert und die Synchronität bleibt gewahrt. ProfiNet unterscheidet also zum einen zwischen einer zeitbasierten Adressierung im roten Intervall, mit welcher die Isochronität sicher gestellt wird und einer adressbasierten Kommunikation im grünen Intervall zum anderen. Während im roten Intervall anhand der Frame-IDs in Abhängigkeit der zuvor definierten Zeitslots weitergeleitet wird, erfolgt die Adressierung im grünen Intervall anhand der MAC-Adressen und nach VLAN-Prioritäten.

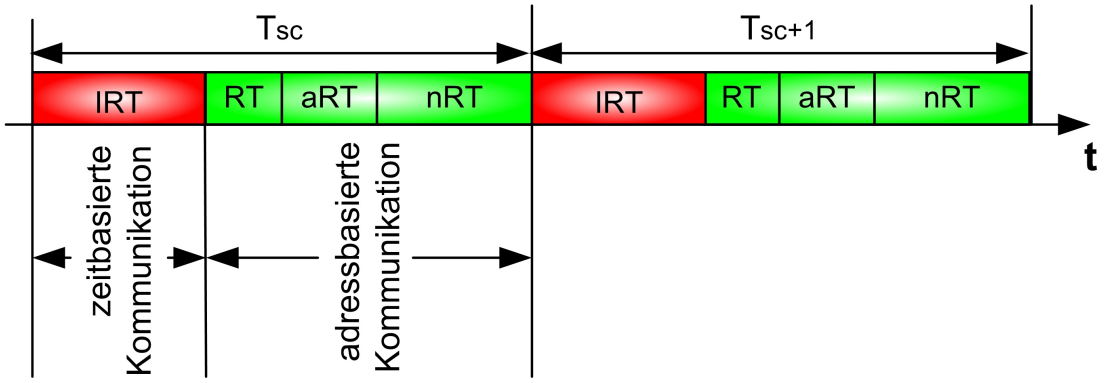


Abbildung 3.26: Kommunikationszyklus einer IRT-Schedule [Pop05]

Das grüne Intervall wird jedoch von den modifizierten Switches in der Art überwacht, dass eine Störung der roten Intervalle ausgeschlossen wird. Diese exakte Reservierung von Bandbreite in festen, sehr kurzen Zeitabständen und die Taktsynchronisation begründet also die Notwendigkeit zur Modifikation der Switching-Hardware.

Die Synchronisation in ProfiNet IRT

Jeder Zyklus von ProfiNet IRT beginnt ebenfalls mit einer Synchronisationsphase, die bei ProfiNet mit Hilfe des *Precision Transparent Clock Protocols* (PTCP) nach IEC 61158 durchgeführt wird. Dabei handelt es sich um ein modifizierte Synchronisation nach IEEE 1588, welche eine noch höhere Genauigkeit zulässt, die laut Popp [Pop05] unterhalb $1\mu s$ liegt. Dies ist nur durch eine Hardware-Unterstützung möglich, die bei ProfiNet in dem bereit gestellten ASIC integriert ist.

Das Netzwerk befindet sich zu Beginn des Anlagen-Hochlaufs zunächst in einem unsynchronisierten Zustand. Die geladenen Projektierungsdaten bestimmen in einem ersten Schritt einen *Clock-Master*, mit dem sich alle anderen IRT-Geräte des Subnetzes synchronisieren. Da das Verfahren innerhalb der Sicherungsschicht zur Anwendung kommt, kann es nicht über die Grenze des Subnetzes hinaus angewendet werden. Durch eine Broadcastsendung des Clock-Masters wird ein Sync-Impuls verbreitet, dessen Auslösezeitpunkt in einem zweiten Frame (dem *Follow-Up*) nachgeschickt wird, vgl. Kapitel 2.2.5.2. Ein Clock-Slave sendet nun einen Delay-Request Frame an den Master, der diesen mit dem Zeitpunkt des Eintreffens in einem *Delay-Response* Frame beantwortet. Mit diesen Daten kann nun sowohl die Verzögerung auf der Leitung, als auch die absolute Zeit durch die Zeitangabe innerhalb des Follow-Up Frames ermittelt werden. Das Verfahren hat Popp [Pop05] exemplarisch skizziert.

3.1.3.2 SERCOS III

Historie

Bei dem *Serial Real Time Communication System Interface* (SERCOS) [Ser07] handelt es sich um ein deutsches Industriekonsortium, dass Mitte der achziger Jahre auf Initiative des Zentralverbands Elektrotechnik- und Elektronikindustrie e.V. (ZVEI) [Zve07] und des

Vereins Deutscher Werkzeugmaschinenfabriken e. V. (VDW) [Ver07] entstand mit dem Ziel, die weitgehend analoge Antriebstechnik zu digitalisieren. SERCOS arbeitet also sehr nah auf der Feldbusebene. Zur Unterstützung existiert eine Interessensgemeinschaft SERCOS Interface e. V. mit 66 teilnehmenden Firmen.

Die erste Version des SERCOS-Interface, zu diesem Zeitpunkt noch ein Feldbus, unterstützt $2Mbit/s$ bis $4Mbit/s$ und besitzt eine weltweit hoher Akzeptanz. Das Interface wurde 1995 als IEC/EN61491 genormt. In der zweiten Version wurde 1999 die Übertragungsrate auf $8Mbit/s$ und $16Mbit/s$ erhöht und ein Service-Kanal zur Übertragung von asynchronen Daten hinzugefügt. Mit der dritten Generation des SERCOS-Interface wird nun $100Mbit/s$-Ethernet als Übertragungsmedium verwendet. Dabei verfolgt SERCOS einen völlig anderen Ansatz als ProfiNet IRT, um die härteste Echtzeitklasse der IAONA zu erreichen.

Infrastruktur

SERCOS I und II verwendeten eine *Ringtopologie* unter Verwendung von Lichtwellenleitern. Diese traditionelle Ringtopologie mit einem Master behält SERCOS ebenso bei wie die Unterteilung in einen zyklischen und einen asynchronen Kanal. Durch die Verwendung von Vollduplex-Ethernet wird der Ring durch den Rückkanal des Ethernet-Kabels gebildet, so dass nach außen hin lediglich eine Linientopologie sichtbar wird. Jedes Gerät eines SERCOS-Rings besitzt einen integrierten Switch, der zwei RJ45-Ports bereitstellt. Am Ende eines solchen Rings kann der letzte Slave an seinem freien Port zum Zwecke der asynchronen Kommunikation an einen Switch angeschlossen werden. Eine Veränderung des Netzwerkes im laufenden Betrieb ist nicht gestattet. Neben der preisgünstigen herkömmlichen Ethernet-Verkabelung können in störungsintensiven Umgebungen alternativ - wie in den vorherigen Versionen - Lichtwellenleiter verwendet werden.

Zur Redundanzerhöhung kann anstelle eines Switches das erste mit dem letzten Gerät verbunden werden, so dass sich ein Doppelring ergibt. Die Redundanzerhöhung hat in diesem Falle keine negative Auswirkung auf die Zykluszeit. Ein Kabelbruch kann erkannt und gemeldet werden.

In jedem SERCOS-Ring darf sich genau ein Master für diesen Ring befinden, die Kommunikation erfolgt also nach dem Master/Slave-Prinzip. Mehrere Ringe können miteinander verbunden werden, indem einzelne Geräte mehr als ein SERCOS-Interface besitzen. Abbildung 3.27 zeigt einen SERCOS-Doppelring in Maschinen-Modul A sowie einen einfachen Ring in Modul B. Die Mikrocontroller-Steuerungen (MC) oder die SPS-Steuerungen besitzen jeweils 2 SERCOS-Interfaces. Auf einer höheren Ebene können Geräte mit weichen Echtzeitanforderungen angebunden werden, beispielsweise über ProfiNet SRT oder EtherNet/IP.

Ein asynchron sendendes Gerät, in Abbildung 3.28 durch einen Laptop dargestellt, kann ausschließlich über ein Gateway eingebunden werden. Eine direkte Anbindung an das echtzeitkritische Netzwerk ist lediglich zu Zwecken der Konfiguration der Anlage gestattet.

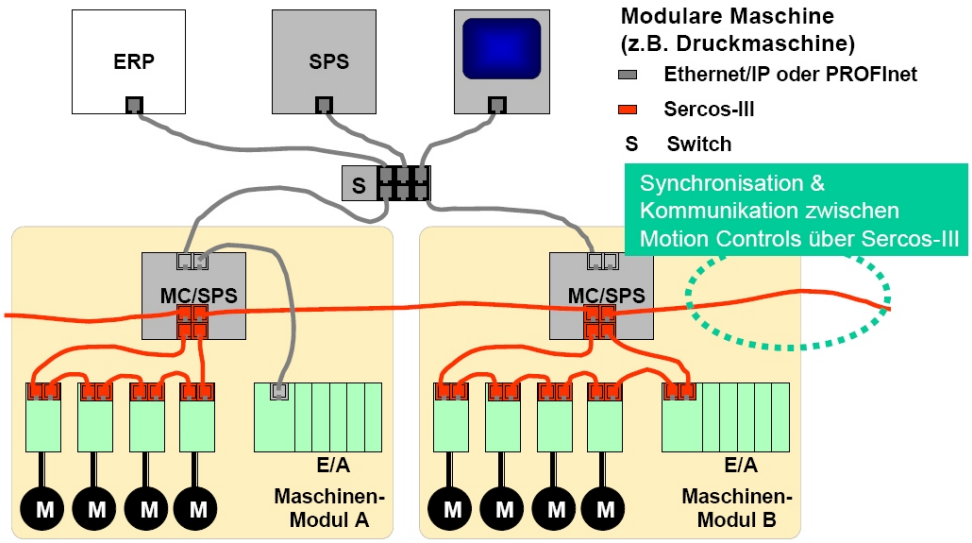


Abbildung 3.27: Modulare Maschine auf Basis von SERCOS III [Ser04]

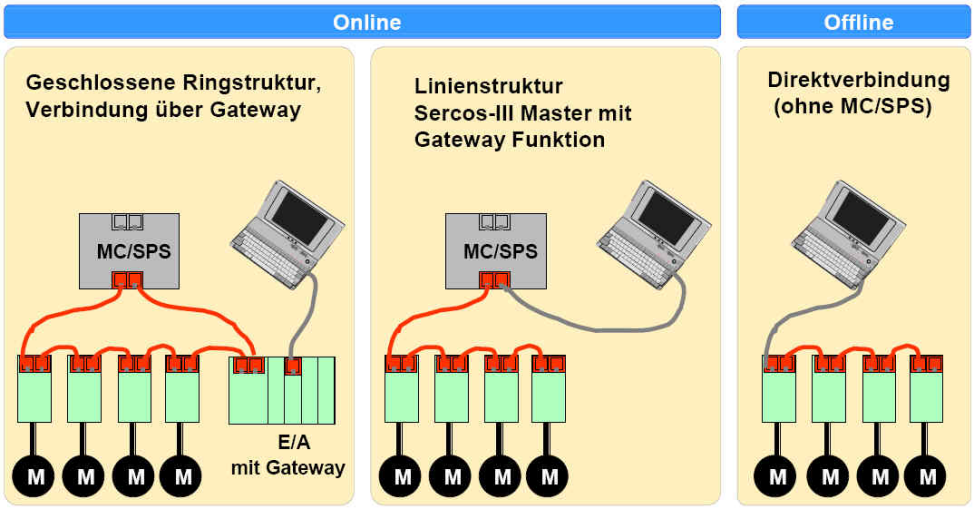


Abbildung 3.28: Anbindung eines asynchron sendendes Gerät [Ser04]

Erreichen der Echtzeitfähigkeit

Die Idee von SERCOS III besteht darin, dass ein Frame ausschließlich vom Master versendet wird und aufgrund der Ringtopologie wieder bei ihm eintrefft. Auf der Leitung befindet sich ein Standard-Ethernet Frame maximaler Größe. Jedes Gerät entnimmt den Frame für eine kurze Zeit, fügt ggf. seine zu sendenden Daten ein und modifiziert die Prüfsumme oder es liest Informationen aus. Der Frame dient also als Datencontainer für alle Geräte. Bei dem Zugriffsverfahren handelt es sich um eine Art des Token-Rings, der vom Master gesteuert wird.

Neben der zyklischen Kommunikation, die als RT-Kanal bezeichnet wird, existiert ein optionales Zeitfenster für asynchrone Übertragungen, das in der Literatur als IP-Kanal [LL05] bekannt ist. Dort werden ausschließlich Standard-Ethernet Frames übertragen, wel-

che unter anderem den TCP/IP- oder UDP/IP-Protokollstapel verwenden können. Der IP-Kanal erhält eine feste Länge in der Konfigurationsphase der Anlage. Die Umschaltung von der zyklischen Kommunikation auf den IP-Kanal erfolgt durch eine synchrone Umschaltung aller Geräte, die vom Master synchronisiert werden. Die Geräte müssen also dem SERCOS-Protokoll folgen; die Sendungen eines nicht-synchronisierten Gerätes würde zu Kollisionen führen.

SERCOS III unterscheidet zwischen vier möglichen *Master Data Telegrammen* (MDTs), vier *Acknowledge Telegrammen* (ATs) [Ost05] der Slaves und den Daten auf dem IP-Kanal. Die MDTs, welche aus einem Datenteil für jeden Slave bestehen, werden vom Master in den Ring abgesendet und von jedem Slave während des Durchlaufes empfangen. Der Datenteil beinhaltet Informationen zur Kontrolle des Slaves, z. B. neue Sollpositionen eines Motors, Service-Daten und/oder Steuerkommandos. Ein spezielles MDT ist das *Master Sync Telegramm* (MST-Frame), mit dem der Master einen Synchronisationsframe per Broadcast an die Slaves versendet, vgl. Abbildung 3.32. Dem Synchronisationsframe ist ein $6Byte$ großer *Header* (HDR) vorgeschaltet. Auch die ATs werden vom Master versendet. Es handelt sich dabei um leere Frames, die vordefinierte Felder enthalten, in die jeder Slave seine Daten direkt hinein schreibt und anschließend die Prüfsumme des Ethernet-Frames aktualisiert. Diese Daten können Statusinformationen, Servicedaten und aktuelle Nutzdaten, wie die Istposition eines Motors, enthalten. Die Anzahl und Länge der Frames werden in der Konfigurationsphase festgelegt. Aufgrund dessen, dass ausschließlich der Master sendet und die Frames einheitlich definiert sind, wird ein hoher Determinismus erreicht.

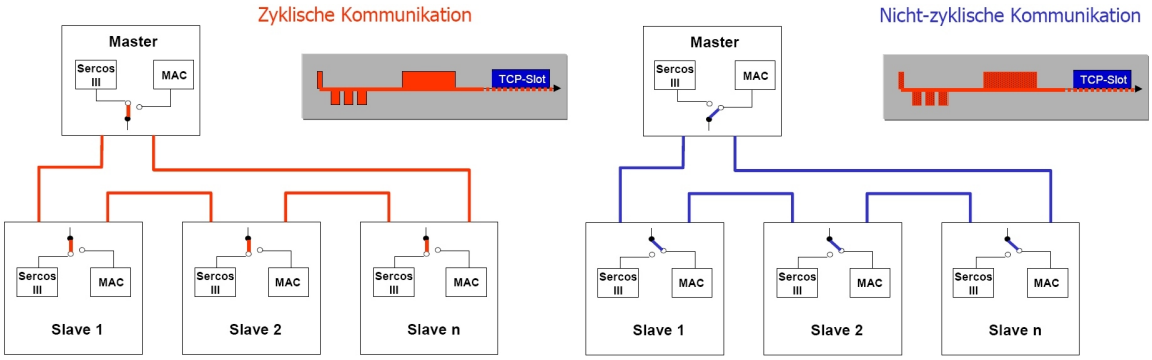


Abbildung 3.29: Umschaltung zwischen zyklischer Kommunikation und IP-Kanal [Ser04]

SERCOS III erlaubt des Weiteren Querverkehr, indem die Slaves direkt die Daten auslesen, die ein anderer Slave zuvor in den Rundsende-Frame geschrieben hat. Dies bedeutet, dass der erste Slave hinter dem Master eine hohe Priorität besitzt und im Normalfall die Master-Achse darstellt. Denn schreibt ein Slave kurz vor dem Ende der Rundsendung in den Frame, so können dessen Daten erst bei einer erneuten Rundsendung von den anderen Slaves ausgelesen werden. Dazwischen steht die Auswertung durch den Master. SERCOS III bezeichnet den Querverkehr als *Controller-to-Controller Communication* (C2C) [Ser06].

Abbildung 3.30 zeigt die zyklische Kommunikation zwischen einem Master und 3 Slave-Antrieben. Der Master besitzt zwei Interfaces bei einer Doppelring-Verkabelung.

Die Abbildung zeigt, auf welche Weise der Master zwei MDT- und AT-Telegramme überträgt. Während die MDTs von den Slaves nur ausgelesen werden, fügen die Slaves

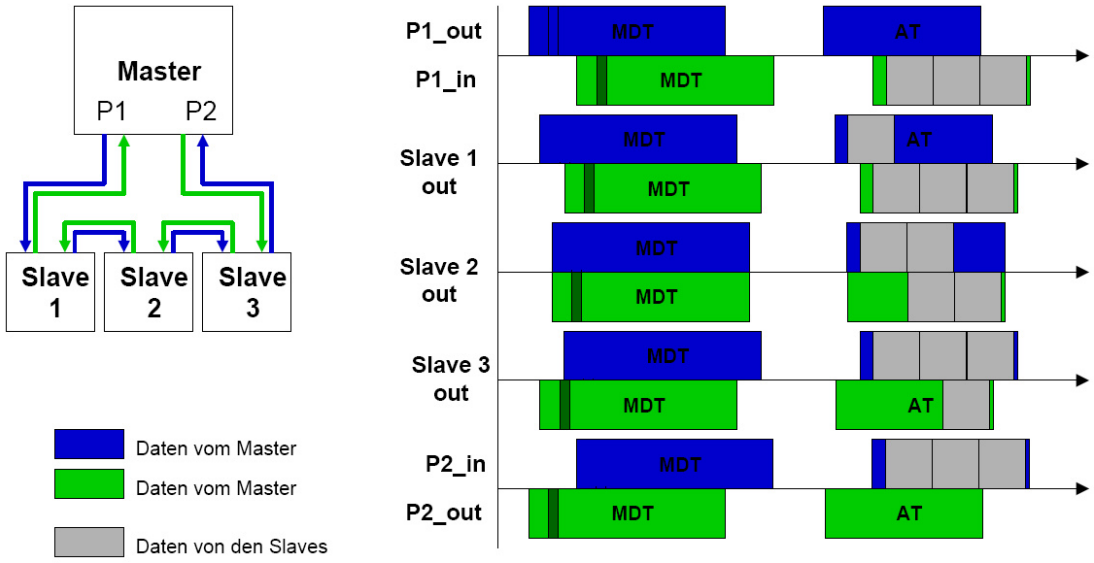


Abbildung 3.30: SERCOS III Ring und Versendung von Frames [Lut04]

ihre zu sendenden Daten in die AT-Telegramme ein. Diese können dann vom Master oder von anderen Slaves im Druchlauf des Frames entnommen werden.

Der SERCOS-Header besteht aus $6Byte$, so dass noch Nutzdaten zwischen $40Byte$ und $1494Byte$ übertragen werden können. Vordefinierte Längen der Zyklen liegen bei $31.25\mu s$, $62.5\mu s$, $125\mu s$, $250\mu s$ und Vielfachen von $250\mu s$ bis maximal $65000\mu s$. Abbildung 3.31 zeigt, dass 8 Achsen mit je $8Byte$ MDT- und AT-Daten mit einem Zyklus von $31.25\mu s$ betrieben werden können. Es ist jedoch zu bedenken, dass in diesem Falle kein asynchroner Bereich vorgesehen ist. Immerhin können 150 Antriebe mit einer Zykluszeit von $1ms$ und 50% asynchronen Datenverkehr betrieben werden. Die Synchronisation besitzt eine Abweichung unter $1\mu s$, so dass die härteste Echtzeitklasse der IAONA erfüllt wird.

Echtzeit-Daten (MDT+AT)	Zykluszeit	Antriebe
8+8 Byte	31,25 µs	8
12+12 Byte	62,5 µs	16
16+16 Byte	125 µs	30
12+12 Byte	250 µs	72
32+32 Byte	250 µs	36
12+12 Byte	500 µs	150
53+53 Byte	1 ms	100
32+32 Byte	1 ms	153
16+16 Byte	1 ms	254
40+40 Byte	2 ms	254
65+65 Byte	3 ms	254

Abbildung 3.31: Zykluszeiten von SERCOS III [Aff06]

Hardware und Software

Während die ersten beiden Versionen von SERCOS noch auf ASICs basierten, handelt es sich bei der Ethernet-Ansteuerung von SERCOS III um ein *Field Programmable Gate Array* (FPGA) [LL05], den SERCON 100. Der Code der *Very High Speed Integrated Circuit Hardware Description Language* (VHDL) wird den beteiligten Herstellern von Geräten zur Verfügung gestellt, so dass diese die SERCOS-Anbindung in ihre Geräte integrieren können. Genügt eine geringere Leistungsfähigkeit, so kann anstelle des FPGAs eine reine Software-Lösung auf Basis eines Mikrocontrollers mit zwei Ethernet-Ports zum Einsatz kommen. SERCOS bietet dazu den in C geschriebenen Quellcode an.

Auch wenn der Protokollstapel von SERCOS III erst über der Ethernet-Schicht aufsetzt, wird diese Lösung dennoch der dritten Felser-Klasse zugeordnet. Dies ist damit begründet, dass die Verkabelung, auch wenn sie physikalisch mit der Ethernet-Verkabelung identisch ist, in ihrer Anordnung nicht dem Ethernet-Standard entspricht. Die Ringtopologie unter Ausnutzung des Rückkanals der Vollduplexübertragung entspricht nicht dem Standard, auch wenn sie eine interessante Idee und einen effizienten Lösungsansatz bietet.

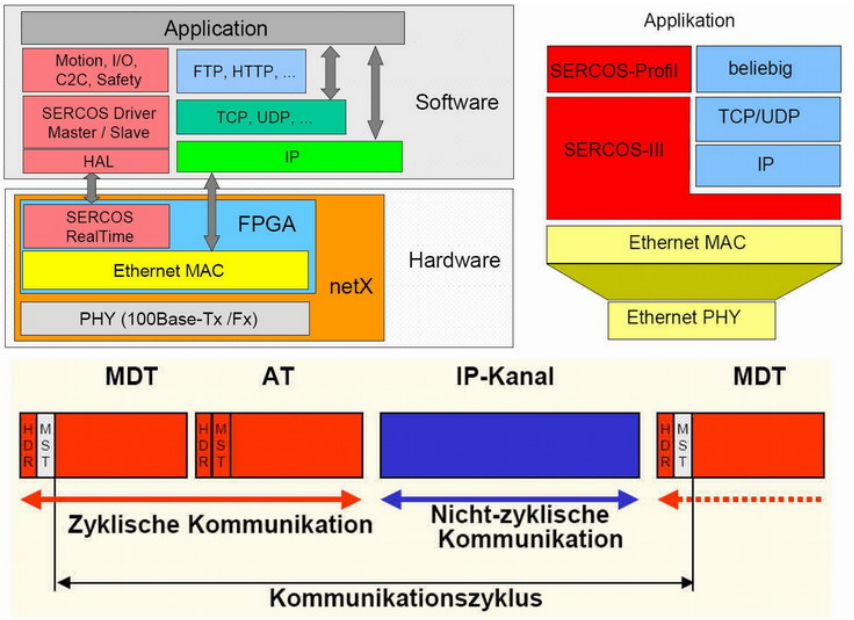


Abbildung 3.32: SERCOS III Protokoll [Aff06] [Ser04]

3.1.3.3 EtherCAT

Historie

Ethernet for Control Automation Technology *(EtherCAT)* [Eth07b] wurde 2003 vom deutschen Steuerungshersteller Beckhoff Industrie Elektronik vorgestellt, ist nach DE 101 63 342 A1, WO 03/054644 A2 und DE 103 04 637.2 patentiert und wird vertreten von der EtherCAT Technology Group (ETG), der über 600 Mitglieder angehören.

Infrastruktur

EtherCAT verfolgt einen ähnlichen Ansatz wie SERCOS III, geht jedoch noch einen Schritt weiter. Die Topologie ähnelt SERCOS III insofern, dass aufgrund der Vollduplexeigenschaft des Ethernets ein physikalischer Ring aufgebaut wird. Die Verdrahtung erfolgt normalerweise in einer Linientopologie, wobei über spezielle Verteiler[3] und/oder durch die Verwendung handelsüblicher Switches auch eine Baumtopologie ermöglicht wird. Auf diese Weise sind alle gängigen Topologien realisierbar. Zusätzlich existieren spezielle Gateways zur Anbindung an ProfiBus- oder CANopen-Subnetze. Des Weiteren existiert pro Subnetz ein einzelner Busmaster, bei dem es sich um eine beliebige Steuerung mit handelsüblichem Ethernet-Controller handeln kann. Es handelt sich also um eine Master/Slave-Architektur. In der Konfigurationsphase kann ein Master mit einem einzelnen Slave auch mittels eines herkömmlichen Ethernet CrossLink-Kabels kommunizieren.

Bei dem Master kann ein herkömmlicher Ethernet-Controller verwendet werden. Die Slaves werden mit einem eigenen EtherCAT-Controller versehen, der als ASIC oder FPGA erhältlich ist. Diese Controller sind bereits von mehreren Herstellern kostengünstig erhältlich und verfügen über ein eigenes DPRAM, in dem zu schreibende Daten von höheren Protokollebenen hinein geschrieben und aus dem Netzwerk gelesende Daten ausgelesen werden. [Sch05]

Erreichen der Echtzeitfähigkeit

Beckhoff hat erkannt, dass die Slave-Achsen der Antriebstechnik üblicherweise in einer Linientopologie angeordnet sind und eben diese Topologie bei der Verwendung von Ethernet problematisch ist. Denn in den Geräten wird jeder Frame zunächst empfangen, ggf. das FCS-Feld geprüft und die Daten in die höheren Protokollschichten weitergegeben. Im Rundsende-Verfahen von SERCOS III werden zu sendende Daten dann in diesen Frame eingefügt, die Protokollschichten nochmals durchlaufen und der Frame zum nächsten Gerät weiter versendet.

Die Idee von EtherCAT [JB04] ist es nun, die Frames bereits zu verarbeiten, während sie weiter gesendet werden. Der Master, beispielsweise eine SPS, versendet ähnlich wie SERCOS III (vgl. Abbildung 3.30) die für die Slaves relevanten Teile seines Prozessabbildes (vgl. Kapitel 2.1.1.3) in sehr großen Ethernet-Frames im Rundsende-Verfahren. Treffen die entsprechenden Daten, z. B. $4Byte$ eines Istwertes, bei einem Gerät ein, so werden diese $4Bytes$ unmittelbar bei ihrem Durchlauf ausgelesen. Sind Prozessdaten von den Slaves zu schreiben, so geschieht dies ebenfalls direkt beim Druchlauf des Frames. Zusätzlich wird der aktuelle Frame auf der Sicherungsschicht zwischengespeichert, so dass die FCS des Ethernet-Frames bei ihrem Eintreffen aktualisiert werden kann. Auf diese Weise bleibt die Prüfsumme konsistent. Das Eintreffen der zu lesenden bzw. zu schreibenden Stelle des Ethernet-Frames an dem jeweiligen Gerät kann als Token angesehen werden, das dem Gerät die Zugriffsberechtigung erteilt. Stehen Daten zum Versand an, so werden diese mit einem Zeitstempel versehen und an die Sicherungsschicht übergeben, welche die Daten bei der nächsten Rundsendung in den Frame einfügt. Empfangene Daten aktuali-

[3] Diese Verteiler werden als T-Koppler bezeichnet.

siert die Sicherungsschicht zyklisch in ihrem eigenen Speicher, der von den überliegenden Protokollen abgerufen werden kann.

Der Kern von EtherCAT besteht in der Hardware, welche Ethernet-Frames *„on-the-fly“* [Bec06] verarbeitet. Zu diesem Zweck besitzt jeder EtherCAT-Slave eine *Fieldbus Memory Management Unit* (FMMU), die den Bus abhört, die Nutzdaten und Kommandos für dieses Gerät ausliest und die zu sendenden Daten und Kommandos in den Frame einfügt. Die FMMU befindet sich in der Kopplerklemme, also im Stecker des jeweiligen Gerätes. Auf der Leitung befindet sich ein gewöhnlicher Ethernet-Frame, der mit einem Protokollanalysators wie Ethereal abgehört werden kann. Die Verarbeitung des Frames in der FMMU ist mit einem Schieberegister zu vergleichen, welches den Frame um ca. $60ns$ verzögert.

Durch den Einsatz der FMMU erhöht sich zusätzlich zu der verringerten Verzögerungszeit die Nutzdatenrate, da nicht an jedes Gerät einzelne Ethernet-Frames mit entsprechendem Overhead gesendet werden [Sch05]. Auch ein Querverkehr ist möglich, indem die Master-Achse als erster Empfänger des Ethernet-Frames der SPS ihre Werte innerhalb des Frames aktualisiert. Die nachgeschalteten Slaves können diese Daten dann ohne den Weg über den Master direkt auslesen.

Ohne nicht-echtzeitfähigen Verkehr können mit EtherCAT Zykluszeiten bis zu $30\mu s$ erreicht werden. Bei zusätzlichen fragmentierten Paketen sind immer noch Zykluszeiten unter $100\mu s$ erreichbar. Damit gehört EtherCAT zu den schnellsten momentan erhältlichen Industrial Ethernet Lösungen. Die folgende Tabelle zeigt typische Update-Zeiten.

Prozessdaten	Update-Zeit
256 verteilte digitale E/As	11 µs
1000 verteilte digitale E/As	30 µs
200 analoge E/As (je 16 Bit)	50 µs
100 Servo-Achsen (je 8 Byte Ein- und Ausgabedaten)	100 µs
1 Feldbus-Master-Gateway (1486 Byte Eingangs- und 1486 Byte Ausgangsdaten)	150 µs

Abbildung 3.33: EtherCAT-Performance [LL05]

Synchronisierung

Da der Master den Versand der großen Ethernet-Frames zentral kontrolliert, die Verzögerungen in den FMMUs relativ konstant sind und der „*Datenaustausch vollständig auf einer reinen Hardware-Maschine*“ [Sch05] basiert, ist der Jitter entsprechend gering. Der Master besitzt die Basis-Uhrzeit und führt die Uhren der Slaves durch das präzise zyklische Rundsenden der Frames nach. Aufgrund der deterministischen Infrastruktur wird dabei ein vereinfachtes IEEE 1588-Protokoll verwendet.

Die Synchronisation der Gesamtanlage kann dann z. B. unter Verwendung einer Synchronisierung nach IEEE 1588 erfolgen, welche die Master untereinander und die Geräte auf höheren Ebenen der Pyramide der Automatisierungstechnik abgleichen.

Aufgrund der Infrastruktur können die verteilten Uhren in einem EtherCAT-Subnetz auf eine Abweichung weit unterhalb $1\mu s$ eingestellt werden. Ein im IAONA-Handbuch [LL05] angegebenes Beispiel mit 300 Slaves und $120m$ Kabel erlaubt eine Genauigkeit, die im Bereich von $20ns$ liegt. Die technischen Daten von EtherCAT sind damit vielen konventionellen Feldbussen überlegen.

Das EtherCAT-Protokoll und asynchrone Sendungen

Innerhalb eines Subnetzes aus EtherCAT-Geräten wird ein Ethernet-Frame mit eigenem EtherType verwendet. In einem solchen Frame können mehrere Kommandos eingebettet werden, die individuelle Geräte und/oder Speicherbereiche ansprechen. Ein EtherCAT-Kommando besteht aus einem Header, einem Datenbereich und einem folgenden *Working Counter* (WC), der von jedem Gerät inkrementiert wird, das angesprochen wurde und Daten ausgetauscht hat. Das Protokoll sowie die EtherCAT-interne Adressierung wurden von Janssen und Büttner [JB03b] beschrieben.

Es ist jedoch auch möglich, EtherCAT-Rundsendungen über Router hinweg zu versenden. Dazu ergänzt der Master den Ethernet-Frame um einen UDP/IP-Header. Diesem folgt dann die übliche Struktur der eingebetteten EtherCAT-Telegramme, wie in Abbildung 3.34 dargestellt ist.

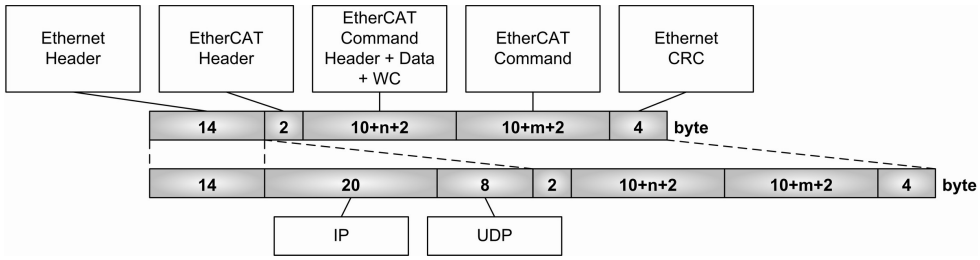


Abbildung 3.34: EtherCAT Header und optionale UDP/IP-Erweiterung [JB03]

Mit dem UDP/IP-Header erhält man einen größeren Overhead. Dieser Header wird von den EtherCAT-Slaves nicht interpretiert. Im Gegenzug erhält man Pakete, die über Router hinweg versendet werden können. Dies ist für weichere Echtzeitanforderungen immer noch ausreichend. Eine weitere Verschlechterung der Echtzeitfähigkeit bringt der Einsatz von Standard-Switches mit sich, da selbst cut-through Switches die Frames wesentlich länger verzögern als ein EtherCAT-Slave. Ebenso erhöht sich der Jitter. Der Einsatz von Standard-Hardware ist also prinzipiell möglich und ist abhängig von dem konkreten Anwendungsfall, in dem das Netzwerk eingesetzt wird. Auf diese Weise wird zwar die hohe Performance von EtherCAT gedrosselt, die Integration in bestehende Netzwerkinfrastrukturen jedoch erhöht.

Eine weitere Integration von Standard-Hardware ist durch den Einsatz von Gateways gegeben, über die beispielsweise ein Standard-Laptop über ein EtcherCAT-Netz hinweg auf einen externen Web-Server zugreifen kann. Die Gateways kapseln die Kommunikation des Laptops und tunneln dessen Frames innerhalb des EtherCAT-Protokolls. Dazu muss bei der Konfiguration des Netzes ein freier Bereich in jedem Rundsende-Frame reserviert werden. Das Gateway fragmentiert nun die Frames des Laptops und fügt die Fragmente

in die Rundsende-Frames ein. Das EtherCAT-Gateway auf der anderen Seite des echtzeitfähigen Subnetzes reassembliert die Fragmente wiederum auf der Ethernet-Schicht. Auf diese Weise wird die Echtzeitfähigkeit des Netzes nicht beeinträchtigt. [Sch05]

Für asynchrone Daten bietet EtherCAT innerhalb seiner Sicherungsschicht ein sogenanntes Mailbox-System an. Dieses gibt gültige Ethernet-Frames an eine übergeordnete Schicht - wahlweise an einen TCP/UDP/IP- oder CANopen-Protokollstapel - weiter, so dass dort eine transparente Weiterverarbeitung statt finden kann. Abbildung 3.35 zeigt die Mailbox-Schnittstelle auf der Sicherungsschicht mit den Schnittstellen FileSystem over EtherCAT (FoE), Ethernet over EtherCAT (EoE), Servodrive over EtherCAT (SoE) sowie CANopen over EtherCAT (CoE).

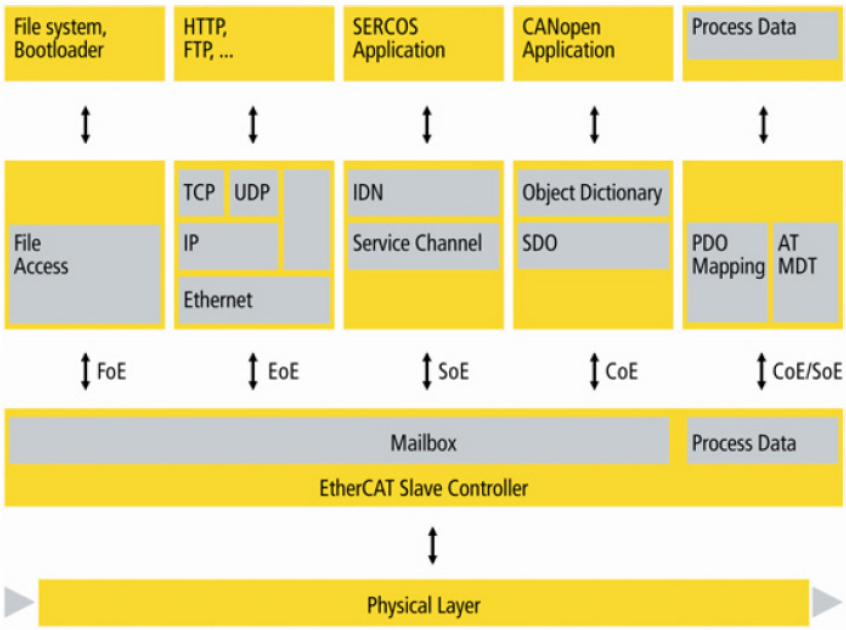


Abbildung 3.35: EtherCAT Protokollstapel [Eth04]

3.1.3.4 SynqNet

Historie

Einen ähnlichen Ansatz wie SERCOS III und EtherCAT stellt *SynqNet* [Syn07] dar, das in Europa jedoch kaum verbreitet ist. SynqNet wurde vom kalifornischen Steuerungshersteller Motion Engineering, Inc. in Zusammenarbeit mit Antriebsherstellern wie z.B. Advanced Motion Controls, Danaher Motion, Glentek, Panasonic und Sanyo entwickelt und ist vor allem in den USA, Japan und Korea verbreitet. Die SynqNet User Group, der unter anderem die oben genannten Konzerne angehören, besteht aus gleichberechtigt agierenden Partnern.

Infrastruktur

SynqNet setzt auf der Bitübertragungsschicht des Ethernets nach IEEE 802.3 auf; es kann eine Linien- oder Ringtopologie aufgebaut werden. Die Verkabelung erfolgt mit RJ45-Steckern und CAT-5 Kabel. Ähnlich wie SERCOS III und EtherCAT besitzt ein SynqNet-Gerät ein Interface mit zwei Ports, mit dem eine Linien- oder eine physikalische Ringtopologie durch Verbindung des letzten Gerätes mit dem ersten Gerät aufgebaut werden kann. Der Ring dient der Einführung von Redundanz, die Kabelbrüche ermittelt. SynqNet benutzt die Vollduplexfähigkeit des Netzes zur Verdoppelung der effektiven Bandbreite. Ähnlich zu den bereits vorgestellten Ansätzen wird bei SynqNet die Redundanz durch den Aufbau einer physikalischen Ring-Struktur (backup channel) erreicht.

Eigene Hubs oder Switches werden auch hier nicht verwendet; es sind 254 Geräte in einem Subnetz möglich. Im Gegensatz zu EtherCAT konzentriert sich SynqNet auf die Minimierung der minimalen Framegröße und des Overheads. Da die MAC-Schicht des Ethernets ausgetauscht wird, kann eine Reduzierung des minimalen Ethernet-Frames von $74Byte$ inclusive Präambel auf $24Byte$ vorgenommen werden, es findet kein Padding statt. Eine Kommunikation über TCP/UDP/IP ist in einem SyncNet-Subnetz nicht erlaubt. Lediglich der Master, der bei SynqNet als Controller bezeichnet wird, kann über eine separate Netzwerkkarte angesprochen werden. Der Controller übernimmt dadurch die Rolle einer Bridge, über die das Subnetz konfiguriert, diagnostiziert und verwaltet werden kann.

Hardwaretechnisch wird im Gegensatz zu EtherCAT kein eigener ASIC verwendet, sondern ein Dual-PHY, der die physikalische Schicht des Ethernets interpretiert und von verschiedenen Herstellern verfügbar ist. Der Frame wird im Gegensatz zu EtherCAT vollständig vom Medium abgegriffen und interpretiert. Dies ähnelt der Vorgehensweise von SERCOS III, vgl. Abbildung 3.30.

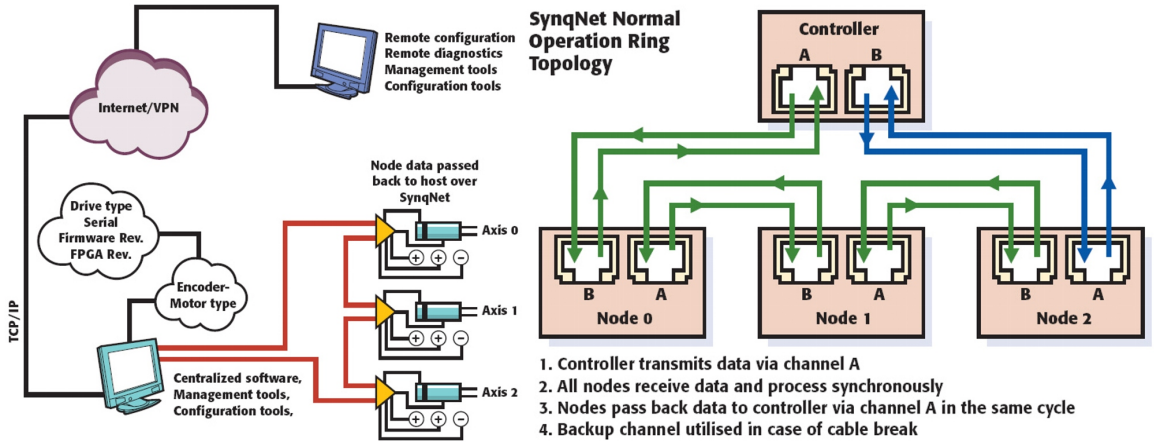


Abbildung 3.36: SynqNet-Infrastruktur [Mat04]

Erreichen der Echtzeitfähigkeit

Das Format des veränderten SynqNet-Frames konnte ebensowenig ermittelt werden wie die Art der Synchronisation. Es ist jedoch anzunehmen, dass ein ähnliches Prinzip wie bei SERCOS III angewendet wird. Die eigene SynqNet MAC-Schicht kann in einen preisgünstigen FPGA implementiert werden, so dass SynqNet ausschließlich auf frei verfügbaren elektronischen Komponenten basiert. Das Application Programming Interface

(API) besteht wahlweise aus einer Sammlung von C/C++ Bibliotheken oder Active-X Bibliotheken.

SynqNet gibt eine minimale Zykluszeit von $< 25\mu s$ an für die Steuerung von vier Achsen [Mat04]. Der Jitter soll dabei stets unter $1\mu s$ liegen, wodurch die härteste vierte Echtzeitklasse der IAONA erfüllt wird.

SynqNet stellt ein gutes Beispiel dafür dar, wie eine höhere Echtzeitfähigkeit die Kompatibilität zu Standard-Ethernet verringern kann. Mit einer Reduzierung der minimalen Framelänge und dem Aufheben des Paddings sowie einer Verringerung des Protokoll-Overheads ist in geschlossenen Netzen eine Erhöhung der Performance leicht erreichbar. Das Resultat ist jedoch ein proprietäres Netzwerk. Die Vorzüge des Ethernet-Standards, wie die Analyse der Datenströme mittels eines Protokollanalysators, sind in diesem Falle nicht mehr einsetzbar. WireShark würde die versendeten Frames, ebenso wie eine Vielzahl von COTS-Switches, als defekt ansehen und verwerfen.

3.1.3.5 GinLink

Historie

Abschließend wird ein Ansatz der Firma Indel aus der Schweiz vorgestellt, der seit 2005 existiert und deren Hardware in den ersten Versionen verfügbar ist. Indel geht mit *GinLink* [Ind04] noch einen Schritt weiter als SynqNet und verwendet einen eigenen Protokollstapel bis hin zur physikalischen Schicht. Wie dieser Ansatz in Bezug zu echtzeitfähigem Ethernet steht, wird im Folgenden skizziert.

Infrastruktur

Als Übertragungsmedium verwendet GinLink einen Lichtwellenleiter, der von Hand konfektionierbar ist und eine störsichere Übertragung bis zu $500m$ gewährleistet mit einer Übertragungsrate von $1Gbit/s$. Dieser Lichtwellenleiter bildet einen geschlossenen Bus, der über Gateways zugänglich ist. Die an die Gateways angeschlossenen Geräte werden in einer Ringtopologie angeordnet und entnehmen die Telegramme im Durchlauf bzw. fügen Daten ein. Die Framerate beträgt dabei maximal $40kHz$. Eines der Geräte stellt den Master dar, der die Frames verwaltet.

Erreichen der Echtzeitfähigkeit

Wie bei EtherCAT werden die GinLink-Frames im Durchlauf verarbeitet. Bei diesen Frames handelt es sich jedoch nicht um Ethernet-Frames, sondern um ein eigenes Format. Die Frameübertragung wird in Zeitslots organisiert, wobei jeder Slot eine Dauer von $12.5\mu s$ besitzt. Acht Slots bilden einen Zyklus. In jedem Slot kann genau ein Ethernet-Frame maximaler Größe von $1536Byte$ übertragen werden.

Jeder zweite Slot wird als High-Speed Frame bezeichnet. Diese werden in einem Zyklus viermal übertragen, so dass eine Framerate von $40kHz$ resultiert. In diesen Frames können Daten mit hohen Echtzeitanforderungen übertragen werden, pro Slot entweder die Steuerdaten von 180 Achsen mit jeweils 32 Bit Soll- und Istwerten oder 720 analogen E/A-Werten. Abbildung 3.37 zeigt einen GinLink-Frame und ein GinLink-Gerät, welches das

Gateway zu dem internen Bus darstellt. Neben den Anbindungen zu Industrial Ethernet wird eine Schnittstelle nach IEEE 1394b FireWire angeboten. Details des Protokolls und der Synchronisierung von GinLink konnten nicht ermittelt werden, da sich dieses System noch im Prototyp-Stadium befindet und nur rudimentär dokumentiert ist.

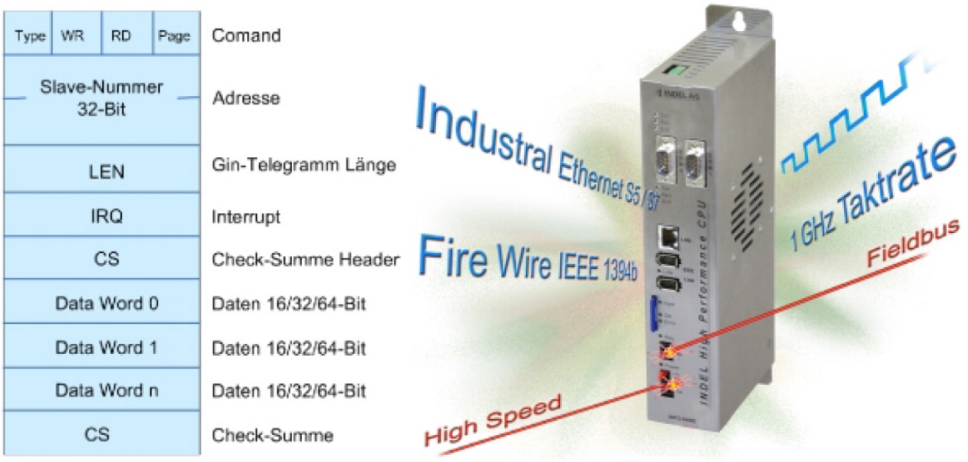


Abbildung 3.37: GinLink-Frame und -Gateway [Ind04]

Je nach Größe kann eine unterschiedliche Anzahl an GinLink-Frames innerhalb eines High-Speed Frames übertragen werden. Abbildung 3.38 zeigt die Einteilung der 8 Slots, bei dem sich jeweils ein High-Speed Frame und ein Low-Speed Frame abwechseln. Für die Low-Speed Frames ergibt sich dabei eine Framerate von $10kHz$, die sich für die Anbindung von langsameren Geräten eignet. Möchte man die Zykluszeit minimieren, so können die Achsen in jedem High-Speed Frame und die E/A-Baugruppen in jedem Low-Speed Frame aktualisiert werden. Dies ermöglicht eine Zykluszeit von $25\mu s$.

Der letzte High-Speed Slot ist in Abbildung 3.38 exemplarisch detailliert dargestellt. Jeder High-Speed Slot ist ein gültiges UDP-Paket mit unterlagertem IP- und Ethernet-Protokoll, das mit einer üblichen CRC-Prüfsumme abgeschlossen wird.

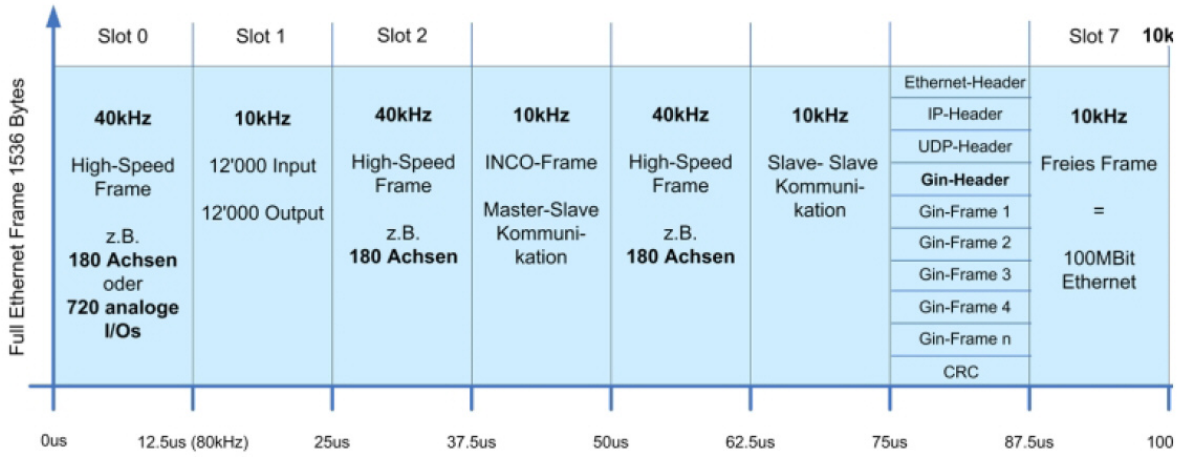


Abbildung 3.38: GinLink-Slots [Ind04]

Jeder achte Frame ist für die Übertragung eines herkömmlichen Ethernet-Frames reserviert, der über das Gateway eingespeist wird. Auf diese Weise kann asynchroner Verkehr einbezogen werden. Als weiteren Vorteil sieht Indel, dass in dem letzten Slot natürlich auch ein Ethernet PowerLink Frame oder ein EtherCAT Frame über GinLink versendet werden kann.

Bei den proprietären GinLink-Gateways handelt es sich um PowerPC 750-FX RISC-CPUs, die mit bis zu $256 MByte$ SDRAM ausgestattet werden können. Das Betriebssystem ist lizenzfrei und basiert auf der Programmiersprache C/C++. Aufgrund der anspruchsvollen Hardware wird mit einem hohen Preis pro Gateway von $2500 CHF$ gerechnet.

3.2 Vergleich und Bewertung der bestehenden Ansätze

Abbildung 3.39 zeigt eine Übersicht der vorgestellten Lösungen. Der Foundation H1-Bus ist separat zu betrachten, da es sich dabei um einen reinen Feldbus handelt. Er wurde in den Vergleich mit aufgenommen, um in Kombination mit HSE eine alternative Lösungsstrategie aufzuzeigen. In diesem Kapitel werden die existierenden Lösungen nun miteinander verglichen.

3.2.1 Einführung der Echtzeit und Übertragungszeiten

Wird die Echtzeitfähigkeit über der Transportschicht eingefügt, wie bei EtherNet/IP oder bei Fondation HSE, so kann die härteste vierte IOANA-Echtzeitklasse aufgrund der Overheads in der TCP- und IP-Schicht und dem nach wie vor existierenden, nicht-deterministischen CSMA/CD des Ethernets nicht erreicht werden. Neben einer hohen, nicht-deterministischen Verzögerungszeit und Jitter des Netzwerkes existiert eine zusätzliche Verzögerung und zusätzlicher Jitter aus dem Betriebssystem, welches die höheren Schichten des Protokollstapels interpretiert.

Jedoch ist auch das Einsatzziel eines Systems über der Transportschicht ein anderes. EtherNet/IP wird ebenso wie der Foundation Fieldbus HSE meist zu Management- oder Diagnosezwecken eingesetzt und ist der Leitebene der Pyramide der Automatisierungstechnik zuzuordnen. Hier ist ein Trend zur Objektorientierung mit Funktionsblöcken zu beobachten, welche den Vorteil der Wiederverwendbarkeit und Vereinfachung bieten. Der Austausch der Parametrierung unter den Geräten erfolgt zumeist in der Konfigurationsphase des Netzwerkes unter Verwendung von XML-Dateien (Extensible Markup Language). In diesen Fällen bildet also ein leicht verändertes Ethernet, beispielsweise durch die Einführung von VLAN-Prioritäten, eine Schnittstelle zwischen dem herkömmlichen Ethernet auf den höheren und den Feldbussen auf den unteren Schichten der Automatisierungspyramide. EtherNet/IP erlaubt beispielsweise die Anbindung von Feldbussen wie ControlNet oder DeviceNet. Der Datenfluß mit weichen Echtzeitkriterien lässt sich parallel zu asynchronen Daten verarbeiten, sofern das Netzwerk nicht überlastet ist. Insbesondere die Lösung der Fieldbus Foundation zeigt, wie Ethernet als zusätzliches Netzwerk auf Managementebene den Feldbus H1 ergänzen kann.

Auf der Feldbusebene zählt hingegen allein eine minimale Übertragungsdauer mit geringstem Jitter. Auf dieser Ebene setzen die Technologien an, welche oberhalb der Sicherungsschicht des OSI-Modells eingeführt werden und der zweiten Felser-Klasse ent-

		Ethernet/IP + CIPsync	Foundation H1	Foundation HSE	EPL open[3]	EPL protected	TCnet	EPA	RTnet	Profinet CBA/SRT	Profinet IO/IRT	SERCOS	EtherCAT	SynqNet	GinLink
		3.1.1.1	3.1.1.2		3.1.2.1		3.1.2.2	3.1.2.3	3.1.2.4	3.1.2.5	3.1.3.1	3.1.3.2	3.1.3.3	3.1.3.4	3.1.3.5
Einführung der Echtzeit															
Echtzeit über Transportschicht		●	-	●	-		-	-	-	-		-	-	-	-
Echtzeit über ETH-MAC		-	-		●		●	●	●	●	-	-	-	-	-
Modifizierte MAC		-	-		-		-	-	-	-	●	●	●	●	●
Eigene Bitübertragungsschicht		-	●	-	-		-	-	-	-		-	-	-	●
Übertragungszeiten															
IAONA-Echtzeitklasse		3	3	2	3	4	3	3	3	3	4	4	4	4	4
minimale Zykluszeit [s]		1m	20m	>100m	<10m	200µ	1m	<10m	2.5m	4m[5]	31µ	31µ	30µ	25µ	25µ
Obergrenze des Jitters [s]		<10µ	-	>1m	<100µ	<1µ	<10µ	<100µ	<40µ	<50µ	<1µ	<1µ	<1µ	<1µ	<1µ
Strategie und Synchronisation															
Einführung von TDMA		-	●	-	●		●	●	●	-	●	●	-	●	●
Einführung von Token oder Token-Aspekt		-	●	-	-		●	●	●	-		●	●	●	●
Einführung einer Priorisierung		●	-		●	-	●	-	●	●	●[2]	●	-	-	-
Polling der Geräte		-	●	-	●		-	●	-	-		-	-	-	-
Frames im Durchlauf bearbeiten		-	-		-		-	-	-	-		-	●	-	●
Master vorhanden		-	●	-	●		●	●	●	-		●	●	●	●
Synchronisation über Sync-Signal		-	●		●		●	-	-	-		●	-	?	-
Zeitsynchronisation (z.B. IEEE1588/PTCP)		●	-		●		-	●	●	●		-	●	?	-
Topologie & Verteiler															
logische Topologie:	Baum	●	●		-		-	-	●	●		-	-	-	-
	Linie/Bus	-	●	-	●		●	●	●	-		-	-	-	-
	Ring	-	-	-	-		-	-	-	-		●	●	●	●
physische Verkabelung:	Baum	●	●		●		●	●	●	●		-	●	-	-
	Linie/Bus	-	●	-	●		-	-	-	●		●	●	●	-
	Ring	-	-	-	-		-	-	-	-	●	●	●	●	●
Verteiler:	Hubs	●	-	●	●		●[4]	●	●	-		●[4]	-	-	-
	Switches	●	-	●	●	-	-	-	●	-	●[4]	-	●	-	-
	eigene	-	●	-	-		●	-	-	●		●	●	●	●
Verteiler-Integration in die Geräte möglich		-	-		●		-	-	-	●		●	●	●	-
Nähe zu Standards															
Standard-Frames auf der Leitung		●	-	●	●		●	●	●	●		●	●	-	-
Ethernet-Standard oberhalb von Schicht 2		●	-	●	●		●	●	●	●		●	●	●	●
Verkabelung entspricht Ethernet-Standard		●	-	●	●		-	●	●	●		●	●	●	-
Verteiler entsprechen Ethernet-Standard		●	-	●	●		-	●	●	●	-	●	-[8]	-	-
TCP/UDP/IP-Pakete im Subnetz erlaubt		●	-	●	●		●	●[7]	-	●	●	●	-	-	●
nRT-Traffic in bestimmten Zeit-Slots		●	-		●		●	-	-	-	●	●	-	-	●
Standard-PCs im Subnetz erlaubt		●	-	●	●	-	-	-	-	●		-	-	-	-
Querverkehr erlaubt		●	-	●	-		-	-	-	●		●[6]	●[6]	●[6]	●[6]
EtherType		0x0800	-	0x0800	0x88AB		0x888B	0x88CB	-	0x8892		0x88CD	0x88A4	-	-

1: Rückleitung ist 2. Kanal des VD-Ethernet
2: im untergelagerten ProfiNet CBA/SRT
3: geplant
4: modifiziert
5: geringere Zeiten sind möglich bei zusätzlicher Verwendung von ProfiNet IO/IRT
6: Querverkehr nur im Rahmen einer Rundsendung
7: nach vorheriger Anmeldung
8: für eine Baum-Verkabelung können jedoch handelsübliche Switches eingesetzt werden

Abbildung 3.39: Bewertung der vorgestellten Echtzeit-Ethernet Ansätze

sprechen. Als bekannteste Vertreter sind hier EPL und ProfiNet SRT zu nennen. Diese Ansätze sind von ihrer Idee her ähnlich, indem sie das CSMA/CD um ein deterministisches Zugriffsverfahren ergänzen. Dies erfordert jedoch einen tieferen Eingriff in die Protokoll-Hierarchie.

Die Ansätze mit modifizierter MAC oder sogar mit eigener Bitübertragungsschicht wie GinLink erreichen mit heutiger Technologie alle die härteste IAONA-Echtzeitklasse und sind damit in der Antriebstechnik einsetzbar. Die Zeitverzögerung der Pakete und deren Jitter sind so gering, dass sie mit neuesten Feldbussen konkurrieren und diese ersetzen können. Der Fokus auf härteste Echtzeitanforderungen bringt jedoch weitere Einbußen in der Kompatibilität mit sich. Dies führt im Falle von GinLink sogar zu einer vollständig veränderten Busphysik.

3.2.2 Strategie und Synchronisation

Zur Erreichung weicher Echtzeitfähigkeit wird sowohl bei EtherNet/IP, als auch bei TCnet und ProfiNet SRT eine Priorisierung der Frames verwendet. EtherNet/IP und ProfiNet SRT verwenden dabei VLAN-Tags, während TCnet eine eigene Priorisierung verwendet. Hier steuert DOMA die Verwaltung der Prioritäten. Auch wenn durch die Einführung von CIPsync der exakte Sendezeitpunkt bei EtherNet/IP bekannt ist, kann es sein, dass die betreffenden Informationen aufgrund von übervollen Warteschlangen von hochprioren Frames zu spät eintreffen oder gar verworfen werden. Aus diesem Grunde gibt ProfiNet Lastgrenzen an, in denen ein sicherer Betrieb unter Verwendung der Priorisierungen ermöglicht wird.

Die Idee der Einführung von Prioritäten im Umfeld von echtzeitfähigen Ethernet verwendet auch Jasperneite in seiner Dissertation mit dem Thema „Leistungsbewertung eines lokalen Netzwerkes mit Class-of-Service Unterstützung für die prozessnahe Echtzeitkommunikation“ [Jas02]. Die existierenden Lösungen deuten jedoch darauf hin, dass mit reinen prioritätsbasierten Verfahren keine harten Echtzeitanforderungen erfüllt werden können.

EPL im Protected Mode ist die einzige Lösung, die unter Verwendung von Standard-Hardware die härteste IAONA-Echtzeitklasse erreicht. Bei allen Ansätzen, die auf Hubs basieren (EPL, TCnet, EPA und RTnet), kann ein zentrales Scheduling leicht errechnet werden, da nur eine Kommunikation zu einem Zeitpunkt auf dem Netzwerk erlaubt ist. Der jeweilige Master pollt nacheinander die Geräte, die im darauf folgenden Zeitslot senden dürfen. Die Schedule kann in der Konfigurationsphase der Anlage berechnet werden, da die Anforderungen der Geräte im Rahmen der Automatisierungstechnik im Vorfeld bekannt sind. Es kann also eine Offline-Schedule erstellt werden. Für kritische Ereignisse, wie die Betätigung eines Not-Aus Schalters, wird ein entsprechender Zeitslot reserviert.

RTnet zeigt die Kapselung des Mediums über der Sicherungsschicht durch die Einführung einer eigenen TDMA-Zugriffskontrolle. Hier zeigt sich jedoch, dass ein reiner Software-Ansatz betriebssystemabhängig ist und generell nicht die Anforderungen der härtesten Echtzeitklasse erfüllen kann. Die Messergebnisse von RTnet zeigen die Grenzen einer software-basierten Lösung auf. Gleichzeitig ist RTnet der einzige Ansatz der zweiten Felser-Klasse, der ein optionales Token-Verfahren anbietet.

Ein Teil eines TDMA-Zyklus ist zumeist für asynchrone Datenübertragung reserviert. Das Problem liegt hier in der maximalen Ethernet-Framelänge. Möchte ein Gerät einen maximal langen Frame abschicken, so wird die Zykluszeit des gesamten Systems stark erhöht. Möglichkeiten zur Lösung bestehen darin, solche Frames nicht zuzulassen oder die MTU des Netzwerkes herabzusetzen, wodurch eine Fragmentierung und Reassemblierung der Pakete notwendig wird. Geht man von der minimalen Paketgröße des Ethernets aus, so würde der Nutzdatenanteil eines asynchronen Paketes im Echtzeit-Netzwerk sehr gering sein. Außerdem müssen alle Geräte des Subnetzes dem TDMA-Protokoll exakt folgen. Ein einzelnes asynchron sendendes Gerät zerstört den Determinismus des Systems. EPL fällt in diesem Fall in den Open Mode zurück, der ähnliche Leistungsdaten wie EtherNet/IP aufweist. Damit ist das Netzwerk nicht mehr für die Antriebstechnik tauglich, was zu einem Ausfall der Anlage führen würde. Denn man muss davon ausgehen, dass ein im Protected Mode arbeitendes Netzwerk auch auf diesen Modus angewiesen ist und im laufenden Betrieb Achsen ansteuert.

Neben der Priorisierung und der Einführung von TDMA existiert eine dritte Klasse von Lösungen wie SERCOS III, EtherCAT, SynqNet und GinLink, die Frames im Rundsende-Verfahren verarbeiten. Bei SERCOS III versendet der Master leere Frames, die von dem entsprechenden Gerät gefüllt und weiter versendet werden. Auf diese Weise wird Querverkehr mit einer Masterachse - die das erste Gerät im Ring darstellt - ermöglicht. EtherCAT verwendet einen einzelnen, sehr großen Frame, in den jedes Gerät auf der Basis der Anlagenkonfiguration seine Prozessdaten während des Durchlaufs schreibt bzw. ausliest. Die Position des Frames, an welcher ein Gerät seine Informationen in den Frame schreibt, kann als Sendeberechtigung und damit als Token aufgefasst werden. Auf diese Weise wird auch die direkte Kommunikation der Slaves untereinander - also der Querverkehr - gesteuert. Sendet ein einzelnes Gerät ohne Beachtung dieses Tokens asynchron, so wird auch hier die Echtzeitfähigkeit des Netzes beeinträchtigt. Denn aufgrund dieser Sendung wird ein weiterer Frame auf dem Medium erzeugt, welcher Einfluss auf den EtherCAT-Frame mit dessen Echtzeitanforderungen nimmt. Dies kann durch die Gateway-Funktion der FMMU-Klemme verhindert bzw. kontrolliert werden.

ProfiNet IRT ist die einzige Technologie, welche mit einer dezentralen Schedule ohne Master arbeitet und mehrere gleichzeitige, unabhängige IRT-Übertragungen durch Verwendung von modifizierten Switches zulässt. Dabei besitzt jeder Switch eine eigene Schedule, die während der Hochlaufphase der Anlage geladen wird. Die Berechnung erfolgt durch ein Konfigurationswerkzeug von Siemens, dessen Algorithmen ebenso wie die Hardware proprietär sind. Interessant ist die Unterscheidung der Schedule in eine grüne, orange und rote Phase, wodurch die Übertragung möglichst vieler asynchroner Frames ermöglicht wird, ohne den isochronen Echtzeitverkehr zu stören. Die SRT-Version von ProfiNet erreicht aufgrund der Verwendung von Standard-Hardware nur die dritte IAONA-Klasse.

Zur Zeitsynchronisation haben sich im Allgemeinen Protokolle nach IEEE 1588 durchgesetzt. Wird der Absendezeitpunkt von Frames in den unteren Protokollebenen nahe am Medium gemessen, so kommt wie bei CIPsync, EPL und EPA ein preisgünstiger Synchronisationsbaustein zum Einsatz. Dies erfordert jedoch eine wiederum Modifizierung der Hardware. ProfiNet IRT verwendet mit PTCP eine leicht veränderte Synchronisation, welche in den ProfiNet-ASIC integriert ist. Alternativ dazu kann eine Synchronisierung nach IEEE 1588 auch softwareseitig erfolgen, wodurch jedoch der Jitter erhöht wird. Eine Software-Synchronisierung wird beispielsweise bei EPA, RTnet und ProfiNet SRT verwendet.

Bei den Lösungen mit einem zentralen Master besitzt dieser eine hochgenaue Zeitbasis, mit denen die Uhren in den Slave-Geräten abgeglichen werden. Bei EPL und TCnet geschieht dies durch die Verwendung eines als Broadcast versendeten Start-Frames (SoC bei EPL bzw. SYN bei TCnet). Dieser Frame dient der Einleitung eines neuen Bus-Zyklus und gleichzeitig zur Synchronisation der Uhren. Bei ProfiNet IRT sind hingegen die Switches untereinander synchronisiert, um die Datenflüsse kontrolliert weiterzuleiten. Hier erfolgt die Synchronisation ebenfalls zu Beginn jedes Zyklus.

3.2.3 Topologie und Nähe zu Standards

Da die Echtzeitfähigkeit von Lösungen nach der ersten Felser-Klasse erst über der Transportschicht eingeführt wird, arbeiten Netze wie EtherNet/IP oder Foundation HSE mit

dem standardisierten Ethernet-Protokollstapel. Es wird ein vollduplexfähiges $100Mbit/s$-Netzwerk mit Baumtopologie verwendet. Standard-PCs im gleichen Subnetz beeinträchtigen IRT-Frames bei hoher Netzlast, wenn die Puffergrenzen der Switches erreicht sind. Liegt eine Standard-Infrastruktur zugrunde, so wird oft für echtzeitkritische Pakete eine Priorisierung vorgenommen, meist nach dem verbreiteten Standard IEEE 802.1p. Dieser Standard wird von den gängigen Switches unterstützt. Ein Problem entsteht, sobald Last aus mehreren hochprioren Datenströmen entsteht. In diesem Fall erhöht sich die Laufzeitverzögerung durch Pufferung in den Switches, es können Frames sogar verworfen werden.

Der Ansatz der Fieldbus Foundation zeigt repräsentativ für viele Ansätze, dass an der Schnittstelle zu der echtzeitkritischen Übertragung ein Gateway zum Einsatz kommt. Die Fieldbus Foundation bindet auf der Feldebene ihren H1-Feldbus ein, so dass eine Unterscheidung in Feld- und Leitsystemebene der Automatisierungstechnik im Netzwerk bestehen bleibt. Auch eine Vielzahl von echtzeitfähigen Ethernet-Ansätzen trennen einen echtzeitkritischen Bereich über ein Subnetz mit einem Gateway zu Ethernet nach IEEE 802.3 ab.

Die zweite Felser-Klasse wird charakterisiert durch die Verwendung von Hubs, welche die Laufzeitverzögerung minimieren. Trotz der hohen Echtzeitfähigkeit ist zu kritisieren, dass die Verwendung von Hubs in einem $100Mbit/s$-Netzwerk unüblich ist. Ein weiterer Nachteil der Hubs liegt in deren Halbduplexbetriebsart sowie in der fehlenden nebenläufigen Kommunikation. Ein Querverkehr wird nur aufgrund der Broadcastsendungen ermöglicht. Auch in der zweiten Felser-Klasse werden Gateways zum Ethernet nach IEEE 802.3 verwendet und es herrscht eine Verkabelung in der Baumtopologie vor. Da sich aufgrund der Verwendung von Hubs lediglich eine einzige Kommunikation zu einem Zeitpunkt störungsfrei auf der Leitung befinden kann, ist eine logische Bustopologie erkennbar. Eine Linien-Verkabelung wird durch die Integration der Verteiler in die Geräte emuliert, um die Verkabelung zu minimieren.

In der dritten Klasse nach Felser herrschen die aus der Automatisierungstechnik üblichen Linien- und Ringtopologien vor. Dabei werden zumeist beide Leitungspaare der herkömmlichen Ethernet-Verkabelung zur Bildung der Ringtopologie verwendet, so dass eine Bus-Verkabelung entsteht. Wird diese Bus-Verkabelung wie bei SERCOS III zusätzlich zu einem Ring geschlossen, so entsteht ein Doppel-Ring, der Hardware-Redundanz ermöglicht. Einzelne Geräte mit der Aufgabe eines Verteilers existieren nicht mehr. Statt der Übertragung unabhängiger Frames hat sich das Rundsenden eines einzelnen Frames maximaler Größe etabliert. Diese Frames entsprechen zwar noch dem Ethernet-Standard, der „on-the-fly“-Zugriff auf die Frames jedoch nicht mehr. GinLink arbeitet sogar mit einer völlig eigenen Busphysik, wobei an einem GinLink-Knoten Ethernet-Frames hineingegeben und an einem anderen wieder entnommen werden können. SynqNet hingegen verbessert die Echtzeiteigenschaften durch eine Entfernung der Präambel und des Interframe-Gap. Auch diese Frames entsprechen damit nicht mehr dem Standard nach IEEE 802.3. Auch wenn das Medium des Subnetzes nicht mehr Ethernet-konform arbeitet, bietet es jedoch in allen Fällen eine Ethernet-kompatible Schnittstelle oberhalb der Sicherungsschicht. Die Ansätze von EtherCAT und SERCOS III zeigen jedoch, welches Potential in der Architektur des Ethernets verborgen ist. Auch wenn eine Frameanalyse mit einem Protokollanalysator in diesen beiden Fällen möglich ist, wird die Kompatibilität der Hardware-Komponenten verringert.

Als Ausnahme ist ProfiNet IRT zu sehen, welches eine auf Switches basierte Netzwerkinfrastruktur verwendet. Dabei ist zu beachten, dass die eingesetzten Switches - wie der ERTEC 400 - proprietäre Hardware verwenden. Die Verkabelung entspricht dem Ethernet-Standard mit der Ausnahme, dass die Switches vermascht verdrahtet werden können, wodurch Redundanz ermöglicht wird. Versendet werden ausschließlich Frames, die dem Ethernet-Standard entsprechen. Es wird eine dezentrale Kommunikation unterstützt, die bei der Anlagenkonfiguration zu definieren ist. Die gleichzeitigen IRT-Übertragungen sind aufgrund der ausschließlichen Verwendung von 4-Port Switches auf maximal zwei Übertragungen pro Switch begrenzt. Auf welche Weise die Schedules berechnet werden ist nicht offen gelegt. Im Gegensatz zu herkömmlichen Switches verzögern die ProfiNet-Switches die Frames aufgrund der modifizierten Hardware weniger stark. In einem solchen Subnetz können auch asynchron sendende Geräte angeschlossen werden, die eine hohe Netzlast verursachen. Dabei wird der Echtzeit-Verkehr nicht beeinträchtigt, da die ProfiNet-eigenen Switches den asynchronen Datenverkehr getrennt behandeln.

3.3 Begründung für ein neues Framework

3.3.1 Zusammenfassung der vorhandenen Ansätze

Aus der Analyse der vorgestellten Ansätze zur Realisierung von Echtzeit-Ethernet läßt sich entnehmen, dass sich die Kompatibilität zum herkömmlichen Ethernet nach IEEE 802.3 konträr zur minimalen Verzögerung und Jitter von Frames verhält. Neben dem nichtdeterministischen CSMA/CD wirkt der Overhead des Ethernet-Frames mit Präambel und Interframe-Gap der Echtzeitfähigkeit entgegen. Eine Modifizierung der Frames sorgt dafür, dass die Kompatibilität nicht mehr sicher gestellt wird.

Zusätzlich werden im Standard-Ethernet Switches mit hoher Verzögerung durch Pufferung von ganzen Frames oder von Teilen von Frames in einer Baumtopologie eingesetzt. Gerade diese Topologie wirkt der in der Automatisierungstechnik typischen Linientopologie zur minimalen Verkabelung entgegen. Die Idee liegt hier in der Emulation der Linientopologie, indem die Verteiler in die Geräte integriert werden. Die Linientopologie, welche von einer Zentrale gesteuert wird, zeigt, dass immer noch ein Master/Slave-Denken innerhalb der Automatisierungstechnik überwiegt und alle Antriebe von einem zentralen Master gesteuert werden. Die Idee der dezentralen Peripherie wird in der Automatisierungstechnik durch die Einführung von Querverkehr noch unzureichend umgesetzt. Lediglich ProfiNet IRT bietet unabhängige Kommunikation zwischen einzelnen Geräten an.

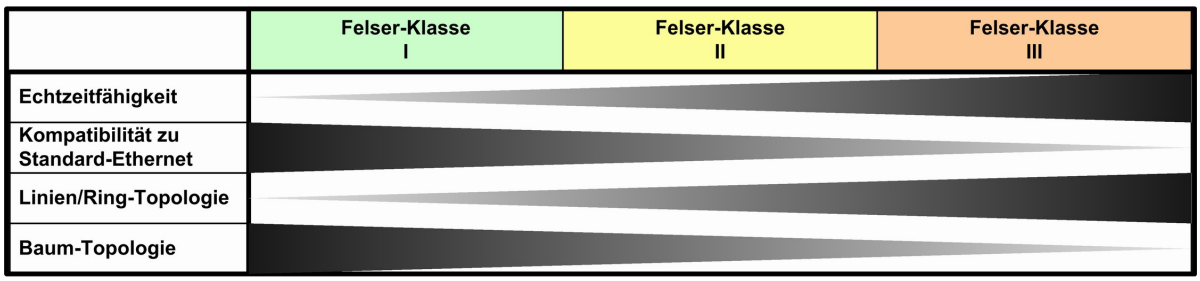


Abbildung 3.40: Gegenläufige Eigenschaften innerhalb der Echtzeitklassen

Bei den für die Antriebstechnik geeigneten Lösungen ist eine Modifizierung der Sicherungsschicht und sogar des Übertragungsmediums üblich, um die Echtzeitbedingungen zu erfüllen. Dabei werden die Netze zu Subnetzen gekapselt, die über ein Gateway mit dem Ethernet auf den Leitebenen kommunizieren. Auch wenn die Kommunikation transprarent ist, man also die Feldgeräte von außen wie Ethernet-Geräte ansprechen kann, kann aufgrund der Subnetz-Bildung gleichermaßen weiterhin ein konventioneller Feldbus eingesetzt werden. Beispiele hierfür sind die H1-Geräte sowie die GinLink-Architektur, welche in Wirklichkeit mit einer anderen Busphysik operieren.

Auch wenn Standard-Frames verwendet werden, lässt nur ProfiNet IRT als hart echtzeitfähige Lösung Standard-PCs im Echtzeit-Subnetz zu, auf denen beispielsweise Büroanwendungen ausgeführt werden können. Dies funktioniert nicht bei einer implementierten Echtzeit über der Transportschicht, bei der Restriktionen in der Echtzeitfähigkeit des Netzes getroffen werden müssen. Ein solches Netz ist für den Einsatz in der Antriebstechnik nicht geeignet. Der Grund für das Misslingen liegt darin, dass nur mit Änderungen innerhalb der Medienzugriffskontrolle der Sicherungsschicht und/oder einer modifizierten Bitübertragungsschicht harte Echtzeit erreichbar ist. Die herkömmlichen Geräte verfügen jedoch nicht über diese Modifizierungen und senden eben nicht nur dann, wenn sie von einem Master dazu aufgefordert werden, es von einer Schedule oder einem Token erlaubt wird. Solche Geräte würden also jedes aufgesetzte Protokoll ignorieren und somit die Echtzeitfähigkeit des Netzes zerstören. ProfiNet IRT bindet diesen asynchronen Verkehr durch modifizierte Switches mit ein, indem asynchrone Frames nur dann weitergeleitet werden, wenn sie isochrone Frames nicht behindern. Füllen sich die Puffer aufgrund hoher asynchroner Netzlast, so verhält sich ein ProfiNet-Switch lediglich für diesen asynchronen Verkehr äquivalent zu einem herkömmlichen Switch.

3.3.2 Ziele und Abgrenzung des Frameworks

Den bislang vorhandenen Lösungen ist zu entnehmen, dass die Erreichung der höchsten IAONA-Echtzeitklasse zu Zwecken der Antriebstechnik bei gleichzeitiger vollständiger Kompatibilität derzeit nicht möglich ist. Gleichzeitig bieten etablierte Lösungen mehrere Modi an, um einen Übergang zwischen dem Ethernet auf der Büroebene und der Feldebene zu schaffen. Ziel dieser Arbeit ist es nicht, einen weiteren *festen* Lösungsansatz zu schaffen, der sich in die bislang analysierten Verfahren zur Erreichung von echtzeitfähigem Ethernet einreiht und einen weiteren Kompromiss zwischen der Kompatibilität zu Ethernet IEEE 802.3 und der Echtzeitfähigkeit schafft.

Statt dessen soll ein *flexibles Framework* erstellt werden, um bei jeder konkreten automatisierten Anlage zwischen Echtzeitfähigkeit und Kompatibilität abwägen zu können. Die Ideen und Vorteile der bestehenden Ansätze für echtzeitfähiges Ethernet sollen dabei in das Framework integriert werden. Hubs sind an Stellen sinnvoll, wo nur eine gleichzeitige Kommunikation unidirektional stattfinden kann. Dies ist bei Linientopologien der Fall oder dann, wenn ein Gerät stets zu allen anderen kommuniziert. Der Fokus soll jedoch auf die Erhaltung der Kompatibilität zu Standard-Ethernet liegen, wobei es sich momentan um $100 Mbit/s$-Netzwerke mit Baumtopologie und Einsatz von Switches handelt. Die Switching-Technologie, wie ProfiNet sie verwendet, soll bei diesem Ansatz eine zentrale Rolle spielen. Der Grund dafür liegt darin, dass es sich hierbei um die Standard-Infrastruktur

des herkömmlichen Ethernet handelt und beliebiger Querverkehr zugelassen wird. Dies entspricht eher dem Trend der dezentralen Peripherie der Automatisierungstechnik als ein beschränkter Querverkehr zwischen Slave-Achsen, die hinter einer Master-Achse angeordnet sind. Die parallele Kommunikation soll also mit Blick auf den Trend zur dezentralen Peripherie gefördert werden.

Die Verzögerungen, wie sie durch cut-through oder gar store-and-forward Switches entstehen, sind dabei zu minimieren. Dies ist mit Standard-Hardware natürlich nur begrenzt zu erreichen. Das zu erstellende formale Modell geht an dieser Stelle unabhängig von der Implementierung von allgemeinen Parametern zur Verzögerung und zum Jitter in jedem Verteiler aus. Es müssen zwar Verhaltenweisen von Verteilern im Netzwerk, Verzögerungszeiten und Jitter beachtet werden, jedoch findet keine Modellierung auf Signalebene statt. Für die Übertragung von Protokollen ist lediglich eine gewisse Zeit pro übertragenem Byte vorzusehen sowie die gegebenen Daten innerhalb eines Ethernet-Frames.

Ein Ziel ist die Entwicklung eines effizienten Scheduling der Übertragungen, die in jedem Switch parallel ablaufen können. Wird ein hochpriores Ereignis, wie das Betätigen eines Not-Aus Schalters, einbezogen, so ist dafür ein fester Zeitslot vorzusehen. Da diese Arbeit auf das Erstellen von Schedules fokussiert ist, wird eine zusätzliche Priorisierung des Datenverkehrs nicht betrachtet. Die Ursache dafür liegt darin, dass bereits gängige Verfahren zur Priorisierung von Ethernet-Daten existieren, wie die Priorisierung nach IEEE 802.1p. Da jedoch die Möglichkeit zu asynchroner Datenübertragung vorgesehen werden soll, kann dort eine Priorisierung problemlos nachträglich hinzugefügt werden. Somit können auch Übertragungen mit weniger harten Echtzeitanforderungen betrachtet werden.

Aus dem gleichen Grunde werden keine verschiedenen Maßnahmen zur Synchronisierung betrachtet, da bereits etablierte Methoden - wie das Senden von SYN-Frames, PTCP und insbesondere IEEE 1588 - existieren. Für die Erstellung der Schedules bedeutet dies, dass lediglich eine gewisse Zeit zu Beginn jedes Zyklus für die Synchronisierung verwendet werden muss. Zu beachten ist lediglich die mögliche Genauigkeit der Verfahren, da diese einen Einfluss auf den Jitter besitzen.

Außerdem wird keine Betrachtung von Hardwareredundanz durchgeführt, was zu Ringen und vermaschten Strukturen der Netzwerktopologie führen würde. Vermaschte Netze mit alternativen Routen sind erst auf IP-Ebene durch die Verbindung von mehreren Ethernet-Subnetzen mittels Routern auf der Vermittlungsschicht etabliert.

Es sollen jedoch Ansätze vorgestellt werden, in denen Geräte wie Standard-Laptops ohne Kenntnis eines speziellen Echtzeit-Protokolls dynamisch in die laufende Anlage hinzugefügt werden können. Die Laptops sollen zum Beispiel in der Lage sein, Sensordaten direkt auszulesen ohne dabei den Betrieb der Anlage zu gefährden. Es soll jedoch auch die Integration handelsüblicher Switches betrachtet werden, die einen fließenden Übergang in ein Büro-Ethernet darstellen. Die Bildung von geschlossenen Subnetzen, wie es bislang in vielen Ansätzen vorherrscht, soll vermieden werden. Das zu entwickelnde Architekturmodell soll vielmehr an die Anforderungen der konkreten Anlage anpassbar sein und gleichzeitig den Eindruck vermitteln, dass es sich durchgängig um ein Standard-Netzwerk nach IEEE 802.3 handelt.

Der neue Ansatz soll also von konkreter Hardware und Protokollen durch die Bildung eines formalen Modells abstrahieren mit dem Ziel der Allgemeingültigkeit. Dieses Modell

soll viele der existierenden Lösungen beinhalten und formal beschreiben können. Das Ziel dieser Arbeit besteht in der Erstellung eines flexiblen formal begründeten *Frameworks*, das in Abhängigkeit der konkreten automatisierungstechnischen Anforderungen eine Abwägung zwischen der Kompatibilität und der Echtzeitfähigkeit des Netzwerkes durchführt.

Als Ausgangspunkt für das Framework wird eine sternförmige Netzwerktopologie verwendet mit beliebiger, einheitlicher Bandbreite, was einem gängigen $100 Mbit/s$-Netzwerk entspricht. Eine in der Automatisierungstechnik übliche Linientopologie soll emuliert werden können, wobei die notwendigen Verteiler in die Geräte integriert sein können. In der Modellierung entspricht ein Glied in der Linientopologie einem Verteiler mit drei Ports, wobei zwei Ports nach außen zugänglich sind und der dritte Port zum Gerät führt. Als Verteiler können sowohl Switches, als auch Hubs eingesetzt werden.

Es wird davon ausgegangen, dass die Infrastruktur des echtzeitfähigen Netzes im Vorfeld bekannt ist. Während die industrielle Anlage in Betrieb ist, werden keine echtzeitfähigen Geräte hinzugefügt oder entfernt, da dies auch die Programmierung der Anlage beeinflussen würde. Das Gleiche gilt für die zur Verfügung stehenden Bandbreite der Hardware ebenso wie für die Sendeanforderungen der Geräte. Die Einführung der Echtzeit soll über ein TDMA-Zugriffsverfahren erfolgen. Die Berechnungen zur Erstellung der Schedules sollen offline erfolgen, beispielsweise im Rahmen einer Konfigurations-Software. Die Ergebnisse, also die generierten Schedules, sollen dann in die Anlage geladen und beim Hochlauf initialisiert werden.

Zusätzlich zu dem TDMA-basierten Verkehr sollen asynchrone Daten übertragen werden können, wobei die betreffenden Geräte Kenntnis von dem modifizierten Protokoll besitzen müssen, z. B. um Sensordaten direkt auszulesen. Zusätzlich dazu sollen herkömmliche PCs mit Standard-Anwendungen in das Netzwerk integriert werden können. Diese Geräte besitzen keine Kenntnis von einem modifizierten Protokoll.

3.4 Vorgehensweise bei der Erstellung des Frameworks

Bei der Entwicklung des Frameworks wird zunächst von einem stark vereinfachten Modell ausgegangen. Aufgrund dieser hohen Anzahl an Einschränkungen wird erwartet, dass eine schnelle Lösungsfindung möglich ist. Im weiteren Verlauf der Arbeit werden dann einzelne vereinfachende Nebenbedingungen entfernt, die daraus resultierenden Schwierigkeiten diskutiert und behandelt.

Im ersten Schritt wird die Netzwerktopologie formal abgebildet. Im ersten Szenario wird von einer reinen Halbduplexübertragung in einem baumförmig verdrahteten Netzwerk ausgegangen unter Vernachlässigung von Verzögerungszeiten und Jitter. Es werden zunächst idealisierte Switches verwendet. Eine Linientopologie als Spezialfall eines Baumes ist bereits in dieser Lösungsmenge enthalten. Im Anschluss daran werden die im Vorfeld bekannten, kausal unabhängigen IRT-Übertragungen der Geräte modelliert, wobei zunächst von Unicast- und Broadcastsendungen mit einheitlicher Framelänge ausgegangen wird. Des Weiteren wird zunächst vereinfachend angenommen, dass jede Kommunikation in jedem Zyklus statt findet. Um sowohl Pufferung in den Switches, als auch Kollisionen auf den Leitungen auszuschließen, müssen die IRT-Übertragungen in eine zeitliche Ordnung gebracht werden. Unabhängige IRT-Übertragungen können dabei parallel - also gleichzeitig - durchgeführt werden. Ziel ist also die Erstellung eines Fahrplans - also ei-

ner Schedule - für die IRT-Übertragungen. Als Nebenbedingung ist zu nennen, dass die Anzahl der Zeitslots zu minimieren ist, woraus eine minimale Zykluszeit als Ergebnis der Berechnung der Offline-Schedules resultiert. Im Anschluss daran werden Möglichkeiten zur Integration von asynchronen Übertragungen diskutiert, welche keine Auswirkungen auf das Echtzeitverhalten des Systems besitzen dürfen.

Als erste Einschränkung wird in dem folgenden Szenario der Halbduplexbetrieb aufgehoben, so dass das Netzwerk, wie es im $100Mbit/s$-Ethernet üblich ist, im Vollduplexmodus arbeitet. Im Anschluss daran werden weitere Eigenschaften modelliert, wie die Integration von Hubs in die Modellierung (Kapitel 4.4), die Integration von asynchron sendenden Geräten (Kapitel 4.5), verschiedene Framegrößen (Kapitel 4.6), zyklische Sendungen in Vielfachen von Produktionszyklen (Kapitel 4.7) sowie die Kalkulation von Verzögerungszeiten (Kapitel 4.8).

Zusätzlich soll berücksichtigt werden, wo die zu erstellenden Schedules umgesetzt werden. Dazu sind verschiedene Fälle zu unterscheiden und zu diskutieren. Es soll aufgezeigt werden, welche Abhängigkeiten zwischen der Umsetzung der Schedules und ihrer Berechnung bestehen. Ebenso sollen Abhängigkeiten zwischen der Umsetzung und der Integration von asynchron sendenden Geräten diskutiert werden.

Die gesamte Vorgehensweise ist in Abbildung 3.41 skizziert. Die Netzwerkinfrastruktur, also die Verdrahtung der Geräte mit Echtzeitanforderungen, wird als gegeben vorausgesetzt. Dies gilt auch für die IRT-Übertragungen der Geräte. Auf der Basis der Programmierung der konkreten automatisierten Anlage ergibt sich, welches Gerät mit welchem anderen Gerät kommunizieren muss.

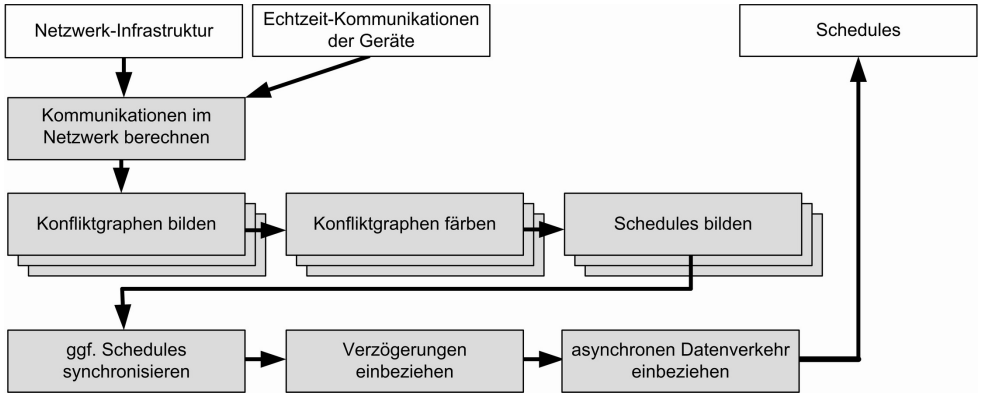


Abbildung 3.41: Vorgehensweise zur Lösungsfindung

In einem ersten Schritt wird der Verlauf der IRT-Übertragungen im Netzwerk berechnet. Dadurch zeigt sich, welche IRT-Übertragungen miteinander in Konflikt stehen und durch eine Schedule verwaltet werden müssen. Im Anschluss daran wird für jeden Switch im Netzwerk ein Konfliktgraph erstellt, der jeweils die IRT-Übertragungen enthält, die diesem Switch betreffen. Durch die Färbung der Konfliktgraphen kann dann eine lokale Schedule erstellt werden. Die einzelnen lokalen Schedules der Switches sind im nächsten Schritt ggf. miteinander zu synchronisieren. Im Anschluss daran können Verzögerungszeiten und Jitter die Schedules nochmals verändern. Im letzten Schritt werden Möglichkeiten zur Einbeziehung von asynchronem Datenverkehr in die Schedule betrachtet. Als Ergebnis ist abschließend die Schedule für das Gesamtnetz als Ausgabe zu erwarten.

3.5 Netzwerke und Schedulingprobleme

Bevor mit der eigenen Modellierung begonnen wird, werden exemplarisch Ansätze betrachtet, die sich mit Schedulingproblemen in einem ähnlichen Problemfeld beschäftigen. Es existiert eine Vielzahl von Verfahren zum Online-Scheduling in Betriebssystemen, in denen gemeinsam genutzte Betriebsmittel zu einer Liste von wartenden Prozessen zugeteilt werden.

Die Idee des prioritätenbasierten Scheduling wird - wie bereits am Beispiel von EtherNet/IP in Kapitel 3.1.1.1 beschrieben - im Umfeld des echtzeitfähigen Ethernets bei der Einführung von VLAN-Prioritäten verwendet. Es wurde gezeigt, dass diese Art der Priorisierung für das Erreichen harter Echtzeitfähigkeit unzureichend ist. Die Idee der formalen Modellierung von Jasperneite [Jas02] ist jedoch trotz des Fokus auf die Priorisierung von Interesse. Jasperneite modelliert die Switches und angeschlossene Geräte mit hohen Echtzeitanforderungen mit Hilfe einer Software und simuliert verschiedene Weiterleitungsmodelle der Switches mit dem Ziel einer Machbarkeitsanalyse. Dieses Modell betrachtet verschiedene Bedienstrategien der Switches wie *First Come First Served* (FCFS), *Priority Queuing* (PQ) und *Fair Queuing* (FQ).

RTnet verwendet als Alternative zu dem Token-basierten Ansatz ein preemptives *Earliest Deadline First Scheduling* (EDF). Dies ist möglich, da das Netzwerk ähnlich einer CPU ein Betriebsmittel bzw. eine Resource darstellt, das von Mehreren in Anspruch genommen wird. Im Umfeld der Betriebssysteme sind dies Prozesse, welche Rechenzeit anfordern. Im Falle eines Netzwerkes sind es Geräte, die ihre zu sendenden Datenströme so absetzen wollen, dass sie rechtzeitig bei den empfangenen Geräten eintreffen: „*Basically a stream that wants to transmit on the network is analogous to a task that wants to execute on a CPU.*“ [HJSM05]

Bei RTnet werden die Übertragungen der sendenden Geräte nach ihren Anforderungen an die Echtzeit preemptiv gescheduled. Dies setzt voraus, dass sich alle sendenden Geräte an das Protokoll zur Medienzugriffskontrolle halten.

Harte isochrone Echtzeitbedingungen können mit einem solchen Online-Scheduling jedoch nicht erfüllt werden, da bereits eine geringe Wartezeit nicht geduldet werden kann. Andererseits sind als vereinfachende Nebenbedingung die anstehenden Übertragungen für jeden Switch im Vorfeld bekannt, was die Basis für ein *Offline-Scheduling* darstellt. Die einzelnen Schedules können also vor ihrer Ausführung kalkuliert werden und sind während der Ausführung unveränderlich.

Zu bedenken ist auch, dass die laufende Übertragung eines Datenframes nicht unterbrochen werden kann, da dies zu einem defekten Frame führen würde. Wird ein solcher defekter Frame im Netzwerk erkannt, kann dies die Performance des Netzwerkes senken. Betrachtet man ein Netzwerkkabel als eine Ressource, so darf zu einem Zeitpunkt genau ein Frame[4] bzw. genau ein Frame in eine Richtung[5] nicht-unterbrechbar übertragen werden. Dies erfordert ein *nicht-preemptives Scheduling* einzelner Frames.

Das Verhalten der Schedules von ProfiNet-Switches beschreibt Felser in einer seiner Veröffentlichungen [Fel05]. Abbildung 3.42 zeigt vernetzte ProfiNet IRT Geräte mit integrierten Switches. Zu den Sendungen werden die Eingangs- und Ausgangsports festge-

[4] im Halbduplexbetrieb
[5] im Vollduplexbetrieb

halten. Wird ein IRT-Frame von einem Gerät nicht weitergesendet, so ist das betreffende Gerät das Ziel der Adressierung gewesen.

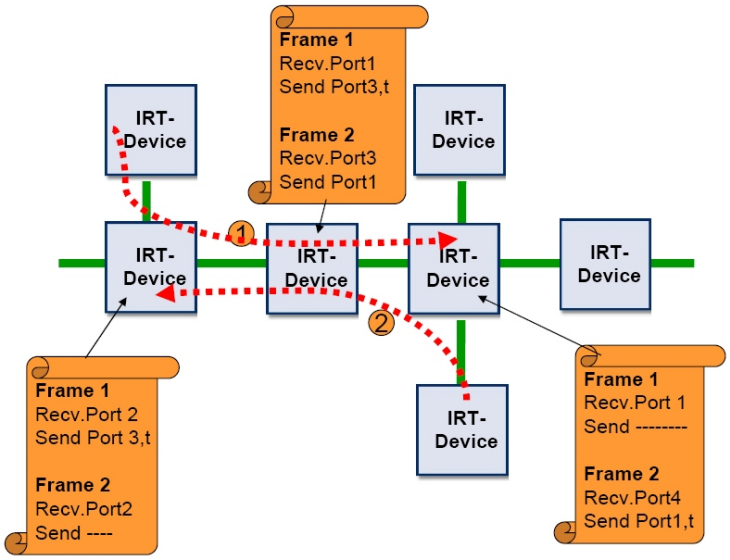


Abbildung 3.42: ProfiNet IRT-Übertragungen [Fel05]

In einem ähnlichen Zusammenhang erwähnt Felser den Begriff des Taktfahrplans. „*Die Weichen werden aufgrund des geplanten Fahrplans und der aktuellen Uhrzeit richtig gestellt. Alle Züge können gleichzeitig auf dem Netz verkehren und, wenn der Plan richtig ist und keine Verspätungen außerhalb der erlaubten Toleranzen auftreten, gibt es auch keine Möglichkeit für Kollisionen.*“ [Fel05]

Eine verwandte Problemstellung aus der Domäne der Fahrpläne behandelt Guckert in seiner Dissertation der „*Anschlussoptimierung in öffentlichen Verkehrsnetzen*“ [Guc96] mit der Ziel der Erstellung eines Fahrplans, also einer Schedule für Züge. „*Die Wahl von Abfahrtszeiten der Linien steht im Mittelpunkt der Untersuchungen.*“ [Guc96] Dabei werden *Taktfahrpläne* für Züge erstellt, die sich auf einem Schienennetz bewegen. Auf einem Gleis dürfen Züge nicht zu einem Zeitpunkt in entgegengesetzter Richtung fahren. Die Fahrzeiten zwischen zwei Bahnhöfen werden fest vorgegeben. Ziel seiner Arbeit ist es, die Umsteigezeit von Passagieren in Zwischenbahnhöfen zu minimieren. Die Passagiere reisen von einer Quelle zu einer Senke, die beides ebenfalls Bahnhöfe darstellen. Dazu wurde sowohl das Netz, als auch die Flüsse von Quellen zu den Senken sowie der resultierende Taktfahrplan formal modelliert.

Ein ähnliches Modell lässt sich auf die Erstellung eines echtzeitfähigen Ethernets anwenden, wenn man eine Analogie zwischen dem Schienennetz und der Infrastruktur eines Ethernetnetzwerkes, zwischen Gleis und Kabel, Passagier und Nutzdaten, Zug und Datenframe mit einer Anzahl von Nutzdaten, Bahnhof und Switch, Gleise für eintreffende Züge und Eingangsports eines Switches, Gleise für abfahrende Züge und Ausgangsports eines Switches, Taktfahrplan und Schedule für einen Produktionstakt sowie Abfahrtszeit und Sendezeitpunkt eines Frames innerhalb eines Taktes vornimmt. Im Gegensatz zu einem baumförmigen Ethernet-Netzwerk existieren im Schienennetz natürlich alternative Wege, so dass neben den Abfahrtszeiten als Zielgröße alternative Routen zur Verfügung stehen.

Von Interesse ist jedoch der Aspekt des Taktfahrplans mit dem Ansatz, „*Züge auf immer gleichen Routen in festen Abständen verkehren zu lassen.*“ [Guc96] Bei fest vorgegebenen Fahrzeiten „*hängt die Qualität der Anschlüsse ausschließlich von der Wahl der Abfahrtszeiten der Züge in den Bahnhöfen ab. Durch eine geschickte Wahl von Abfahrtszeiten und Haltezeiten in den Bahnhöfen sollen die Wartezeiten beim Umsteigen auf ein Minimum reduziert werden.*“ [Guc96]

Diese Aussagen sind auf die Forderungen der Automatisierungstechnik übertragbar, indem man den Taktfahrplan auf den Produktionszyklus einer automatisierten Anlage überträgt. Die „*Wahl der Abfahrtszeiten*“ deutet die Schedule an, indem konkurrierende Züge das gemeinsame Betriebsmittel, hier ein Gleis, ohne eine zeitliche Überschneidung benutzen. Die „*Wartezeiten beim Umsteigen*“ entsprechen den Zeiten für die Pufferung von Paketen in Switches, wodurch die Verzögerungszeit der Sendung vergrößert wird.

Für die Übertragung von unterschiedlich großen Frames existieren zwei Verfahren zur Erzeugung einer Schedule. Der *First-Fit* (FF) Algorithmus basiert auf dem Problem des Bin-Packing [JDUGG74], bei dem eine Menge von Teilen unterschiedlicher Länge in eine möglichst geringe Anzahl vorgefertigter Schachteln verpackt werden muss. Ein ähnlicher Ansatz ist das List-Scheduling, dass bereits 1969 von Graham [Gra69] vorgestellt wurde und von Erlebach [Erl98] zur Erstellung der Schedules verwendet wurde. Einen interessanten Ansatz zur Modellierung und zur Lösungsfindung beschreibt Erlebach in seiner Dissertation mit dem Thema „*Scheduling Connections in Fast Networks*“. Er beschreibt ein formales Modell optischer *Asynchronous Transfer Mode Netzwerke* (ATM). Jede Unicastübertragung bildet einen Pfad innerhalb des Netzes und benötigt einen Teil der Bandbreite. Im Gegensatz zu $100 Mbit/s$-Ethernet mit Kupferleitern können in optischen Netzen mehrere Übertragungen gleichzeitig ein Medium verwenden, indem sie auf unterschiedliche Wellenlängen übertragen werden. Jede Leitung besitzt jedoch eine Obergrenze an Bandbreite, die auf die anstehenden Übertragungen zu verteilen ist. Dazu muss ein entsprechender Fahrplan erstellt werden, den Erlebach unter Verwendung der Graphenfärbung, wie in Kapitel 2.3.3.5 beschrieben wurde, erstellt. Im vierten Kapitel seiner Arbeit beschreibt Erlebach Algorithmen für das Scheduling von Unicast-Nachrichten in Halbduplex- und Vollduplexnetzen mit baumförmiger Infrastruktur, die auf die Problemstellung echtzeitfähiger Ethernet-Netze anwendbar sind. Er findet heraus, dass die zu färbenden Graphen eines Verteilers im Netzwerk bipartit und damit in polynomieller Zeit exakt färbbar sind; das Verfahren wird als Call Scheduling bezeichnet. Jeder Call besteht aus

- dem sendenden Gerät u_c als Kommunikations-Endpunkt,
- dem empfangenden Gerät v_c als Kommunikations-Endpunkt,
- einer geforderten Bandbreite b_c und
- einer Zeitdauer d_c, welche die Kommunikations-Anforderung vom Senden des ersten Bits von u_c bis zum Empfangen des letzten Bits durch v_c benötigt.

Erlebach definiert im nächsten Schritt das *Call-Scheduling* Problem mit einer gegebenen Netzwerkinfrastruktur, die als Graph modelliert ist, Bandbreiten-Kapazitäten sowie mit einer Menge von Calls. Die Lösung des Problems besteht in der Zuweisung von Pfaden und Startzeiten zu jedem Call in der Art, dass die Summe der geforderten Bandbreiten an

keiner Kante des Netzwerk-Graphen zu keiner Zeit die maximale Bandbreite übersteigt. Die Schedule ist so zu optimieren, dass sie die kleinste Dauer besitzt. Für baumförmige Netze, in denen laut Definition keine alternativen Routen existieren, und in denen technologiebedingt die zur Verfügung stehenden Bandbreite nicht auf mehrere Übertragungen verteilt werden kann, identifiziert Erlebach das Problem des Path Coloring. Dabei wird jedem konkurrierenden Kommunikationspfad durch die Verwendung von Graphenfärbung (s. Kapitel 2.3.3) eine eigene Farbnummer zugeordnet, wobei die Anzahl der verwendeten Farben zu minimieren ist. Die Farbnummer entspricht der Sequenznummer bzw. der Nummer des Zeitslots. Calls mit einer einheitlichen Farbnummer können parallel ausgeführt werden. Die Farbnummer n werden nacheinander abgearbeitet, so dass sich aus der längsten Übertragungszeit eines Calls einer Farbe die Zeitdauer der Abarbeitung dieser Farbnummer ergibt. Die Dauer aller Farbnummern ergeben die Dauer der Gesamtschedule.

Die in dem folgenden Kapitel beschriebene Modellierung des formalen Frameworks orientiert sich an den Arbeiten von Erlebach, Felser und Guckert und verbindet deren Ansätze mit neuen, eigenen Ansätzen zur Formalisierung von echtzeitfähigen Ethernet-Netzwerken.

Kapitel 4

Modellierung des Frameworks

Dieses Kapitel beinhaltet die formale Modellierung des Frameworks zur Erreichung von echtzeitfähigem Ethernet. Dazu wird zunächst ein stark vereinfachtes Modell der Netzwerkinfrastruktur und der IRT-Übertragungen gebildet und eine Vorgehensweise zur Ermittlung der Schedules beschrieben. In den folgenden Unterkapiteln wird das Modell komplexer und näher an die realen Bedingungen herangeführt.

Die mathematische Modellierung und die Beweise der Aussagen wurden im Rahmen der Diplomarbeit von Nowak [Now06] durchgeführt. Nachfolgend sollen nun diese Aussagen und Ergebnisse diskutiert und deren Bedeutung für den Kontext der Automatisierungstechnik herausgearbeitet werden.

4.1 Idealisierte Halbduplexübertragung

Zu Beginn der Modellierung wird die ausschließliche Verwendung von idealisierten, nicht blockierenden Switches angenommen. Dies bedeutet, dass bei n Ports im Idealfall $n/2$ Übertragungen gleichzeitig möglich sind, ohne dass Daten zwischengespeichert oder verworfen werden. Ein auf Switches basierendes Netzwerk arbeitet nach IEEE 802.3 zwar im Vollduplexmodus, jedoch wird zur Vereinfachung des Modells zunächst von einem Halbduplexbetrieb ausgegangen. Eine solche Lösung ist auch in einem Vollduplexnetzwerk gültig; lediglich die Bandbreite des Netzes wird schlechter ausgenutzt. Verzögerungszeiten und deren Jitter werden zunächst ignoriert.

Des Weiteren werden zunächst einheitliche Framelängen angenommen. Für die Automatisierungstechnik würde man $64Byte$ als Framegröße wählen, also die minimale Framelänge eines IEEE 802.3 Frames. Damit bleibt die Kompatibilität zu Standard-Ethernet gewahrt. Der Datenteil in Höhe von $46Byte$ ist für Anwendungen der Automatisierungstechnik absolut ausreichend. Soll die Zykluszeit weiter verringert werden, so kann die minimale Framelänge zu Lasten der Kompatibilität auch weiter gesenkt werden. Die zusätzliche Übertragung von asynchronen Frames wird zunächst nicht betrachtet und später in die entstandene Schedule der Echtzeitübertragungen integriert. Es wird zunächst angenommen, dass jede IRT-Übertragung in jedem Sendezyklus auftritt und innerhalb eines Sendezyklus keine kausalen Abhängigkeiten existieren. Es existiert also innerhalb eines Sendezyklus keine vorgegebene Reihenfolge. Im ersten Schritt werden ausschließlich Unicastübertragungen betrachtet.

4.1.1 Modellierung des Netzwerkes

Zu Beginn wird die Netzwerkinfrastruktur als Graph modelliert. Über diesen Graph verlaufen die IRT-Übertragungen, deren Sendezeitpunkte zu planen sind. Das Netzwerk besteht aus einer endlichen Menge $\mathbb{S}$ von Switches und einer endliche Menge $\mathbb{D}$ von Geräten bzw. Devices, die zu einer Knotenmenge $\mathbb{V} = \mathbb{S} \cup \mathbb{D}$ zusammengefasst werden. Zusätzlich dazu existiert eine endliche Menge $\mathbb{E}$ von Netzwerkkabeln, wobei ein Netzwerkkabel zwischen zwei Knoten x und y als Kante $\{x, y\} \in \mathbb{V}^2$ zwischen den verbundenen Knoten repräsentiert wird. Die Menge der Netzwerkkabel, die an einen Switch $v \in \mathbb{S}$ angeschlossen sind, wird mit $\mathbb{E}(v)$ bezeichnet.

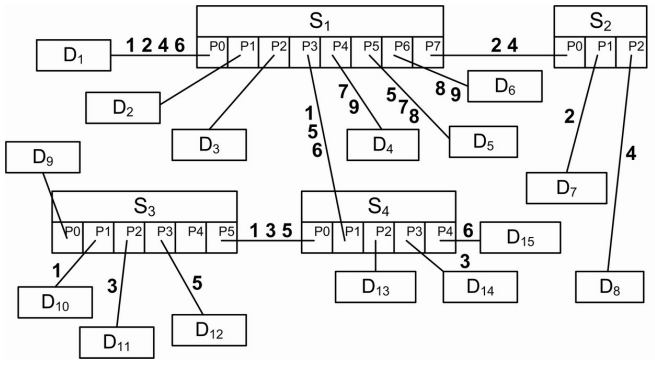
Zusätzlich dazu existiert eine endliche Menge Γ von IRT-Übertragungen. Jede IRT-Übertragung kann anhand einer ID, die als $\Gamma - ID$ bezeichnet wird, identifiziert werden. Durchläuft eine Anzahl von IRT-Übertragungen einen Netzwerkknoten v, so werden diese mit $\Gamma_v \subset \Gamma$ bezeichnet. Jede *Unicastübertragung* $r \in \Gamma$ bekommt durch $\delta : \Gamma \mapsto \mathbb{R}^{>0}$ eine positive Übertragungsdauer $\delta(r)$ zugeordnet und verläuft von genau einem Quellgerät $Quelle(r) \in \mathbb{D}$ zu genau einem Zielgerät $Ziel(r) \in \mathbb{D}$.

Da Switches baumförmig verdrahtet werden, wird gefordert, dass $G_{Netz} = (\mathbb{V}, \mathbb{E})$ ein Baum ist. Dieses erste Modell eines Netzwerkes wird im Folgenden als Tupel $N = (G_{Netz}, \Gamma)$ beschrieben.

4.1.2 Modellierung der Übertragungen

Eine Kommunikationslinie repräsentiert eine Unicastübertragung mit deren verwendeten Knoten und Kanten und wird mit einer ID versehen. Als Notation für eine *Kommunikationslinie* wird $KL(\Gamma - ID) := KL(Quelle(r), Ziel(r))$ vereinbart. Da es sich bei dem Netzwerk um einen Baum handelt, ist der Weg von $Quelle(r)$ nach $Ziel(r)$ in G_{Netz} eindeutig bestimmt.

Das in Abbildung 4.1 skizzierte Beispiel einer Netzwerkinfrastruktur mit den gegebenen Übertragungen der Geräte wird im weiteren Verlauf des Kapitels zur Illustration des Modells verwendet.



Γ-ID	Quelle	Ziel
1	1	10
2	7	1
3	14	11
4	8	1
5	12	5
6	1	15
7	4	5
8	5	6
9	6	4

Abbildung 4.1: Beispielhafte Netztopologie und Übertragungen der Geräte

Um die Kommunikationslinien zu berechnen, müssen zunächst die Wege von den Quellen zu den Zielen bestimmt werden. Dazu wird zunächst ein beliebiger Switch $w \in \mathbb{S}$ als

Wurzel verwendet und die Teilwege von w zu den Geräten berechnet, indem die überquerten Netzwerkknoten festgehalten werden. Die Menge an Netzwerkknoten bzw. Netzwerkkanten, welche eine Unicastübertragung $r \in \Gamma$ durchläuft, wird als $V_r := V_{KL(r)}$ bzw. $E_r := E_{KL(r)}$ definiert.

Im Folgenden wird der Weg der Kommunikationslinie 3 aus Abbildung 4.1 betrachtet. Dieser Weg führt von dem Gerät 14 zu dem Gerät 11. Als Wurzelswitch wird Switch 1 gewählt. Somit ergeben sich die Wege $P = (S_1, S_4, D_{14})$ und $Q = (S_1, S_4, S_3, D_{11})$ vom Wurzelswitch zu den beiden Geräten. Die Komplexität zum Auffinden des Weges von einem Gerät zu w ist abhängig von der maximalen Länge eines Weges des Baumes $l(G)$. Im Anschluss daran werden die Teilwege so konkateniert, dass sie die Wege von den Quellen zu den Zielen darstellen. Dabei werden zunächst die identischen Teile der Wege gekürzt und der letzte Knoten der Kürzung als Z zwischengespeichert.

$P' = (\cancel{S_1, S_4}, D_{14})$
$Q' = (\cancel{S_1, S_4}, S_3, D_{11})$
$Z = (S_4)$

Die Konkatenation ($\circ$) der Teilwege erfolgt dann über die Invertierung eines Teilweges (vgl. Def. 2.3.21) nach dem folgenden Prinzip:

$$KL(3) := KL(14, 11) = P'_{inv} \circ Z \circ Q' = (D_{14}, S_4, S_3, D_{11})$$

4.1.3 Definition eines Konfliktes und dessen Eigenschaften

Ein *Konflikt* tritt ein, wenn zwei Frames gleichzeitig eine Leitung belegen. In diesem Fall entsteht eine Kollision, die das nicht-deterministische Exponential Backoff Verfahren nach sich zieht. Ein Konflikt tritt auch dann ein, wenn mindestens zwei Frames in einem Switch einen gemeinsamen Port verwenden wollen und daher gepuffert werden. Auf diese Weise entsteht eine lastabhängige Verzögerungszeit der Frames, die in harten Echtzeitumgebungen nicht tolerierbar ist.

Ein Konflikt von zwei Kommunikationslinien $K, L \in \Gamma$ in einer Halbduplexverbindung tritt also dann auf, wenn sie ein gemeinsames Kabel $e \in \mathbb{E}$ verwenden. Dann sagt man, K und L haben einen Konflikt in e, ausgedrückt durch $K \leftrightsquigarrow_e L$. Da ein Kabel an zwei Knoten (Switches und/oder Geräten) angeschlossen ist, besitzen die beiden Kommunikationslinien auch in diesen Knoten einen Konflikt. Betrachtet man die Geräte als Quellen und Senken, so stehen alle Übertragungen miteinander in Konflikt, die von dem jeweiligen Gerät versendet werden bzw. die für das jeweilige Gerät bestimmt sind.

Ein Konflikt in einem Knoten v wird ausgedrückt durch $K \leftrightsquigarrow_v L$, ein Konflikt in einer Menge V von Knoten durch $K \leftrightsquigarrow_V L$. Entsprechend sind K und L konfliktfrei in e, konfliktfrei in v, konfliktfrei in V oder global konfliktfrei, wenn sie dort keinen Konflikt haben. [Now06, Def. 4.2.2]

Durch die Definition des Konflikt-Begriffes können nun zwei Aussagen getroffen werden, die für die Berechnung der Schedule von Bedeutung sind. Die erste Aussage besteht darin, dass eine *lokale Konfliktfreiheit* eine *globale Konfliktfreiheit* impliziert [Now06, Satz 4.2.4]. Durchlaufen also zwei verschiedene Kommunikationslinien K und L einen gemeinsamen Knoten v und haben dort keinen Konflikt, so sind sie global konfliktfrei. Es ist anschau-

lich, dass zwei Kommunikationslinien, welche sich in einem Switch treffen und dort keinen gemeinsamen Port verwenden, in der Baumstruktur des Netzwerkes keine weitere Möglichkeit besitzen, sich nochmals zu treffen. Dies ist im Beispiel bei den IRT-Übertragungen 1 und 7 der Fall.

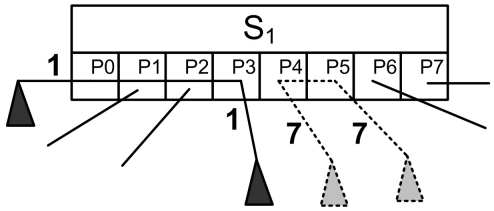


Abbildung 4.2: Zwei konfliktfreie Übertragungen in einem Switch

Die zweite Aussage besteht darin, dass Konfliktknoten stets einen Weg bilden [Now06, Def. 4.2.6], also jeweils durch eine zusammenhängende Folge von Knoten miteinander verbunden sind. Seien $K, L \in \Gamma$ zwei Kommunikationslinien, V die Menge der Switches, und E die Menge der Kabel, in denen K und L einen Konflikt haben. Dann gibt es einen Weg P mit $V_P = V$ und $E_P = E$. Betrachtet man den Baum der Netzwerkinfrastruktur, so können zwei Kommunikationslinien sich an einem Port eines Switches treffen und verlaufen dann gemeinsam über eine gewisse Anzahl von gemeinsamen Switches und Kabeln. Trennen sie sich jedoch an einer Stelle wieder, so können sie aufgrund des Baumes als Netzwerkinfrastruktur nicht wieder zusammen finden. Im Beispiel ist dies bei den IRT-Übertragungen 1 und 5 der Fall.

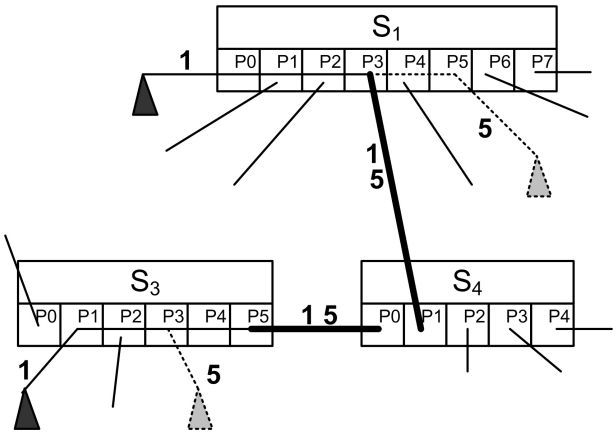


Abbildung 4.3: Zwei Kommunikationslinien mit einem Konflikt

Diese beiden zentralen Aussagen haben zur Folge, dass die Schedules der einzelnen Switches unabhängig voneinander berechnet werden können. Es lässt sich also für jeden einzelnen Switch eine eigene Schedule erstellen. Die resultierenden Schedules sind nach ihrer Erstellung synchronisierbar [Now06, Satz 6.2.1], wie in Kapitel 4.1.7 beschrieben wird. Im weiteren Verlauf dieses Kapitels wird nun die Vorgehensweise zur Erstellung der lokalen Schedules beschrieben.

4.1.4 Erstellung von Konfliktgraphen

Wie bereits in Kapitel 2.3.3.5 angedeutet wurde, können die Schedules über die Färbung von Konfliktgraphen erstellt werden. Die Konflikte werden dabei durch die Bildung des Ablaufplans bzw. Fahrplans zeitlich entzerrt, wobei diese Bildung bei idealisierter Halbduplexübertragung für jeden Switch unabhängig erfolgen kann.

4.1.4.1 Knotenkonfliktgraph

In einem lokalen Knotenkonfliktgraph $C_{Knoten}(\Gamma_v)$ werden die IRT-Übertragungen $\Gamma_v \subset \Gamma$, die durch einen Switch v verlaufen, durch Knoten repräsentiert. Zwei Knoten $r, s \in \Gamma_v$ des Knotenkonfliktgraphen sind genau dann benachbart, wenn sie in Konflikt zueinander stehen. Der Knotenkonfliktgraph $C_{Knoten}(\Gamma_v) = (V, E)$ ist definiert durch [Now06, Def. 4.4.1]

$$V := \Gamma_v$$
$$E := \{\{r, s\} \in V^{\underline{2}} : r \leftrightsquigarrow_v s\} \ .$$

Die Knoten des Konfliktgraphen stellen also die IRT-Übertragungen der Geräte dar. Jede Kante markiert zwei IRT-Übertragungen, die zueinander in Konflikt stehen. So entsteht ein einfacher schlingenfreier Graph [Now06, Satz 4.4.2], da zwei IRT-Übertragungen nicht mehrfach zueinander in Konflikt stehen können und da eine Übertragung nicht zu sich selbst in Konflikt steht.

Angewandt auf das Beispiel in Abbildung 4.1 weist der Knotenkonfliktgraph für Switch 1 die folgende Struktur auf.

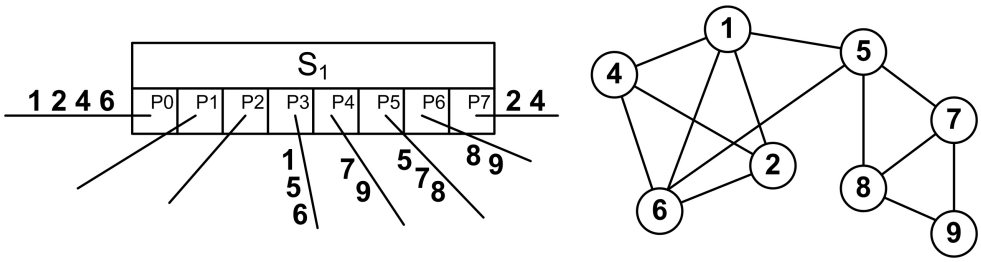


Abbildung 4.4: Switch 1 und sein Knotenkonfliktgraph

Im Gegensatz zu der üblichen Problemstellung der Knotenfärbung ist in diesem Fall aufgrund der Anwendung ein weiteres Faktum bekannt. Alle Kommunikationslinien, welche über denselben Port eines Switches verlaufen, stehen in Konflikt zueinander und bilden somit eine *Clique* im Knotenkonfliktgraphen. Kennt man eine Clique, so lässt sie sich leicht in optimaler Weise knotenfärben, indem man jedem Knoten eine andere Farbe zuweist. Das Auffinden von Cliquen innerhalb eines Graphen ist jedoch ein NP-hartes Problem. In dem Anwendungsfall dieser Arbeit können jedoch eine Vielzahl großer Cliquen durch das einfache Zählen von Kommunikationslinien, welche über einen gemeinsamen Port eines Switches verlaufen, ermittelt werden.

Die größte Clique von Switch 1 befindet sich an Port 0 und definiert eine Untergrenze für die Zahl der benötigten Farben. Die vier Kommunikationslinien 1, 2, 4 und 6 sorgen dafür, dass der Knotenkonfliktgraph auf jeden Fall mindestens vier Farben benötigt, da jede dieser Linien eine eigene Farbe und damit ein eigener Zeitslot zugewiesen werden muss.

Da die IRT-Übertragungen, welche über ein gemeinsames Kabel verlaufen, stets in Konflikt zueinander stehen, können diese Übertragungen mit einer Hyperkante verbunden werden. Abbildung 4.5 zeigt die bekannten Cliquen des beispielhaften Knotenkonfliktgraphen, bei dem es sich um einen Hypergraphen handelt.

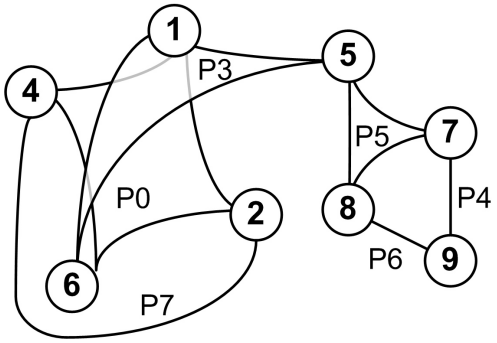


Abbildung 4.5: Der Knotenkonfliktgraph unter Berücksichtigung bekannter Cliquen

Ist die größte Clique bekannt, so ist auch die Untergrenze für die chromatische Zahl $\chi(G)$ bereits durch die Anzahl der an der Clique beteiligten Knoten bekannt. Durch das Abzählen der bekannten Cliquen ist jedoch nicht zwingend die größte Clique bekannt, wie das folgende Gegenbeispiel zeigt. Die Abbildung 4.6 zeigt einen Switch, bei dem maximal zwei Kommunikationslinien über einen Port verlaufen. Es entsteht jedoch eine Clique aus drei Knoten. Die Ursache dafür liegt im Auftreten einer kreisförmigen Kommunikation. Dieser Kreis erzeugt letztlich eine größere Clique als die bislang bekannten Cliquen.

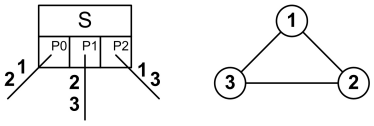


Abbildung 4.6: Kreisförmige Kommunikation

Der Knotenkonfliktgraph kann erstellt werden, indem zunächst für jede Kommunikationslinie ein neuer Knoten im Konfliktgraph angelegt wird. Im Anschluss daran werden die Ports durchlaufen und alle Kommunikationslinien, die über einen Port verlaufen, als *Hyperkanten* dem Graph hinzugefügt. Werden die Konflikte statt dessen als einfache Kanten in den Graphen hinzugefügt, so gehen die Informationen der bekannten großen Cliquen verloren, wodurch ein schlechteres Ergebnis der Färbung zu erwarten ist.

4.1.4.2 Kantenkonfliktgraph

Der lokale Kantenkonfliktgraph wird so aufgebaut, dass die Ports eines Switches die Knoten und die Kommunikationslinien die Kanten des Graphen bilden. Eine einzelne Unicastübertragung verbindet dabei genau zwei Knoten miteinander. Besitzen zwei Kanten einen gemeinsamen Knoten, so stehen sie in Konflikt zueinander und sind daher im weiteren Verlauf unterschiedlich zu färben. Bezogen auf einen Switch ist dies der Fall, wenn zwei Kommunikationslinien mindestens einen gemeinsamen Port verwenden. Es handelt sich dabei um einen ungerichteten *Multigraphen*, da mehrere Kommunikationslinien sowohl

den selben Quellport, als auch den selben Zielport besitzen können. Da es sich nicht um einen einfachen Graphen handelt, muss zusätzlich die Randabbildung $g(r)$ definiert werden, die jeder Kante ihre Randknoten zuordnet. So entsteht der ungerichtete Multigraph $C_{Kanten}(\Gamma_v) = (V, E)$ [Now06, Def. 4.4.3] mit

$V := \mathbb{E}(v)$
$E := \Gamma_v$.
$g(r) : E \mapsto V^{\underline{2}}, r \to E_{KL(r)}(v)$

Der Grad dieses Kantenkonfliktgraphen entspricht der größten bekannten *Clique* des Knotenkonfliktgraphen; im Beispiel ist der Grad $\Delta(C_{Kanten}) = 4$ an Port 0. Angewandt auf das Beispiel in Abbildung 4.1 weist der Kantenkonfliktgraph für Switch 1 die folgende Struktur auf, wobei die Beschriftung der Kanten den Identifikatoren der Übertragungen entspricht.

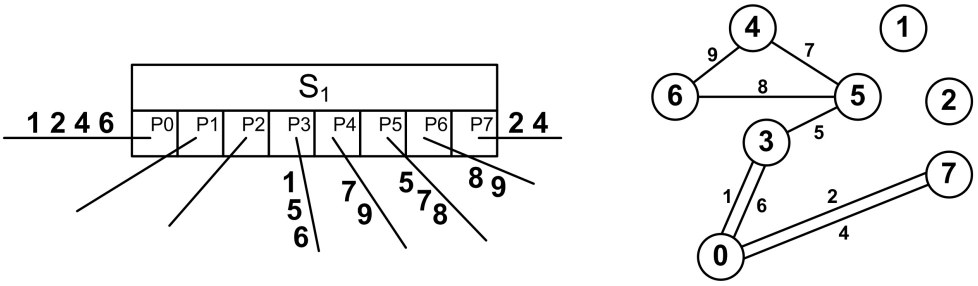


Abbildung 4.7: Switch 1 und sein Kantenkonfliktgraph

Der Graph kann erstellt werden, indem man zunächst für jeden Port eines Switches einen Knoten im Graphen erzeugt. Im Anschluss daran werden die IRT-Übertragungen, welche über diesen Switch verlaufen, als Kanten eingetragen.

4.1.5 Färbung der Konfliktgraphen

In diesem Schritt sollen die Graphen nun gefärbt werden. Dazu existiert bereits eine Vielzahl von bekannten und genügend guten Algorithmen, die in Kapitel 2.3.3.1 zusammengetragen wurden.

4.1.5.1 Knotenfärbung

Bei der Färbung des einfachen Knotenkonfliktgraphen ist dann ein Ergebnis von verminderter Güte oder erhöhter Laufzeit im Gegensatz zum Kantenkonfliktgraphen zu erwarten, wenn die Information über die großen bekannten Cliquen nicht in dem Algorithmus der Färbung berücksichtigt wird. Findet man einen solchen Algorithmus oder passt man einen Standard-Färbealgorithmus entsprechend an, so dass er die Cliquen berücksichtigt, so ist der Graph schneller oder als Alternative dazu in gleicher Laufzeit besser zu färben als bei der Betrachtung des einfachen Graphen. Heuristisch ist es sinnvoll, zunächst die größten Cliquen zu färben und die restlichen Knoten mit freien Farben aus der Menge der bereits benutzten Farben zu versehen. Der gefärbte Knotenkonfliktgraph des Beispiels ist in Abbildung 4.8 skizziert, wobei die Knoten unter Verwendung eines Greedy-Algorithmus

mit der Reihenfolge der Kommunikationslinien-IDs als Vorsortierung mit den Farben F_i gefärbt wurden.

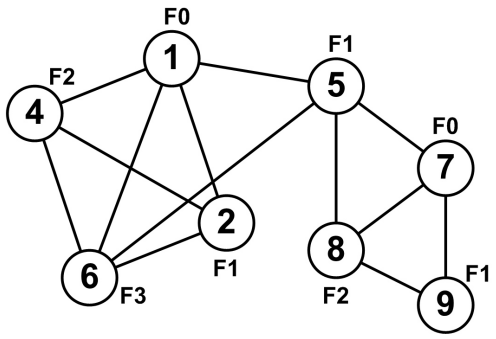


Abbildung 4.8: Greedy-gefärbter Knotenkonfliktgraph

4.1.5.2 Kantenfärbung

Nach dem *Satz von Vizing* besteht eine Untergrenze für die Kantenfärbung eines Multigraphen in dem Grad des Graphen, der im Beispiel in Abbildung 4.9 $\Delta(C_{Kanten}) = 4$ beträgt. Als eine Obergrenze wird von Vizing $\Delta(C_{Kanten}) + d$ angegeben, so dass sich als Obergrenze für den chromatischen Index des Beispiels 6 Farben ergibt, da die Multiplizität $d = 2$ beträgt. Die einfache Färbeheuristik *Greedy-Seq-Edgecolor* besitzt eine Komplexität von $O(|E| \cdot \Delta(C_{Kanten}))$ und erzeugt eine Färbung mit maximal $2 \cdot \Delta(C_{Kanten}) - 1$ Farben [Now06].

Um die Verarbeitungszeit und die praktische Güte dieses Algorithmus zu testen, wurden im Rahmen einer Softwaresimulation mit MS Visual Basic 6.0 exemplarisch Switches mit 4, 5, 6, 8, 12, 16, 24 und 32 Ports generiert. Über jeden Switch wurden dann 500, 1000, 2000, 4000, 8000 und 16000 zufällig erzeugte Kommunikationslinien gelegt. Für jede Kombination aus Anzahl an Ports und Kommunikationslinien wurden jeweils 32 Messungen der benötigten Rechenzeit für die Kantenfärbung der resultierenden lokalen Konfliktgraphen vorgenommen und der Mittelwert gebildet. Die Messungen wurden durchgeführt mit einem Acer TravelMate 250 Laptop mit Intel Pentium 4 Prozessor, $2.4GHz$ und $512MB$ RAM. Die Ergebnisse wurden in [DW06] veröffentlicht. Die Abbildung 4.9 zeigt, dass der Kantenkonfliktgraph von 4000 Kommunikationslinien in ca. 5 Sekunden auf einem Standard-Laptop mit Greedy-Seq-Edgecolor gefärbt werden kann.

In der Realität ist das Vorhandensein von 4000 Kommunikationslinien über einen einzelnen Switch bereits unrealistisch, da unter der Berücksichtigung der Übertragungszeit eines einzelnen minimalen Ethernet-Frames die Zykluszeit unverhältnismässig hoch wäre. Auffallend ist jedoch, dass die Färbung mit 16000 Kommunikationslinien in einem 32-Port Switch in ca. 8 Sekunden berechnet wird, während die gleiche Anzahl in einem 4-Port Switch fast 75 Sekunden benötigt. Der Grund für die Abnahme liegt darin, dass bei konstanter Anzahl der Kommunikationslinien, welche auf mehr Ports verteilt werden, die Dichte des Graphen sinkt. Der Grad $\Delta(C_{Kanten})$ reduziert sich also, wodurch die nächste freie Farbe schneller ermittelt werden kann. Die Abbildung 4.10 zeigt nochmals, wie die benötigte Zeit mit der Anzahl der Ports sinkt.

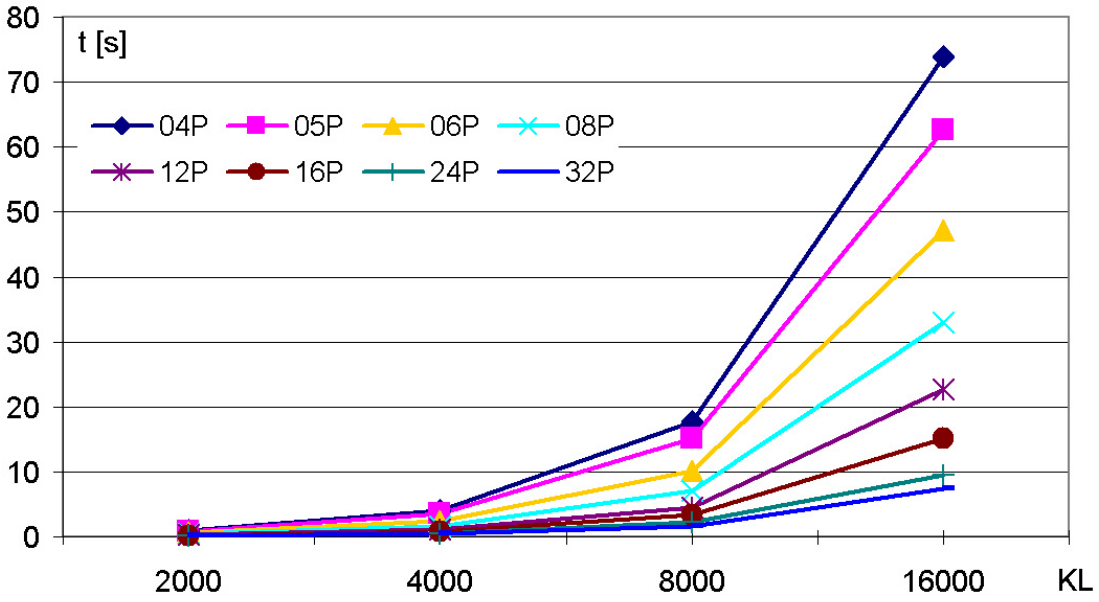


Abbildung 4.9: Rechenzeit in Abhängigkeit der Ports und Kommunikationslinien

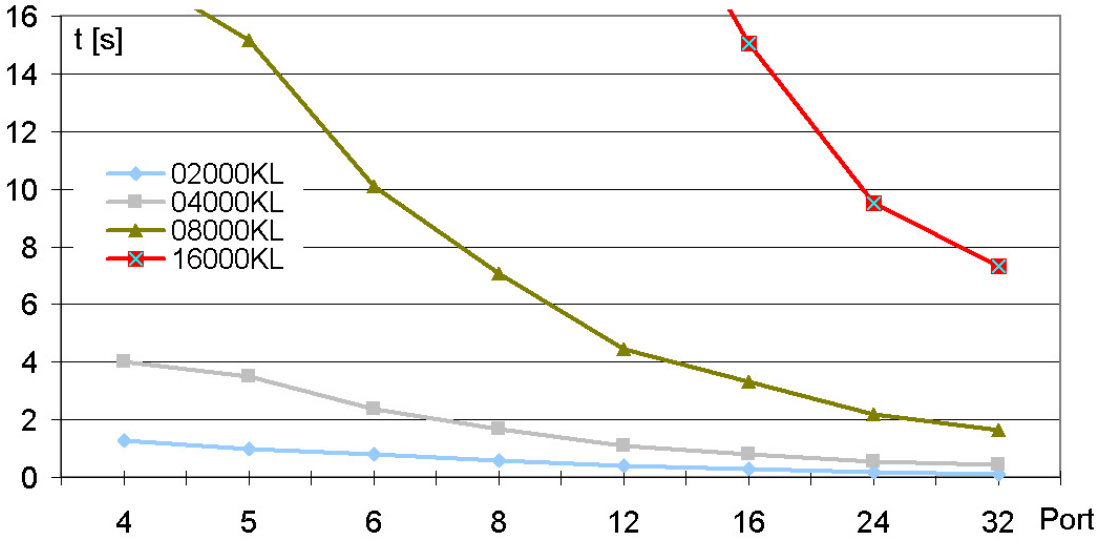


Abbildung 4.10: Abnehmende Zeit bei konstanter Anzahl der Kommunikationslinien

Neben der Laufzeit ist insbesondere die Güte der ermittelten Lösung zu prüfen. Nach dem Satz von Vizing kann der chromatische Index eines kantenzufärbenden Multigraphen nicht geringer sein als dessen Grad. Ist der betreffende Konfliktgraph schwierig zu färben, so kann die optimale Färbung auch deutlich über dem Grad liegen. Ein Graph gilt unter anderem dann als schwierig zu färben, wenn er eine große Anzahl von Kreisen mit ungerader Anzahl an Knoten enthält.

Zu diesem Zweck wird in Abbildung 4.11 der Grad des Graphen von der ermittelten Anzahl der Farben abgezogen. Auf diese Weise bleibt die Anzahl der Farben übrig, welche über dem Grad des Konfliktgraphen liegen. Die erstellten Färbungen lagen alle nahe der unteren Grenze $\Delta(C_{Kanten})$. Die durchschnittliche Multiplizität $\mu(G)$ der erzeugten

Graphen lag beispielsweise bei den 4-Port Switches mit 8000 Kommunikationslinien bei 2663, der durchschnittliche Grad der Kantenkonfliktgraphen bei 5102. Im Durchschnitt wurden jedoch nur 10 Farben mehr verwendet als der jeweilige durchschnittliche Grad der Graphen beträgt.

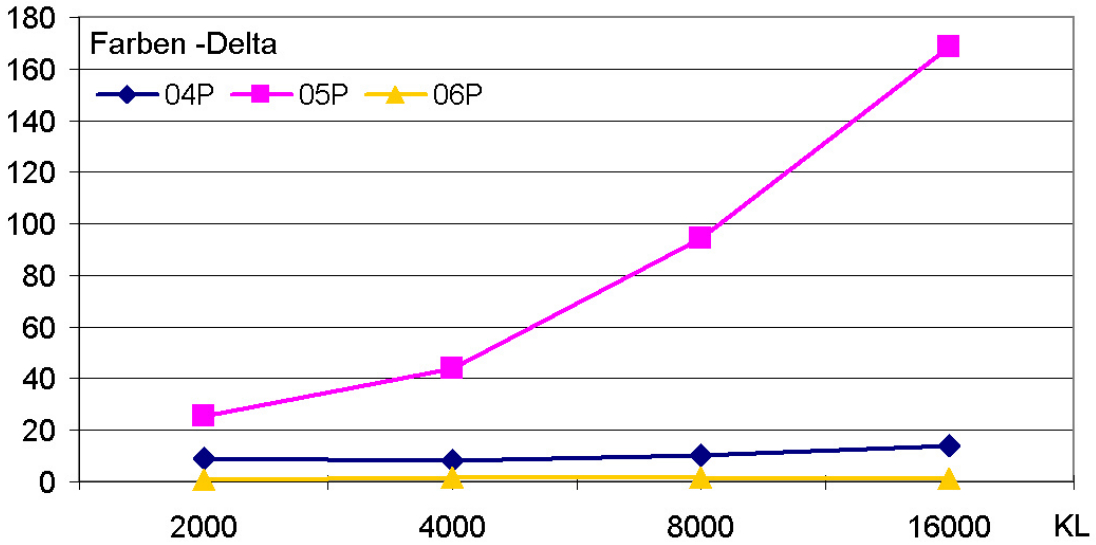


Abbildung 4.11: Güte der Lösungen

Diese Abbildung zeigt, dass der einfache Greedy-Algorithmus bereits sehr gute Ergebnisse liefert. Bei Switches mit mehr als 6 Ports entspricht die Färbung nahezu dem Grad des Graphen. Diese Werte sind in der Abbildung nicht mehr dargestellt. Jedoch ist zu bedenken, dass es sich um zufällig generierte Graphen handelt. Es lassen sich ungünstige Graphen konstruieren - z. B. unter Verwendung von Kreisen von hoher ungerader Knotenzahl - und die Reihenfolge der Kanten so ungünstig anordnen, dass der Greedy-Algorithmus eine nahezu beliebig schlechte Färbung erzeugt. Auffallend ist das Ergebnis bei 5-Port Switches. Durch die ungerade Anzahl der Ports und der wenigen möglichen Kombinationen ist hier die Wahrscheinlichkeit für die Erzeugung von ungünstigen Graphen höher. Innerhalb einer automatisierten Anlage kann es natürlich vorkommen, dass beispielsweise von einem 8-Port Switch lediglich 5 Ports häufig verwendet werden, deren Anforderungen sich gegenseitig überlappen, so dass Kreise im Graphen mit ungerader Knotenzahl entstehen. Zusammenfassend ist jedoch zu sagen, dass der einfache Greedy-Algorithmus bereits gute Resultate liefert. Zu bedenken ist auch, dass Lösungen mit vielen Farben (>100) im Rahmen des echtzeitfähigen Ethernets gar keine Rolle spielen werden, da die resultierende Schedule und damit die resultierende Zykluszeit viel zu lang ist, um eine harte Echtzeitklasse der IAONA erfüllen zu können (vgl. Kapitel 2.1.3.3).

Zur Verbesserung der Güte können bessere Algorithmen, welche in der Regel komplexer und mit einer höheren Laufzeit verbunden sind, verwendet werden. Eine sinnvolle Heuristik kann darin bestehen, zunächst alle Switches nacheinander schnell mittels einem Greedy-Algorithmus zu färben. In einem zweiten Schritt können kritische Switches, welche zu diesem Zeitpunkt die meisten Farben verwenden und somit bei der Erstellung der Schedule die Zykluszeit erhöhen, mit besseren Algorithmen wie der DSATUR-Heuristik (vgl. Kapitel 2.3.3.3) oder gar mit exakten Algorithmen (vgl. Kapitel 2.3.3.1) neu gefärbt werden. Denn

gerade bei einer geringen Anzahl an IRT-Übertragungen sind auch nicht-polynomielle Algorithmen von Interesse.

Die abschließende Abbildung dieses Unterkapitels zeigt den Kantenkonfliktgraphen für Switch 1 des Beispiels nach der Anwendung der Greedy-Färbung. Auch hier wurde die Reihenfolge der Kommunikationslinien-IDs als Vorsortierung verwendet und die ermittelten Farben F_i an die Kanten neben den Kommunikationslinien-IDs notiert. In diesem einfachen Fall entspricht die Färbung dem Grad des Graphen.

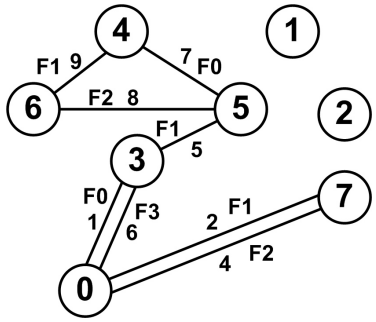


Abbildung 4.12: Greedy-gefärbter Kantenkonfliktgraph

4.1.5.3 Vergleich der Färbungen

Sowohl bei der Knoten-, als auch bei der Kantenfärbung dieses Halbduplex-Szenarios treten allgemeine ungerichtete Konfliktgraphen auf, deren exakte Färbung NP-hart ist. Auffallend ist, dass der Kantenkonfliktgraph mit weniger Knoten und Kanten die gleiche Problematik modelliert. Dies ist nicht nur in dem Beispiel der Fall, da es sich bei den Knoten im Kantenkonfliktgraph um die feste Anzahl der Ports des Switches handelt. Diese sind in der Praxis auf 64 beschränkt, da Switches mit mehr als 64 Ports unüblich sind. Im Knotenkonfliktgraphen würden auf diese Weise 64 Cliquen bekannt sein, wobei diese Kenntnis algorithmisch schwieriger auszunutzen ist. Dazu müssten die Standard-Algorithmen angepasst bzw. erweitert werden.

Die Anzahl der Kanten im Kantenkonfliktgraph entspricht der Anzahl der Knoten im Knotenkonfliktgraph. In der Praxis kann man von $\Gamma_v >> Ports(v)$ ausgehen, also dass über einen Switch wesentlich mehr Kommunikationslinien verlaufen als er Ports besitzt. Im Falle des Kantenkonfliktgraphen führt dies zu einem Graphen mit wenigen Knoten und sehr vielen Kanten, also zu einem übervollen Graphen. Alternativ dazu besitzt der entsprechende Knotenkonfliktgraph sehr viele Knoten - nämlich $|\Gamma_v|$ - mit wenigen, wahrscheinlich sehr großen Cliquen.

Da der Multigraph der Kantenfärbung die Informationen über große Cliquen bereits in dem Grad jedes Knoten enthält und er bereits mit schnellen Algorithmen in ausreichender Güte gefärbt werden kann, ist die Kantenfärbung für diesen Fall als besserer Ansatz anzusehen.

4.1.6 Erstellung der lokalen Schedules

Eine Schedule für einen Switch v bezieht sich auf eine Menge von IRT-Übertragungen $\Gamma_v \subset \Gamma$ und ist eine Abbildung, welche der IRT-Übertragung $r \in \Gamma_v$ einen Zeitpunkt $\alpha(r)$ zuordnet, zu dem die Ausführung der Kommunikation beginnt. Da alle IRT-Übertragungen eine einheitliche Framegröße und dieselbe Übertragungsdauer haben, kann man das Zeitintervall $\hat{\alpha}(r)$ der Ausführung einer IRT-Übertragung mit $\hat{\alpha}(r) = 1$ belegen. Dieses Zeitintervall wird im Folgenden als *Aktivitätsintervall* der Übertragungen bezeichnet. Den IRT-Übertragungen kann man dann als Startzeit nicht-negative ganze Zahlen zuordnen. Sind $r, s \subset \Gamma_v$ Übertragungen, die über den Switch v verlaufen, so ist demnach die Abbildung $\alpha : \Gamma_v \mapsto \mathbb{N}_0$ genau dann eine Schedule für Γ_v, wenn $\alpha(r) \neq \alpha(s)$ gilt, sofern $r \leftrightsquigarrow_v s$. Die IRT-Übertragungen r und s werden also nicht im selben Zeitslot ausgeführt, falls sie in Konflikt zueinander stehen. Ist α eine Schedule für Γ_v, so ist $|\alpha| := max\,\{\alpha(r) + 1 : r \in \Gamma_v\}$ die Zyklenlänge der Schedule. Eine solche Schedule wird als natürliche Schedule bezeichnet [Now06, Def. 6.1.1].

Die Schedule für einen Switch aus der Netzwerkinfrastruktur lässt sich leicht erzeugen, sofern der gefärbte Konfliktgraph gegeben ist [Now06, Satz 6.3.1/2]. Man durchläuft alle gefärbten Kanten bzw. Knoten und sortiert die Anforderungen mit gleicher Farbe F_i jeweils in einen Vektor. Aufgrund der Beschaffenheit des Konfliktgraphen sind gleich gefärbte Elemente im Konfliktgraphen nicht benachbart und stehen somit nicht zueinander in Konflikt. Eine optimale Schedule für den Switch v hat genau die Länge, wie sie aus dem chromatischen Index eines Kantenkonfliktgraphen bzw. aus der chromatischen Zahl eines Knotenkonfliktgraphen resultiert.

Abbildung 4.13 zeigt die beiden gefärbten Konfliktgraphen und die resultierende Schedule für Switch 1 aus dem Beispiel. Dass sich in beiden Fällen die gleiche Schedule ergibt, ist darin begründet, dass die Kommunikationslinien in beiden Graphen auf die gleiche Weise indiziert wurden. Dies führt zu einer identischen Reihenfolge der zu färbenden Knoten bzw. Kanten.

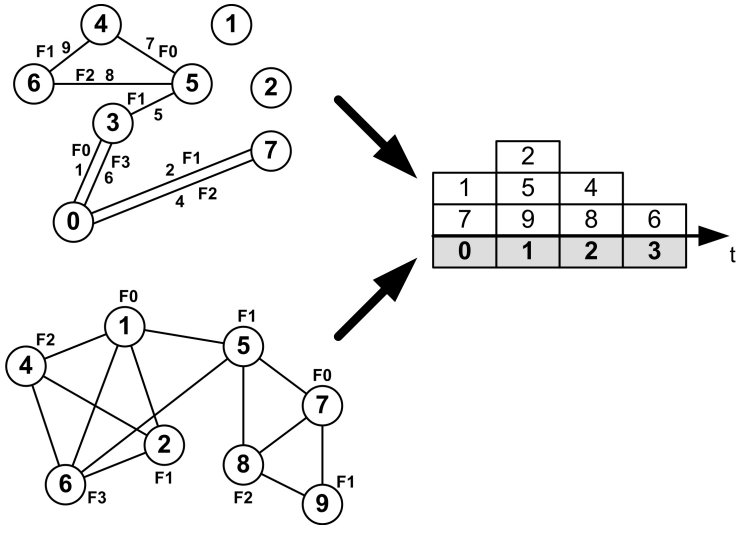


Abbildung 4.13: Konfliktgraphen und deren Überführung in eine Schedule

4.1.7 Synchronisation der Schedules

Bislang wurde gezeigt, wie aus vorgegebenen Unicastübertragungen, die über mehrere Switches des Netzwerkes verlaufen, Schedules für einzelne Switches erstellt werden.

Es bleibt zu zeigen, dass diese unabhängige Betrachtung der einzelnen Knoten überhaupt korrekt ist. Denn dass für einzelne Switches Schedules existieren, garantiert noch nicht zwangsläufig die Funktionalität des Gesamtnetzwerkes. Um dies zu erreichen, müssen die Schedules zueinander synchron sein. Dies ist bei einer unabhängigen Färbung bereits dann nicht der Fall, wenn bei einem Greedy-Algorithmus verschiedene Vorsortierungen für verschiedene Switches gewählt werden, um die Zykluszeit zu optimieren. Es kann beispielsweise sinnvoll sein, bei Switches mit einer großen Anzahl von Kommunikationslinien den DSATUR-Algorithmus zu wählen, um die *lokale Schedule* zu optimieren. Dadurch können den Komunikationslinien andere Farben zugewiesen werden, wodurch sie in einem anderen Zeitslot liegen. Abbildung 4.14 zeigt die unsynchronisierten Schedules des Beispiels.

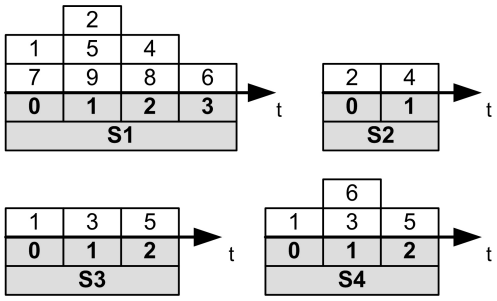


Abbildung 4.14: Unsynchronisierte Schedules

Wie man bereits in dem einfachen Beispiel erkennt, bekommen die IRT-Übertragungen 2 und 4 in Switch 1 und Switch 2 andere Zeitslots zugewiesen. Des Weiteren wird die Kommunikationslinie Nummer 6 in Switch 1 im vierten und in Switch 4 im zweiten Zeitslot ausgeführt. In Switch 4 wird zeitgleich die Kommunikationslinie Nummer 3 verarbeitet.

Um eine Synchronisation zu erreichen, müssen die einzelnen Slots in einer Weise permutiert werden, dass die gleichen Übertragungen in den gleichen Slots ausgeführt werden. Zu diesem Zweck wird die Umindizierung [Now06, Def. 6.1.4] auf der Basis einer gegeben Schedule $\alpha : \Gamma_v \mapsto \underline{n}$ definiert. Sei $\rho : \underline{n} \mapsto \underline{m}$ eine injektive Abbildung, dann ist $\rho \circ \alpha$ eine Schedule für Γ_v mit der Länge $|\rho \circ \alpha| \leq m$. Die Schedule $\rho \circ \alpha$ entsteht also aus α durch Umindizierung mit ρ.

Zwei Schedules $\alpha : \Gamma_v \mapsto \underline{n}$ und $\beta : \Gamma_w \mapsto \underline{m}$ der Switches $V, W \in \mathbb{V}$ sind synchronisierbar, wenn es injektive Abbildungen $\rho : \underline{n} \mapsto \mathbb{N}_0$ und $\psi : \underline{m} \mapsto \mathbb{N}_0$ gibt, so dass $\rho \circ \alpha$ synchron zu $\psi \circ \beta$ ist.

Im Rahmen der Forschungstätigkeit wurde bewiesen [Now06, Def. 6.2.1], dass in Halbduplex-Netzwerken je zwei Schedules von benachbarten Switches durch die Vertauschung ganzer Zeitslots stets synchronisierbar sind. Die Synchronisation ist damit in polynomieller Zeit möglich. Der folgende Algorithmus synchronisiert zwei direkt verbundene Switches $S1$ und $S2$:

1. Wähle den Switch mit den meisten Zeitslots als Wurzelknoten W. Besitzen beide Schedules die gleiche Anzahl an Zeitslots, so kann ein beliebiger Switch als Wurzel verwendet werden. Der Switch, der nicht die Wurzel darstellt, wird im Folgenden als $S2$ bezeichnet.

2. Die Schedule der Wurzel α_W ist endgültig. Die Verbindung von $S2$ mit W kann nur über genau ein Kabel erfolgen. Die Kommunikationslinien, welche sowohl in α_W, als auch in α_{S2} auftreten, können nur über dieses eine Kabel verlaufen. Diese Anforderungen stehen jedoch alle in Konflikt zueinander und können weder in W, noch in $S2$ in gemeinsamen Zeitslots auftreten.

3. Ordne die Zeitslots, welche in W und $S2$ auftreten, so an, dass sie in $S2$ den gleichen Index besitzen wie in W. Dazu werden die IRT-Übertragungen von $S2$ durchlaufen. Ist eine gemeinsame IRT-Übertragung mit W gefunden, so kann der gesamte Zeitslot von $S2$ verschoben werden. Außerdem können alle IRT-Übertragungen, die in diesem Zeitslot enthalten sind, als bereits durchlaufen markiert werden.

4. Bei dem Vorgehen in 3. treten freie Zeitslots in $S2$ auf. Diese werden mit IRT-Übertragungen gefüllt, welche nur in $S2$ und nicht in W auftreten. Ist die Zykluszeit von $S2$ im Anschluss daran kürzer als die Zykluszeit von W, so wird sie mit freien Slots aufgefüllt.

Der Algorithmus kann nun im Anschluss an die unabhängige Färbung der lokalen Konfliktgraphen der Switches angewendet werden. Der Switch mit der längsten Schedule wird als Wurzel gewählt und der Baum der Netzwerk-Infrastruktur in einer Breiten- oder Tiefen-Suche durchlaufen. Dabei werden die Schedules der Switches wie beschrieben synchronisiert. Nowak [Now06] hat diesen Algorithmus formalisiert und eine Laufzeitkomplexität von $O(max|\Gamma_W|, n)$ für die Synchronisierung von zwei Schedules bestimmt mit den IRT-Übertragungen über den Wurzel-Switch $\Gamma_W \subset \Gamma$ und $\alpha : \Gamma_W \mapsto \underline{n}$.

Nach der Anwendung dieses Algorithmus sind die Schedules aus dem Beispiel wie in Abbildung 4.15 synchronisiert. Alle Switches besitzen die gleiche Anzahl an Zeitslots, aus der die Zykluszeit der Anlage resultiert. In diesem idealisierten Fall ist dies das Vierfache der zu versendenden, einheitlichen Paketgröße. Es entstehen freie Zeitslots - zum Beispiel Slot 0 und 3 im zweiten Switch. In diesen Slots werden keine Kommunikationslinien mit isochronen Echtzeitanforderungen über diesen Switch übertragen.

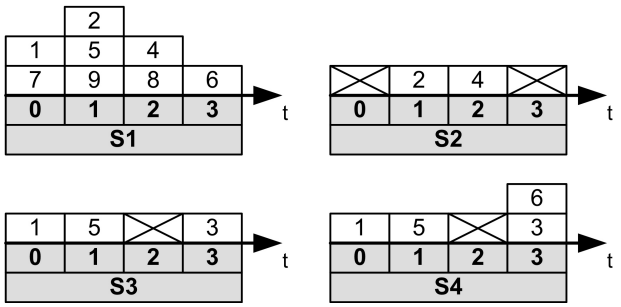


Abbildung 4.15: Synchronisierte Schedules

4.1.8 Zusammenfassung

In diesem Kapitel wurde ein erster Ansatz der Formalisierung der Infrastruktur des Netzwerkes und der Übertragungen aufgezeigt. Über die Erstellung der ungerichteten Konfliktgraphen und deren Färbung können die Schedules für die einzelnen Switches unabhängig voneinander offline berechnet werden. Abbildung 4.16 fasst die Vorgehensweise bis zur Erstellung der synchronen Schedules zusammen.

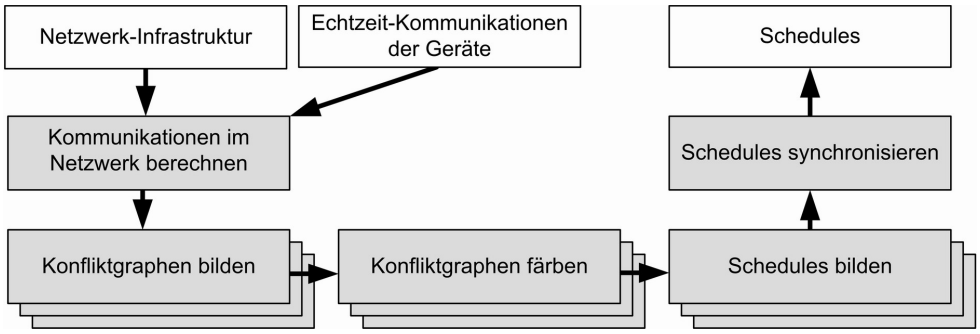


Abbildung 4.16: Vorgehensweise zur Erstellung synchroner Schedules

Bis auf die Färbung der Graphen können alle Algorithmen in polynomieller Zeit ausgeführt werden. Im Falle der Knotenfärbung sind bereits große Cliquen bekannt, die man bei der Anwendung der Färbungsalgorithmen berücksichtigen sollte. Bei der Kantenfärbung ist mit einem übervollen Multigraphen zu rechnen. Die exemplarische Anwendung einfacher Färbungsheuristiken lieferte bereits gute Ergebnisse. Die längsten Schedules können bei Bedarf mit aufwendigeren Heuristiken vor der Synchronisation zur Optimierung der Zykluszeit nachberechnet oder sogar mit exakten Algorithmen optimiert werden.

4.2 Idealisierte Vollduplexübertragung

Als erste Einschränkung der Modellierung wird die Halbduplexübertragung aufgehoben. Wie in Kapitel 2.2.2.1 beschrieben, kann im heute üblichen Ethernet auf einem Kabel im Vollduplexmodus gleichzeitig gesendet und empfangen werden. Im 100 BaseTX-Standard ist dies durch das Vorhandensein von zwei Leitungspaaren realisiert.

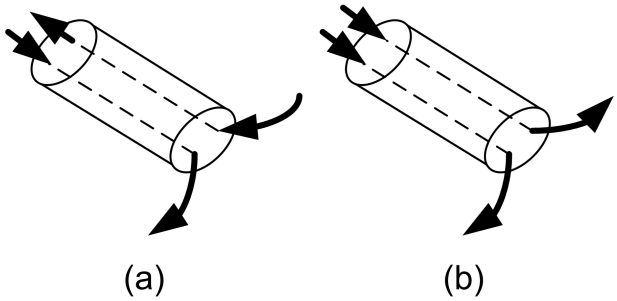


Abbildung 4.17: Erlaubte (a) und konfliktauslösende (b) Kommunikation

Die physikalische Verkabelung der Netzwerktopologie bleibt ebenso unverändert wie die Modellierung der Switches und der Geräte. Die Definition einer Kommunikationslinie kann auch unverändert übernommen werden, ebenso der Algorithmus zur Berechnung von Kommunikationslinien, da er die Richtung des Weges beim Zusammensetzen der Teilwege zur Wurzel nicht verändert.

Da ein Kabel zur Vermeidung eines Konfliktes im Vollduplex-Fall nicht mehr exklusiv für eine Kommunikation verwendet werden muss, ist der Begriff des Konfliktes zu spezialisieren. Verläuft mehr als eine Kommunikationslinie zu einem Zeitpunkt in eine Richtung, so löst sie wie im Halbduplex-Fall einen Konflikt aus. Der CSMA/CD-Algorithmus findet zwar keine Anwendung mehr, jedoch werden Frames in den Switches nicht-deterministisch und verkehrsabhängig gepuffert und im Falle der Überlast sogar verworfen.

4.2.1 Definition eines Konfliktes

Zwei Kommunikationslinien $K, L \in \Gamma$ in einem Vollduplexnetzwerk stehen dann in einem Konflikt zueinander, wenn sie über mindestens ein gleiches Kabel $e \in E_K \cap E_L$ verlaufen mit $e \in \mathbb{E}$ und die Richtung identisch ist, d. h. entsprechend der Bezeichnung für Wege (s. Kapitel 2.3.2) $e^{K-} = e^{L-}$ bzw. $e^{K+} = e^{L+}$ gilt. Innerhalb eines Switches $v \in \mathbb{V}$ tritt dann ein Konflikt auf, wenn ein Konflikt an einem angeschlossenen Kabel vorliegt, also $(\exists e E(v) : K \nleftrightarrow_e L)$ gilt.

Die Aussagen für Halbduplexnetzwerke, dass eine lokale Konfliktfreiheit eine globale Konfliktfreiheit impliziert [Now06, Satz 4.2.4] und dass Konfliktknoten stets einen Weg bilden [Now06, Satz 4.2.6], trifft auch für Vollduplexnetzwerke zu. Dies ist in dem veränderten Konfliktbegriff begründet. Treffen sich zwei Kommunikationslinien an einem Port eines Switches, so stehen sie nur dann in Konflikt, wenn sie in die identische Richtung verlaufen. Ist dies nicht gegeben, so stehen diese beiden Kommunikationslinien an keiner anderen Stelle des Baumes der Netzwerkinfrastruktur in Konflikt.

Stehen die beiden Kommunikationslinien jedoch in Konflikt, so verlaufen sie so lange in die identische Richtung, wie sie über gemeinsame Kabel im Netzwerk verlaufen. Im Anschluss daran besteht aufgrund der Baumtopologie keine Möglichkeit, dass sich diese beiden Kommunikationslinien nochmals beeinflussen.

4.2.2 Konfliktgraphen und deren Färbung

Für diesen Fall wird das Beispiel aus Abbildung 4.1 weiter verwendet, jedoch spielt nun die Richtung der Übertragungen eine Rolle. Diese werden in Abbildung 4.18 exemplarisch für Switch 1 berücksichtigt.

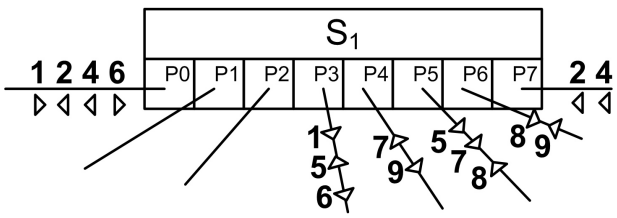


Abbildung 4.18: Übertragungen von Switch 1 im Vollduplexmodus

Zunächst soll der Kantenkonfliktgraph erstellt werden. Ein erster Ansatz kann dabei sein, den ungerichteten Graph aus Kapitel 4.1 in einen gerichteten Graph umzuwandeln und dann einen für gerichtete Graphen modifizierten Färbealgorithmus anzuwenden. Dieser Ansatz ist jedoch nicht effizient, da die Färbung NP-vollständig bleibt und einen modifizierten Färbealgorithmus bedingt.

Eine interessante Idee besteht darin, auf die Modellierung der Ports eines Switches zurückzugreifen. Jeder Port besitzt bei Vollduplexfähigkeit einen unabhängigen Eingangs- und Ausgangsteil, der als $V_{in} = \{e_{in} : e_{in} \in \mathbb{E}(v)\}$ bzw. $V_{out} = \{e_{out} : e_{out} \in \mathbb{E}(v)\}$ modelliert wird. Aufgrund der neuen Konfliktdefinition stehen zwei Kommunikationslinien nur dann in Konflikt, wenn sie einen gemeinsamen Teil eines Ports verwenden. Daher wird der lokale Kantenkonfliktgraph $C_{Kanten}(\Gamma_v) = (V, E, g)$ wie folgt definiert [Now06, Def. 4.4.6]:
$V := V_{in} \cup V_{out}$
$E := \Gamma_v$
$g(r) : E \mapsto V^{\underline{2}}, r \rightarrow \left\{ v_{in}^{KL(r)-}, v_{out}^{KL(r)+} \right\}, r \in \Gamma_v$

Jede Kommunikation verläuft also von genau einem Eingangsport zu genau einem Ausgangsport. Der Graph besitzt die gleiche Menge an Kanten wie im Halbduplex-Fall, da die Anzahl der IRT-Übertragungen konstant geblieben ist. Durch die Aufspaltung der Ports des Switches in Eingangs- und Ausgangsports hat sich die Anzahl der Knoten des Konfliktgraphen verdoppelt. Es handelt sich wieder um einen ungerichteten Multigraphen, da mehrere Kommunikationslinien sowohl den selben Quellport, als auch den selben Zielport in der gleichen Richtung durchlaufen können. Abbildung 4.19 zeigt den Kantenkonfliktgraph des Beispiels.

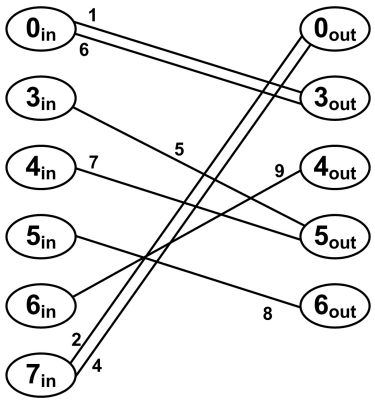


Abbildung 4.19: Kantenkonfliktgraph von Switch 1

Der Kantenkonfliktgraph eines Vollduplexnetzwerkes ist *bipartit* [Now06, Satz 4.4.8], so dass die Färbung erheblich vereinfacht wird. Zunächst ist bereits bei der Entstehung des Konfliktgraphen bekannt, dass er mit $\chi'(G) = \Delta(G)$ gefärbt werden kann. Auf diese Weise kann im Gegensatz zum Halbduplex-Fall vor der Färbung bereits der Switch definitiv ermittelt werden, der für die meisten Zeitslots der Schedule verantwortlich sein wird. In Kapitel 2.3.3.1 wurden bereits Ansätze zur Kantenfärbung bipartiter Multigraphen vorgestellt. Ein einfacher optimaler $O(|E| \cdot log(|E|))$-Algorithmus stammt von Alon [Alo03] aus dem Jahre 2003.

Der Aufbau eines Knotenkonfliktgraphen kann im Vollduplexmodus beibehalten werden, da die Knoten immer noch durch die IRT-Übertragungen der Geräte Γ_v und die Kanten durch die Konflikte repräsentiert werden. Dadurch entsteht wiederum ein einfacher ungerichteter Graph. Die Cliquen können algorithmisch aufgrund der neuen Definition eines Konfliktes nicht mehr einfach durch die IRT-Übertragungen, welche gemeinsam über einen Port verlaufen, ermittelt werden. Statt dessen werden alle Kommunikationslinien, welche über einen Port verlaufen und die gleiche Richtung besitzen, einer Clique zugeordnet und dem Konfliktgraphen hinzugefügt. Abbildung 4.20 zeigt den resultierenden Konfliktgraphen des Beispiels.

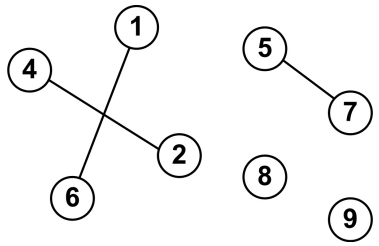


Abbildung 4.20: Knotenkonfliktgraph von Switch 1

Aufgrund der Vollduplexfähigkeit des Switches reduziert sich die Anzahl der Konflikte und durch die Äquivalenz zwischen Kanten- und Knotenfärbung ist anzunehmen, dass es sich bei dem Konfliktgraphen ebenfalls um einen leicht zu färbenden Graphen handelt. Diese Vermutung wird durch das folgende Beispiel unterstützt.

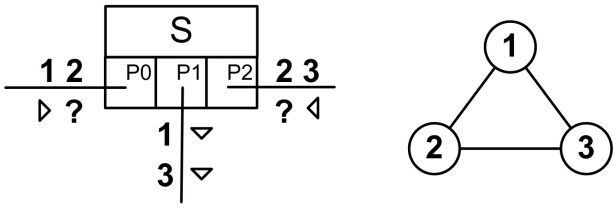


Abbildung 4.21: Problem des ungeraden Kreises

Obwohl es im Rahmen dieser Arbeit nicht formal bewiesen wird, scheint aufrund der Problemstellung die Erzeugung eines Kreises mit einer ungeraden Anzahl an Knoten im Knotenkonfliktgraphen nicht möglich zu sein. Die zweite Kommunikationslinie kann nicht so verlaufen, dass sie sowohl mit der ersten, als auch mit der dritten Linie in Konflikt steht. Die Existenz solcher Kreise deuten in der Graphentheorie auf einen schwierig zu färbenden Graphen hin.

Das folgende Beispiel zeigt, dass es Existenz von großen - auch ungeraden - Cliquen durchaus möglich ist. Sie entstehen, wenn mehrere Kommunikationslinien in der selben Richtung über einen Port verlaufen.

Aufgrund der Problemstellung können jedoch nicht wie in Abbildung 4.22 dargestellt, größere Cliquen als die bislang bekannten Cliquen entstehen. Dies bedeutet gleichzeitig, dass die größte existierende Clique für diesen Fall bereits im Vorfeld bekannt ist und daher auch dieser Graph leicht zu färben ist.

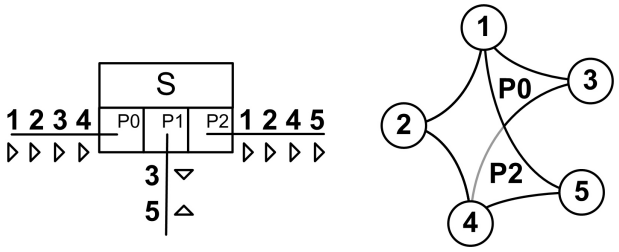
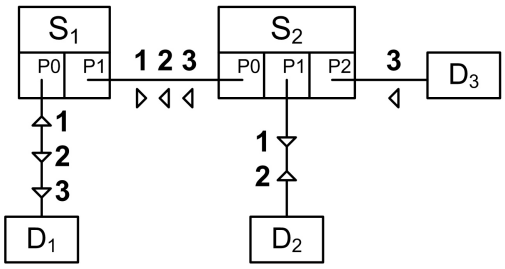


Abbildung 4.22: Bekannte Cliquen im Vollduplex-Konfliktgraph

4.2.3 Lokale Schedules und Synchronisierung

Die einzelnen Schedules aus den gefärbten Graphen lassen sich genauso erzeugen wie im Halbduplex-Fall, da die Beziehung zwischen den gefärbten Graphen und den Schedules beibehalten wird. Dies gilt jedoch nicht für die Synchronisierbarkeit von unabhängig erzeugten Schedules [Now06, Satz 6.2.2]. Das Problem bei der Synchronisierung liegt darin, dass ein Kabel zu einem Zeitpunkt im Vollduplexmodus nicht mehr exklusiv verwendet wird. Abbildung 4.23 zeigt ein einfaches Beispiel mit zwei Switches und drei Anforderungen von Geräten.



Γ-ID	Quelle	Ziel
1	1	2
2	2	1
3	3	1

Abbildung 4.23: Beispiel zur Nicht-Synchronisierbarkeit

Für dieses Beispiel werden nun die Knotenkonfliktgraphen gebildet, die anschließend gefärbt werden. Abbildung 4.24 illustriert außerdem die beiden erstellten Schedules aus den Färbungen.

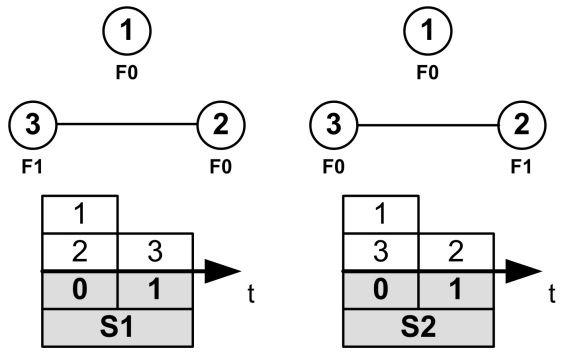


Abbildung 4.24: Nicht-synchronisierbare Knotenkonfliktgraphen und deren Schedules

Die beiden Knotenkonfliktgraphen sind zufälligerweise identisch, jedoch muss der dritten Kommunikationslinie bei einer unabhängigen Färbung der einzelnen Graphen nicht immer die gleiche Farbe zugeordnet werden. Dies kann aufgrund der Anwendung von unterschiedlichen Färbealgorithmen der beiden Switches oder nur aufgrund einer unterschiedlichen Vorsortierung der Eingangsknoten in den Greedy-Algorithmus geschehen. Im Gegensatz zu Halbduplexnetzwerken sind die beiden Schedules der Switches nicht durch ein einfaches Austauschen ganzer Zeitslots synchronisierbar. Beim Austauschen einzelner IRT-Übertragungen müssen wieder alle Permutationen betrachtet werden, so dass ein NP-hartes Problem entsteht. Um dennoch eine für das Gesamtnetzwerk gültige Schedule zu erhalten, sind zwei Verfahren denkbar.

Das erste Verfahren besteht darin, zunächst alle Kantenkonfliktgraphen zu erzeugen und zu färben. Ein Switch mit der größten Anzahl an Farben stellt dann den Wurzel-Switch w für die weitere Färbung dar. Treten IRT-Übertragungen auch in Switches auf, die direkt mit w verbunden sind, so erhalten sie jeweils dieselbe Farbe, wie sie in w besitzen. Im Anschluss daran werden die restlichen Übertragungen des Konfliktgraphen gefärbt. Dieser Vorgang wiederholt sich für das gesamte Netzwerk. Auf diese Weise erhält man in einem Schritt synchronisierte Schedules. Der Algorithmus ist in [Now06, S. 118] formal beschrieben.

Nowak [Now06, S. 119] betont, dass durch die Vorbelegung der Farben Umfärbe-Heuristiken oder Matchings nicht mehr anwendbar sind. In Bezug auf die Arbeit von Erlebach und Jansen [EJ97] ist es aber dennoch möglich, in polynomieller Zeit eine optimale globale Schedule zu erzeugen:

„Individual schedules for the calls touching every single node v can be combined into a schedule for all calls in polynomial time, such that the resulting schedule is no longer than the longest of the individual schedules.“

Abbildung 4.25 zeigt den greedy-gefärbten Kantenkonfliktgraphen von Switch 1 aus Abbildung 4.18 sowie die daraus resultierende Schedule.

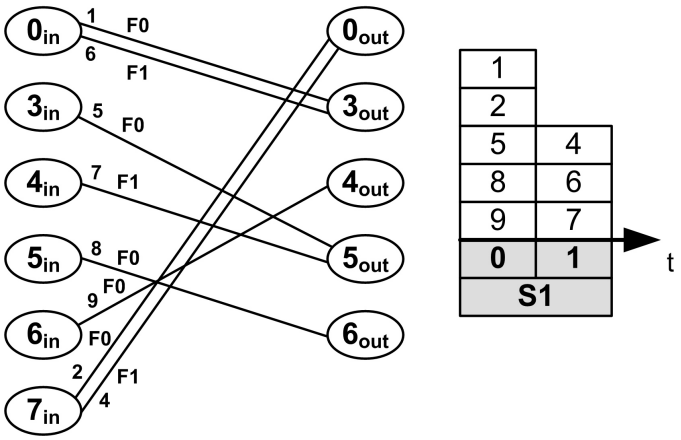


Abbildung 4.25: Gefärbter Kantenkonfliktgraph und die Schedule von Switch 1

Abbildung 4.1 zeigt, dass Switch 4 direkt mit Switch 1 verbunden ist. Die beiden Kantenkonfliktgraphen können unabhängig voneinander erstellt werden. Eine unabhängige Färbung der beiden Konfliktgraphen ist jedoch aufgrund der fehlenden Synchronisierbarkeit der erzeugten Schedules nicht möglich. Nimmt man Switch 1 als Wurzel-Switch

w an, so muss nach dem ersten vorgestellten Verfahren zunächst dessen *lokale Schedule* erzeugt werden (s. Abbildung 4.25).

Um nun eine mit Switch 1 synchrone Schedule für Switch 4 zu erhalten, müssen nach diesem Verfahren zunächst die Farben der Kommunikationslinien 1, 5 und 6 von Switch 1 übernommen werden. Im Anschluss daran werden lediglich die Kommunikationslinien gefärbt, die bislang noch keine Farbe erhalten haben. Dazu kann ein einfacher *Greedy*-Algorithmus verwendet werden. In diesem Beispiel wird lediglich die Komunikationslinie 3 gefärbt.

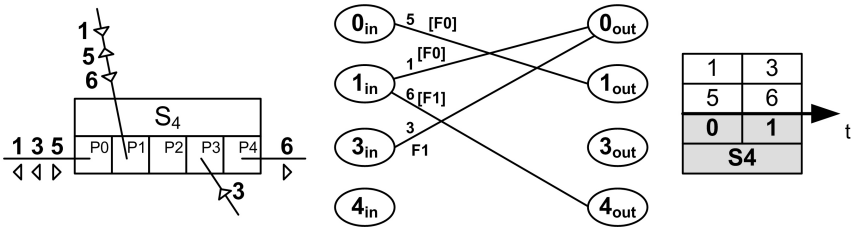


Abbildung 4.26: Switch 4, sein Kantenkonfliktgraph und die Schedule

Das zweite Verfahren zur Synchronisation verwendet modifizierte Knotenkonfliktgraphen. Dabei wird zunächst der herkömmliche Konfliktgraph eines Wurzel-Switches $w \in \mathbb{V}$ gebildet und dessen Schedule $\alpha : \Gamma_w \mapsto \underline{n}$ mit $\Gamma_w \subset \Gamma$ erzeugt. Als Wurzel-Switch kann beispielsweise der Switch mit den meisten Kommunikationslinien gewählt werden. Ziel ist es nun, synchronisierbare Knotenkonfliktgraphen der Switches zu bilden, die direkt mit w verbunden sind. Im Folgenden wird eine synchronisierbare Schedule für den Switch $v \in \mathbb{V}$ mit $\Gamma_v \subset \Gamma$ gebildet [Now06, S. 122]. Dazu wird für Γ_v im ersten Schritt die Äquivalenzrelation $\sim_\alpha$ bezüglich der bereits berechneten Schedule α zwischen zwei IRT-Übertragungen $r, s \in \Gamma$ definiert. Die IRT-Übertragungen sind dann äquivalent, wenn sie im gleichen Zeitslot der Schedule α auftreten. Somit ist $r \sim_\alpha s \Leftrightarrow (r = s) \vee (r, s \in \Gamma_w \wedge \alpha(r) = \alpha(s))$. Im Anschluss daran wird die Äquivalenzklasse zu einer Kommunikation r bezüglich der gegebenen Schedule mit $[r]_\alpha$ bezeichnet und die Menge der Äquivalenzklassen mit $\Re_\alpha = \{[r]_\alpha : r \in \Gamma_w\}$ definiert. Die Knoten des synchronisierbaren Knotenkonfliktgraphen werden nun gebildet durch die Menge der Äquivalenzklassen. Eine Kante zwischen zwei Äquivalenzklassen entsteht einerseits, falls zwei Äquivalenzklassen in Konflikt zueinander stehen.

Zur Synchronisation werden zusätzliche Kanten definiert. Sie treten auf, falls zwei IRT-Übertragungen von v auch bereits im Wurzel-Switch w vorhanden sind und dort zu verschiedenen Zeitslots zugewiesen wurden. Der synchronisierbare Knotenkonfliktgraph ist somit folgendermaßen definiert [Now06, Def. 6.3.13]:

$V := \Re_\alpha$

$E := E_{konf} \cup E_{sync}$ mit

$E_{konf} := \{\{[r], [s]\} \in V^{\underline{2}} : [r] \neq [s], r \nleftrightarrow s\}$,

$E_{sync} := \{\{[r], [s]\} \in V^{\underline{2}} : [r] \neq [s], (r, s \in \Gamma_w) \wedge (\alpha(r) \neq \alpha(s))\}$.

Zur Abschätzung der neuen Knotenmenge hat Nowak zusätzlich bewiesen [Now06, Lemma 6.3.15], dass jede Äquivalenzklasse maximal zwei Elemente enthält. Dies ist mit den zwei möglichen konfliktfreien Übertragungsrichtungen eines Kabels zu erklären.

Der entstehende ungerichtete einfache Graph ist mit beliebigen exakten Algorithmen oder Heuristiken zur Knotenfärbung färbbar. Die Konfliktgraphen von Switches, die jeweils eine Ebene tiefer im Baum der Netzwerktopologie liegen, können jedoch erst nach der Bildung der Wurzel-Schedules erstellt werden. Die Switches können also auch bei diesem Verfahren nicht unabhängig voneinander betrachtet werden. Nach der Durchführung einer Färbung wird die erstellte synchronisierbare Schedule mit der Schedule des Wurzel-Switches synchronisiert. Dieser Algorithmus ist in [Now06, S. 124] formal beschrieben.

4.2.4 Zusammenfassung

Für den Vollduplexbetrieb des Netzwerkes wurde zunächst der Konfliktbegriff neu definiert. Die daraus entstehenden Konfliktgraphen sind einfacher färbbar als im Halbduplexbetrieb. Für die Kantenfärbung wurde bewiesen, dass bipartite Konfliktgraphen entstehen, die in polynomieller Zeit exakt färbbar sind. Bei der Erstellung von Knotenkonfliktgraphen können keine Kreise mit einer ungeraden Anzahl an Knoten entstehen. Die größten Cliquen sind hier bereits im Vorfeld bekannt.

Die erstellten Schedules sind jedoch nicht mehr in polynomieller Zeit synchronisierbar, sofern sie unabhängig voneinander erstellt worden sind. Es wurden zwei Verfahren zur Lösung der Synchronisation skizziert. Beim ersten Verfahren werden existierende Färbungen in die direkt verbundenen Switches übernommen. Dahingegen werden beim zweiten Verfahren die Konfliktgraphen der direkt verbundenen Switches in einer Weise modifiziert, dass sie zur Schedule des jeweiligen Wurzel-Switches synchronisierbar sind. Dies ähnelt dem Ansatz des Path-Colorings von Erlebach [Erl98]. Während beim ersten Verfahren die Färbung mit der Synchronisation zu einem Algorithmus zusammenfällt, bleibt die Synchronisation beim zweiten Verfahren ein unabhängiger Prozess.

In der Praxis ist die Kantenfärbung der bipartiten Multigraphen vielversprechend, wobei der exakt gefärbte Switch mit der größten Anzahl an Zeitslots als Wurzel-Switch verwendet wird. Erlebach [Erl98, Kapitel 4.2] hat für ATM-Netzwerke einen übertragbaren polynomiellen Algorithmus erstellt, der auf Vollduplexnetzwerke und Unicastübertragungen angewendet werden kann. Damit kann eine Schedule mit maximaler Länge $\kappa = |\frac{5}{3} \cdot \vec{\psi}|$ erstellt werden, wobei $\vec{\psi}$ die maximale gerichtete Last im Netzwerk darstellt. Verlaufen beispielsweise an der höchst belasteten Stelle des Netzwerkes 30 Kommunikationslinien über ein Kabel, so ist die erzeugte Schedule maximal so lang, wie 50 Kommunikationslinien für ihre Übertragung im Netzwerk benötigen. Zusätzlich wird bewiesen, dass kein polynomieller Algorithmus existieren kann, der eine Schedule mit $\kappa = |\frac{4}{3} \cdot \vec{\psi}|$ erzeugt [Erl98, S. 53f.].

4.3 Multicast- und Broadcastübertragungen

Im nächsten Schritt werden neben den Punkt-zu-Punkt Unicastübertragungen auch *Multi-* und *Broadcastübertragungen* in die Modellierung einbezogen. IRT-Multicastübertragungen treten in der Automatisierungstechnik beispielsweise dann auf, wenn eine Master-Achse ihre Position mit ihren Slave-Achsen synchronisiert. Eine IRT-Broadcastübertragung kann unter anderem für die Zeitsynchronisation verwendet werden.

4.3.1 Modellierung von Kommunikationsbäumen

Multicastübertragungen resultieren nicht in Kommunikationslinien, sondern in Kommunikationsbäumen. Ein *Kommunikationsbaum* wird als spezielle Kommunkationsmenge modelliert, wobei eine Kommunikationsmenge aus überlappenden Kommunikationslinien besteht, die nicht in Konflikt zueinander stehen. Ebenso wie eine Kommunikationslinie steht eine Kommunikationsmenge K aus einer Anzahl von Knoten/Switches V_K und einer Anzahl von Kanten/Kabel E_K mit

$$V_K = \bigcup_{K' \in K} V_{K'}, V_K = \bigcup_{K' \in K} E_{K'}$$

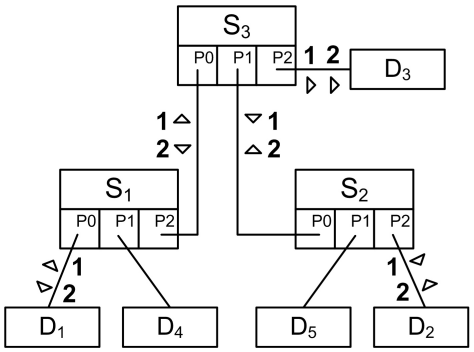
.

Eine Multicastübertragung ist ein Kommunikationsbaum [Now06, Satz 4.2.10] von genau einem sendenden Gerät $v \in \mathbb{D}$ zu einer Menge von empfangenen Geräten $W \in \mathbb{D}$. Ein Kommunikationsbaum wird definiert als die Menge der Kommunikationslinien vom Sender zu den Empfängern, also $KB(v, W) := \{KL(v, w) : w \in W\}$.

4.3.2 Definition eines Konfliktes und dessen Eigenschaften

Ein Kommunikationsbaum kann nicht zu sich selbst in Konflikt stehen, jedoch mit anderen Kommunikationsbäumen. Daher wird der Konflikt zwischen zwei Kommunikationsmengen K und L beschrieben mit $K \leftrightsquigarrow L \Leftrightarrow \exists K' \in K$ und $L' \in L$ mit $K' \leftrightsquigarrow_e L'$. Existieren also zwei Kommunikationslinien in beiden Mengen, die miteinander in Konflikt stehen, so stehen auch die beiden Kommunikationsmengen in Konflikt zueinander. Der Konflikt von Kommunikationslinien ist identisch mit der Definition von Konflikten von Kommunikationslinien in Halbduplex- bzw. Vollduplexnetzwerken.

In Vollduplexnetzwerken kann anhand eines einfachen Beispiels gezeigt werden, dass eine lokale Konfliktfreiheit zwischen zwei Kommunikationsbäumen keine globale Konfliktfreiheit impliziert. Der Kommunikationsbaum 1 wird vom Gerät 1 zu den Geräten 2 und 3 gesendet, der Kommunikationsbaum 2 vom Gerät 2 zu den Geräten 1 und 3.



Γ-ID	Quelle	Ziel
1	1	2,3
2	2	1,3

Abbildung 4.27: Zwei Multicastübertragungen

Es fällt auf, dass die beiden Bäume in den Switches 1 und 2 nicht in Konflikt zueinander stehen. Bei einer Färbung der lokalen Konfliktgraphen könnten also beide Bäume in dem selben Zeitslot ausgeführt werden. Im Switch 3 ist dies jedoch nicht erlaubt, da die beiden Kommunikationsbäume dort an Port 2 in Konflikt stehen. Dies hat zur Folge, dass die

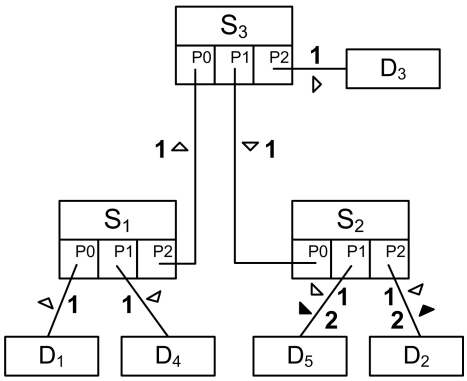
Switches im Vollduplexbetrieb bei der Übertragung von Multicastübertragungen nicht unabhängig voneinander betrachtet werden können und damit stets das gesamte Netzwerk zu betrachten ist [Now06, Bemerkung 4.2.14].

Für eine Halbduplexmulticastübertragung gilt hingegen wieder das Prinzip der Implikation von globaler Konfliktfreiheit bei lokaler Konfliktfreiheit innerhalb eines Switches [Now06, Satz 4.2.15]. Im Beispiel der Abbildung 4.27 stehen die Kommunikationsbäume im Halbduplexbetrieb bereits in Switch 1 und 2 jeweils an den Ports 0 und 2 in Konflikt zueinander.

Broadcastübertragungen können als spezielle Multicastübertragungen angesehen werden, bei denen die Menge der empfangenden Geräte gleich der Gesamtmenge der Geräte im Netzwerk mit Ausnahme des sendenden Gerätes ist. Der resultierende Kommunikationsbaum ist dann $KB(d, \mathbb{D} \setminus \{d\})$.

Es ist leicht ersichtlich, dass jede andere Kommunikation - sei es Unicast, Multicast oder Broadcast - mit einer laufenden Broadcastübertragung im Halbduplexnetzwerk in Konflikt steht, da die Broadcastübertragung über alle Kabel im Netzwerk verläuft.

Dies gilt gewöhnlicherweise auch in einem Vollduplexnetzwerk. Das folgende Beispiel zeigt eine Broadcastübertragung mit der $ID = 1$ sowie eine Unicastübertragung mit der $ID = 2$. Unabhängig von der Netzwerkinfrastruktur stehen diese beiden Übertragungen in Konflikt zueinander. Lediglich ein einziges Gerät könnte unabhängig von der Broadcastübertragung gleichzeitig eine Unicast-Übertragung zu dem Sender der Broadcastübertragung durchführen. So kann in diesem Beispiel das Gerät 3 eine Übertragung zu Gerät 1 vornehmen.



Γ-ID	Quelle(r)	Ziel(r)
1	1	2,3,4,5
2	2	5

Abbildung 4.28: Broadcast- und Unicastübertragung

Da sich eine Multicastübertragung aus Kommunikationslinien zusammensetzt, gilt auch im Vollduplexbetrieb, dass die Multicastübertragung mit der Broadcastübertragung in Konflikt steht. Dies bedeutet, dass eine Broadcastübertragung stets einen eigenen Zeitslot benötigt.

Bei echtzeitfähigen Ethernet-Ansätzen wie EPL, die ausschließlich auf Broadcastübertragungen basieren, ist die Erstellung der Schedules trivial, da jede Kommunikation einen eigenen Zeitslot benötigt. Dadurch entsteht natürlich eine längere Schedule und damit zunächst eine höhere Zykluszeit. Dies wird jedoch durch die Verwendung von Hubs mit einer geringeren Verzögerungszeit pro Verteiler im Gegensatz zu Switches teilweise kompensiert.

4.3.3 Konfliktgraphen, Färbung und Schedules

Werden Multicastübertragungen in einem Vollduplexnetzwerk unterstützt und sollen die Konfliktgraphen für jeden Switch dennoch unabhängig berechnet werden, so kann das gesamte Echtzeit-Netzwerk in der Modellierung auf Halbduplexbetrieb umgeschaltet werden. Die erzeugten Echtzeit-Schedules werden dadurch in der Regel länger, da ein Teil der Bandbreite für den Echtzeitbetrieb ungenutzt bleibt. Diese Bandbreite kann jedoch nach der Berechnung der Schedules für asynchronen Datenverkehr verwendet werden.

Sind die Anforderungen der Hardware an die Echtzeitfähigkeit jedoch sehr hoch, so sollte der Fokus auf eine möglichst optimale Echtzeit-Schedule gelegt werden. Dazu kann ein *globaler Knotenkonfliktgraph* gebildet werden, bei dem jeder Knoten einer Kommunikation[1] entspricht. Eine Kante wird dann eingezeichnet, sobald zwei IRT-Übertragungen in Konflikt zueinander stehen. Damit ist der Knotenkonfliktgraph $C_{Knoten}(\Gamma) = (V, E)$ definiert durch

$$\begin{aligned} V &:= \Gamma \\ E &:= \{\{r, s\} \in V^{\underline{2}} : r \leftrightsquigarrow_v s\} \end{aligned} .$$

Eine mögliche Vorgehensweise zur Erstellung dieses Konfliktgraphen besteht darin, zunächst alle IRT-Übertragungen als Knoten zu modellieren. Im Anschluss daran werden die Switches nacheinander durchlaufen, die Konflikte innerhalb jedes Switches ermittelt und diese dann als Kanten im Konfliktgraphen eingetragen. Nowak zeigt, dass dieser Konfliktgraph [Now06, Algorithmus 21] in polynomieller Zeit erstellt werden kann. Der resultierende Graph ist einfach und schlingenfrei, kann jedoch beliebige Kreise enthalten. Eine exakte Färbung ist daher NP-vollständig. Große bekannte Cliquen sind wiederum die IRT-Übertragungen, welche über ein gemeinsames Kabel in der gleichen Richtung verlaufen. Ein solcher Graph kann beispielsweise mit einer in der Praxis guten Färbeheuristik wie DSATUR gefärbt werden. Der globale Knotenkonfliktgraph zu Abbildung 4.27 mit einer Greedy-Färbung in der Reihenfolge der Kommunikations-IDs ist in Abbildung 4.29 dargestellt.

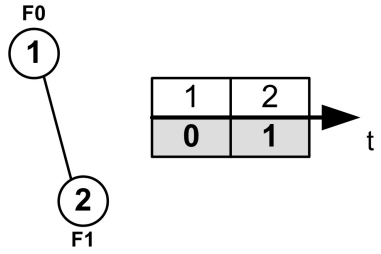


Abbildung 4.29: Gefärbter globaler Knotenkonfliktgraph und globale Schedule

Aus dem gefärbten Graphen lässt sich nun wie gehabt eine *globale Schedule* ableiten. Die lokalen Schedules für die einzelnen Switches lassen sich erstellen, indem alle Übertragungen, welche den jeweiligen Switch nicht betreffen, aus der Schedule entfernt werden. Dabei können leere Zeitslots entstehen, die jedoch nicht gestrichen werden dürfen, da ansonsten die Synchronität der zyklisch ablaufenden Schedules aufgehoben würde. Ein solcher globaler Knotenkonfliktgraph kann natürlich auch direkt ausschließlich für IRT-

[1] also einer Kommunikationslinie bzw. einem Kommunikationsbaum

Unicastübertragungen erstellt werden, falls die Switches nicht einzeln betrachtet werden sollen. Auf diese Weise kann auch das in Kapitel 4.2.3 vorgestellte Verfahren zur Äquivalenzklassenbildung mit dem Ziel der Synchronisierung von Vollduplexübertragungen umgangen werden. Dabei verzichtet man zwar auf die parallele Auswertung der unabhängigen Teilbäume, reduziert jedoch gleichzeitig den algorithmischen Overhead zur Synchronisierung der Schedules.

Ein äquivalenter globaler Kantenkonfliktgraph für Vollduplex-Multicast-Scheduling, bei dem die Switches die Knoten und die Übertragungen die Kanten bilden, kann aufgrund des Modellierungsansatzes nur schwer gebildet werden. Die Ursache dafür liegt darin, dass eine Kommunikation in der Regel über mehrere Knoten verläuft. Jede Kommunikation müsste dann als Hyperkante im Konfliktgraphen eingetragen werden, die bei der Färbung eine einzige Farbe erhält. Zu diesem Zweck müsste ein Kantenfärbungsalgorithmus verwendet werden, der Hyperkanten einbeziehen kann. Aus dem gefärbten Kantenkonfliktgraphen lässt sich, wie bereits beim Knotenkonfliktgraphen beschrieben, eine globale Schedule erstellen, die auf lokale Schedules für die einzelnen Switches umgerechnet werden kann.

Da bei einer Multicast-Halbduplexübertragung die Switches unabhängig voneinander betrachtet werden können, ist die Vorgehensweise bei der Erstellung der Konfliktgraphen [Now06, Algorithmus 17 und 19], deren Färbung, der Errechnung der synchronisierbaren Schedules sowie deren Synchronisierung [Now06, Algorithmus 28] identisch mit der Vorgehensweise von Unicastübertragungen im Halbduplexmodus.

Wie bereits im vorherigen Kapitel beschrieben wurde, benötigt jede Broadcastübertragung einen eigenen Zeitslot in der Schedule. Dort wurde auch bereits die triviale Erzeugung einer globalen Schedule skizziert für den Fall, dass ein Netzwerk wie *EPL* ausschließlich Broadcastübertragungen verwendet. Treten nun IRT-Broadcastübertragungen zusätzlich zu Unicast- oder Multicastübertragungen in einem Anwendungsfall auf, so können diese bei der Berechnung der Schedules zunächst vernachlässigt werden. Die entsprechenden Kommunikationsbäume können aus der Kalkulation entfernt werden, so dass die Algorithmen wie bereits beschrieben ausgeführt werden können bis hin zur Erzeugung der lokalen Schedules. In einem letzten Schritt kann nun die Anzahl von Zeitslots an die lokalen synchronen Schedules hinzugefügt werden, welche der Anzahl der Broadcastübertragungen entspricht.

Abbildung 4.30 zeigt die lokalen Schedules der Switches 1 und 2 aus Abbildung 4.27 als Resultat der Berechnung der globalen Schedule, die in Abbildung 4.29 dargestellt ist. Da alle Übertragungen über diese beiden Switches verlaufen, entsprechen die lokalen Schedules in diesen Fällen der globalen Schedule. Die lokalen Schedules werden nun um drei Broadcastübertragungen erweitert.

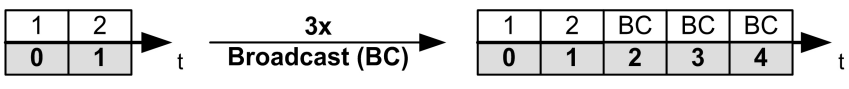


Abbildung 4.30: Hinzufügen von Broadcast-Zeitslots

4.3.4 Zusammenfassung

Durch das Hinzufügen von Multicastübertragungen in Vollduplexnetzwerke können Switches nicht mehr prinzipiell unabhängig voneinander betrachtet werden. Um dennoch synchrone Schedules zu erhalten, wurde ein globaler Knotenkonfliktgraph erstellt, dessen Färbung zu einer Schedule für das gesamte Netzwerk führt. Daraus lassen sich die lokalen Schedules ableiten. Die Abbildung 4.31 stellt die Vorgehensweise nochmals dar.

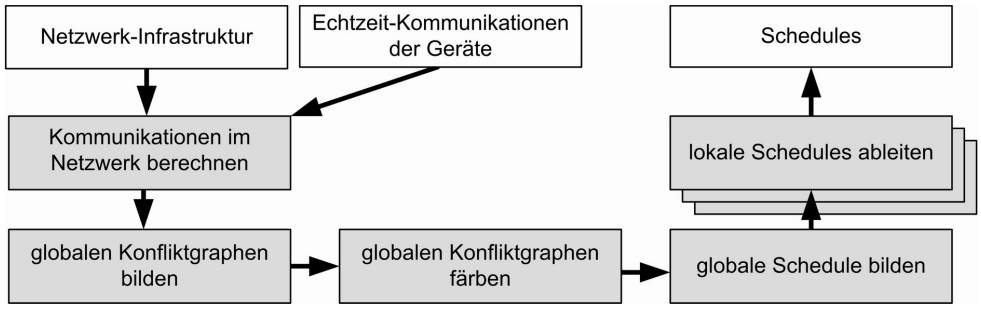


Abbildung 4.31: Vorgehensweise bei Multicastübertragungen in Vollduplexnetzwerken

Multicastübertragungen in Halbduplexnetzwerken können nach dem gleichen Schema wie Unicastübertragungen in Halbduplexnetzwerken berechnet werden, s. Abbildung 4.16.

Sowohl bei Halbduplex-, als auch bei Vollduplexüertragungen sind Broadcastsendungen leicht zu einer gegebenen Schedule hinzuzufügen. Geht man von einem üblichen Vollduplexnetzwerk aus, in welchem sowohl Unicast-, als auch Broadcastübertragungen ausgeführt werden, so ergibt sich die Vorgehensweise nach Abbildung 4.32.

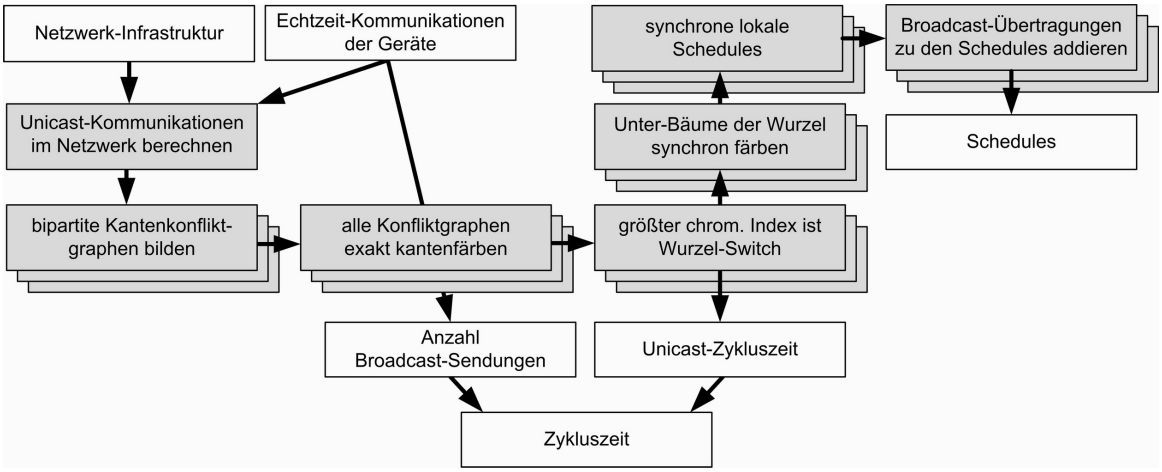


Abbildung 4.32: Vorgehensweise bei Unicast- und Broadcastübertragungen

Wie bereits beschrieben wurde, resultieren aus Unicastübertragungen in Vollduplexnetzwerken bipartite Multigraphen für jeden Switch, die in polynomieller Zeit exakt kantenfärbbar sind. Aus der größten lokalen Schedule lässt sich zusammen mit der Anzahl der benötigten Broadcastübertragungen mit hoher Wahrscheinlichkeit (vgl. [Erl98, Kapitel 4.2]) die Gesamtzykluszeit exakt ermitteln. Ist diese Gesamtzykluszeit ausreichend, so kann man mit der synchronen Färbung der Teilbäume der Netzwerkinfrastruktur beginnen, die

sich unterhalb des Switches mit der größten lokalen Schedule befinden. Daraus resultieren synchrone *lokale Schedules*, zu denen man die Broadcastübertragungen addieren kann.

4.4 Integration von Hubs

Im vorherigen Kapitel wird bereits beschrieben, dass die Erzeugung von Schedules in Netzwerken mit ausschließlicher Verwendung von *Hubs*, die aufgrund ihrer Technologie lediglich Broadcastsendungen zulassen, sehr leicht realisierbar ist. Zusätzlich dazu wurde beschrieben, dass Hubs stets im Halbduplexmodus arbeiten. Wird jedoch ein Hub an einem Switch angeschlossen, so wird dieser Port des Switches unter Verwendung der Autonegotiation ebenfalls in den Halbduplexmodus geschaltet. Die restlichen Ports des Switches werden, sofern an diese Ports andere Switches oder vollduplexfähige Geräte angeschlossen sind, weiterhin im Vollduplexmodus betrieben.

In heutigen $100Mbit/s$-Netzwerken ist die Kombination von Hubs und Switches aufgrund von preisgünstigen COTS-Switches nicht weit verbreitet. Dazu kommt, dass bei dem Einsatz von Switches durch ihre Vollduplexfähigkeit die zur Verfügung stehende Bandbreite verdoppelt wird. Daher stellt sich die Frage, warum diese Kombination im Umfeld der Automatisierungstechnik betrachtet werden soll. Der Grund dafür liegt in der wesentlich geringeren Verzögerungszeit von Hubs. Da die Linientopologie gerade in der Automatisierungstechnik zur Minimierung von Verkabelungsaufwand beliebt ist, werden die Verteiler oft kaskadiert. Innerhalb einer solchen Linie werden meist parallel angeschlossene Antriebe mit einer Master-Achse zu einer „*Software-Welle*“ zusammengeschaltet (vgl. Kapitel 2.1.2.4). Diese Übertragungen besitzen harte Echtzeitanforderungen bezüglich ihrer Verzögerungszeit. Zusätzlich sollte der Empfang des ersten Slaves zeitlich nicht zu weit von dem Empfang des letzten Slaves auseinander liegen. Eine solche „Software-Welle“ ist meist über genau einen Punkt mit dem restlichen Netzwerk der automatisierten Anlage verbunden, was Abbildung 4.33 darstellt.

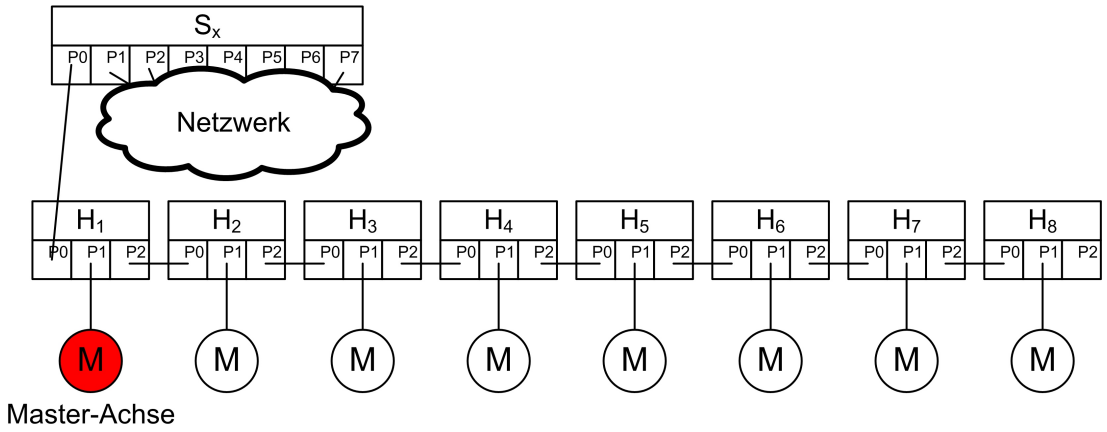


Abbildung 4.33: Master-Achse mit ihren Slaves an Hubs angeschlossen

Bei einer emulierten Linientopologie unter Verwendung von 3-Port Switches muss man zusätzlich bedenken, dass die Wahrscheinlichkeit von Konflikten zunimmt. Es können also nicht mehrere IRT-Übertragungen in einem Netzwerk-Segment, wie in Abbildung 4.33 dargestellt, durchgeführt werden.

Ein typischer Anwendungsfall liegt darin, dass die Master-Achse zyklisch ihre Istposition an die Slave-Achsen sendet. Diese antworten dann dem Master mit ihren zu sendenden Daten, welche auch von den anderen Slaves empfangen werden können. Dies wird in der Automatisierungstechnik als Querverkehr bezeichnet (s. Kapitel 2.1.1.1). Es ist unüblich, dass einzele Slave-Achsen im Rahmen der Echtzeit-Kommunikation direkt von außen angesprochen werden oder Daten direkt nach außen versenden. Ebenso werden Hubs im Umfeld der Automatisierungstechnik in der Regel nicht als Teil der Baumtopologie von Switches umgeben, da dies durch die Halbduplexübertragung auf allen Ports die parallel mögliche Datenübertragung einschränkt. Dies bedeutet, dass Hubs üblicherweise an Randbereichen des Netzwerkes eingesetzt werden, bei denen die Verzögerungszeit zu minimieren ist. Dabei kommuniziert die Master-Achse zumeist öfter mit dem Netzwerk außerhalb der „Software-Welle“ als die Slave-Achsen. Die Master-Achse empfängt dabei zumeist neue Sollwerte oder übermittelt ihre Istposition an eine Steuerung.

4.4.1 Definition eines Konfliktes und dessen Eigenschaften

Betrachtet man ein Netzwerk aus Hubs, so können die Übertragungen äquivalent zu Broadcastübertragungen unter Verwendung von Halbduplex-Switches angesehen werden. Zu einem Zeitpunkt existiert maximal eine Kommunikation im Netzwerk, die von allen Geräten empfangen werden kann.

Ein Problem ergibt sich, sobald ein Hub zusammen mit Switches in einem Netzwerk kombiniert wird. Abbildung 4.34 zeigt zwei Kommunikationslinien, die vom Gerät 2 zum Gerät 1 und vom Gerät 1 zum Gerät 3 verlaufen. Zusätzlich dazu ist eine Kommunikation mit der $ID = 3$ zwischen Gerät 1 und Gerät 4 vorgesehen, die jedoch auch von allen anderen am Hub angeschlossenen Geräten empfangen werden kann. Eine vierte und fünfte Übertragung ist unabhängig von dem Hub. Diese beiden IRT-Übertragungen verlaufen zwischen den Geräten 2 und 3 und betreffen dadurch aussschließlich den Switch 2.

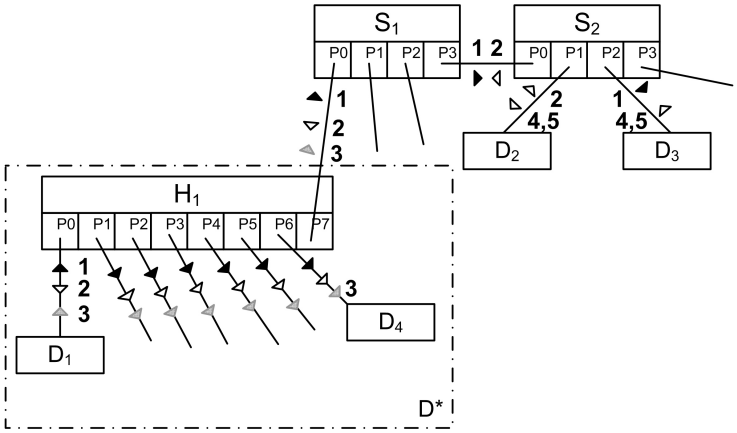


Abbildung 4.34: Übertragungen mit einem Hub und Switches

Während die beiden Kommunikationslinien 1 und 2 aufgrund des Halbduplexbetriebes des Hubs sowie an Port 0 von Switch 1 in *Konflikt* stehen, sind sie in Switch 2, der ausschließlich im Vollduplexbetrieb arbeitet, konfliktfrei. Wie beim Multicast-Betrieb impliziert also auch hier die lokale Konfliktfreiheit keine globale Konfliktfreiheit. Die Schedules

der beiden Switches im Beispiel können im Vollduplexmodus nicht unabhängig voneinander berechnet werden.

Des Weiteren ist zu beachten, dass die dritte Kommunikationslinie auch vom Port 0 des Switches 1 empfangen wird. Trotz der Übertragung innerhalb des Hubs ist der gesamte Port 0 dieses Switches für die Dauer der internen Übertragung gesperrt. Da der Hub ausschließlich Broadcastsendungen zulässt, kann er als einziges Gerät $D*$ im Halbduplexbetrieb angesehen werden. Alle Übertragungen stehen dort in Konflikt zueinander und benötigen einen eigenen Zeitslot.

4.4.2 Konfliktgraphen, Färbung und Schedules

Um dennoch eine unabhängige Berechnung durchführen zu können, kann das gesamte Netzwerk - wie bereits im Multicast-Vollduplexbetrieb skizziert - im Halbduplexbetrieb kalkuliert werden. Dadurch stehen die IRT-Übertragungen 1 und 2 während ihres gesamten gemeinsamen Verlaufes in Konflikt miteinander. Überwiegt die Anzahl der Hubs in der konkreten Anlagenverkabelung bzw. werden die IRT-Frames zum Großteil über die Hubs abgewickelt, so ist eine Halbduplex-Modellierung des Gesamtnetzes durchaus sinnvoll. Der reale Vollduplexmodus der Switches stellt in diesem Fall die Reserve für asynchrone Übertragungen dar. Auch hier werden die Echtzeit-Schedules länger als im optimalen Falle. Die Ursache dafür liegt darin, dass auch IRT-Übertragungen, welche lediglich über Switches verlaufen, im Halbduplexmodus betrachtet werden. Dadurch werden ggf. parallel ausführbare IRT-Übertragungen hintereinander geschaltet.

Die IRT-Übertragungen, welche über die Hubs verlaufen, können dabei als Multicastübertragungen im Gesamtnetzwerk betrachtet werden. Dies ist möglich, da alle Geräte, die an die Hubs angeschlossen sind, die Frames empfangen können und während einer aktiven Übertragung nicht selbst senden dürfen. Gleichzeitig ist anzumerken, dass die modellierte Multicast-Kommunikation mit $ID = 3$ an dem Port 0 von Switch 1 endet, also nicht bis zu einem Gerät als Senke verläuft. Alternativ dazu kann ein Hub inklusive aller angeschlossenen Geräte als ein einzelnes Gerät betrachtet werden, für das keine Schedule zu erstellen ist. Durch diese Betrachtung entstehen die folgenden Konfliktgraphen für die beiden Switches.

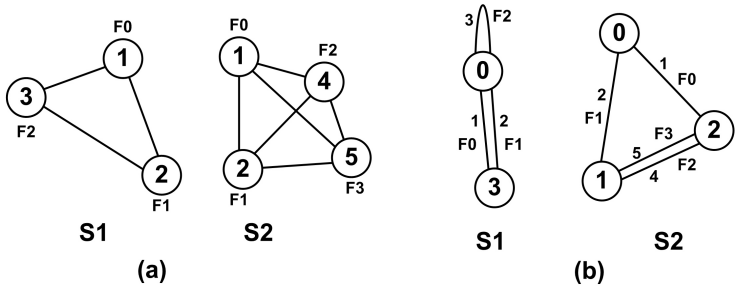


Abbildung 4.35: Knotenkonfliktgraphen (a) und Kantenkonfliktgraphen (b) des Beispiels

Diese Graphen sind synchronisierbar im Sinne der Definition der Synchronisierbarkeit eines Halbduplexnetzwerkes und können unabhängig voneinander gefärbt werden. Im Beispiel wurde wieder eine Greedy-Färbung durchgeführt. Der Kantenkonfliktgraph ist ein ungerichteter Multigraph, der aufgrund des Hubs auch Schlingen enthalten kann, im

Beispiel an Port 0 von Switch 1. Im Knotenkonfliktgraphen existiert die dritte Kommunikation im Switch 1 als Knoten neben den anderen Übertragungen, so dass wiederum ein schlingenfreier Graph resultiert.

Aus diesen Färbungen lassen sich die Schedules für jeden Switch erzeugen. Um freie Kapazitäten in jedem Zeitslot zu ermitteln, lassen sich nach Erstellung der Schedule die Portbelegungen in jedem Zeitslot in polynomieller Zeit berechnen, indem die belegten Ports sowie deren Richtung in einer Tabelle abgelegt werden.

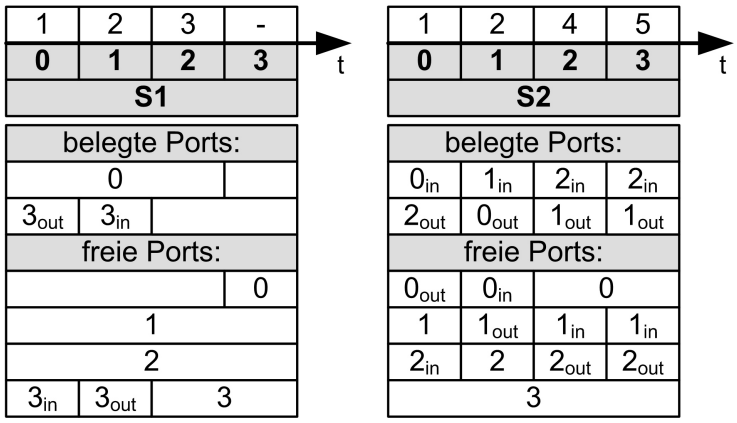


Abbildung 4.36: Synchrone Schedules (Halbduplex)

Werden Ports an ihren Eingängen bzw. Ausgängen nicht für IRT-Übertragungen verwendet, so können diese für asynchrone Übertragungen verwendet werden. Abbildung 4.36 zeigt dabei die freien Kapazitäten der Switches. Die Ports 1 und 2 von Switch 1 sowie der Port 3 von Switch 2 werden in keinem Slot für IRT-Frames verwendet. Deren Bandbreite steht demnach vollständig für asynchrone Kommunikation zur Verfügung. Da aufgrund der Vollduplexfähigkeit die Eingangs- und Ausgangsteile der betreffenden Ports unabhängig voneinander gesteuert werden können, bleibt auch an Ports mit harten Echtzeitanforderungen eine gewisse Bandbreite ungenutzt. Lediglich der Port 0 von Switch 1 ist aufgrund seiner Halbduplexbetriebsart und seiner hohen Belastung durch die Übertragungen 1 bis 3 in jedem Zeitslot belegt. Nur durch das Hinzufügen des Slots 3 können die Geräte, die am Hub angeschlossen sind, asynchron kommunizieren. Dieser Slot ist aufgrund der Synchronisation mit der längeren Schedule von Switch 2 notwendig. Die Einbindung von asynchron gesendeten Frames wird in Kapitel 4.5 genauer betrachtet.

Als Alternative zu dieser Berechnung kann wiederum ein *globaler Knotenkonfliktgraph* gebildet werden, der im Regelfall zu einer besseren Echtzeit-Schedule führt und in Abbildung 4.37 dargestellt ist. So verlaufen die Kommunikationslinien 4 und 5 des Beispiels ausschließlich über Switches und damit im Vollduplexmodus. Sie stehen mit keiner anderen Kommunikationslinie in Switch 2 in Konflikt. Da die Übertragungen 1 und 2 im Halbduplexteil des Netzwerkes in Konflikt zueinander stehen, im Vollduplexteil zwischen Switch 1 und 2 jedoch nicht, muss das Netzwerk als Gesamtheit betrachtet werden.

Aus diesem Graphen lassen sich wiederum die lokalen Schedules der Switches ableiten, indem lediglich die IRT-Übertragungen betrachtet werden, die über den betreffenden Switch verlaufen.

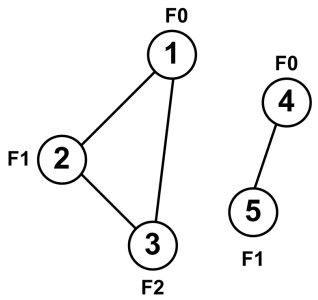


Abbildung 4.37: Globaler Konfliktgraph (Vollduplex)

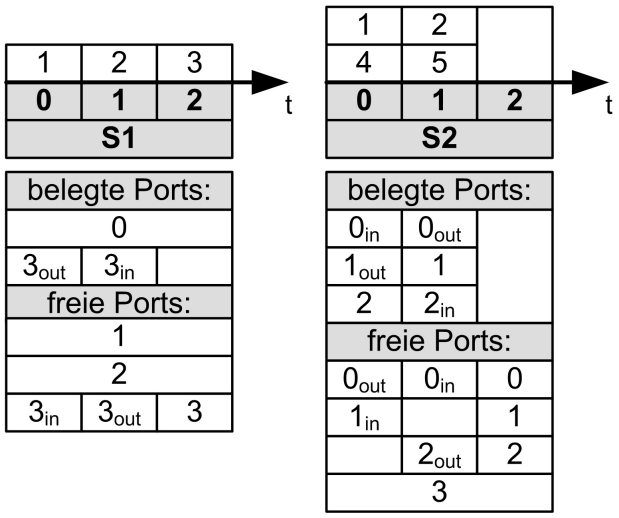


Abbildung 4.38: Synchrone Schedules (Vollduplex)

Durch den Vollduplexbetrieb reduziert sich die Schedule des Switch 2 so stark, dass in der globalen Schedule ein Zeitslot weniger benötigt wird. Zusätzlich wird im letzten Zeitslot des Switch 2 keine Kommunikation mit Echtzeitanforderungen ausgeführt. Über den Port 0 von Switch 1 und zu Switch 2 kann jedoch innerhalb der Schedule keine asynchrone Kommunikation verlaufen.

Eine weitere neue Eigenschaft kann bei Betrachtung der lokalen Kantenkonfliktgraphen im Vollduplexmodus festgestellt werden. Aufgrund der Hubs ist dieser Kantenkonfliktgraph nicht mehr bipartit und kann Kreise ungerader Länge enthalten. Dazu wird in Abbildung 4.39 ein beispielhaftes Netzwerk mit dessen Kantenkonfliktgraphen für Switch 1 erstellt. Der Hub ist dabei an Port 7 des Switches angeschlossen, welcher in den Halbduplexbetrieb umgeschaltet wird. Zusätzlich dazu können, wie bereits in der Halbduplexbetrachtung, Schlingen auftreten.

Lokale Kantenkonfliktgraphen sind also bei der Einbeziehung von Hubs in das Netzwerk aufgrund ihrer schwierigen Färbbarkeit sowie aufgrund der fehlenden Synchronisierbarkeit zu vermeiden. Bei lokalen Knotenkonfliktgraphen bleibt das Problem der Synchronisierbarkeit bestehen; eine leichtere Färbbarkeit existiert auch in diesem Falle nicht.

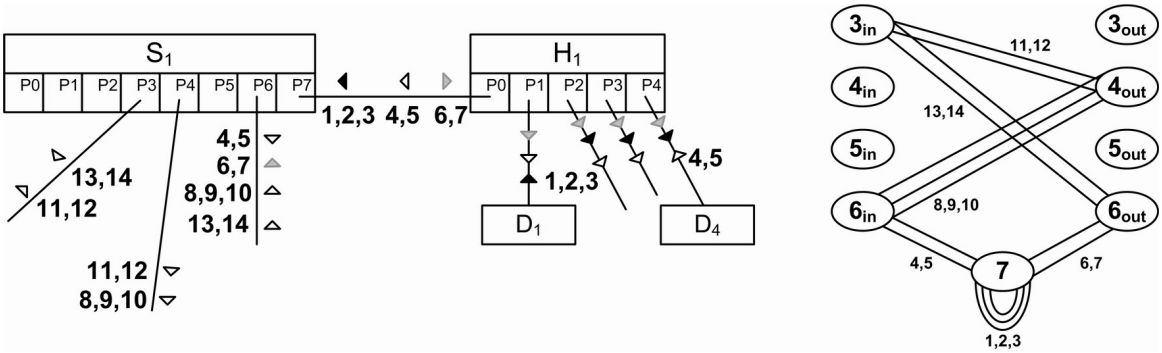


Abbildung 4.39: Netzwerk mit Schlinge und Kreis im Konfliktgraph

4.4.3 Zusammenfassung

Zusammenfassend ist zu sagen, dass Hubs in die Modellierung der Schedules an den Enden eines auf Switches basierenden Netzwerkes eingebunden werden können. Aufgrund der geringeren Verzögerungszeit der Hubs können direkt angeschlossene Geräte schneller kommunizieren. Zusätzlich ist ein Querverkehr einer Master-Achse zu ihren Slaves sowie der Slaves untereinander möglich. Ähnlich der Multicastübertragungen im Vollduplexmodus können die Schedules nicht unabhängig voneinander betrachtet werden. Dazu wurden zwei Lösungsansätze aufgezeigt.

Der erste Ansatz besteht in der Modellierung eines vollständigen Halbduplexnetzwerkes. Damit ist eine unabhängige Betrachtung der Switches möglich. Gleichzeitig werden jedoch die Schedules mit IRT-Übertragungen, die über Switches verlaufen, sub-optimal erstellt. Andererseits bleibt ein größerer Teil der Bandbreite des Netzwerks für asynchrone Kommunikation erhalten.

Der zweite Ansatz führt über einen globalen Knotenkonfliktgraphen. Dessen Berechnung führt zu einer optimierten Echtzeit-Schedule und sollte vor allem dann verwendet werden, wenn die Zykluszeit der Anlage minimiert werden soll oder wenn ein Großteil der Echtzeit-Kommunikation über Switches verläuft, da hier deren Vollduplexfähigkeit ausgenutzt werden kann. Die Vorgehensweise ist identisch zu Abbildung 4.31.

4.5 Integration von asynchronen Übertragungen

In diesem Szenario werden Möglichkeiten diskutiert, auf welche Weise asynchroner Datenverkehr in die Schedule einbezogen werden kann. Während in diesem Kapitel die formalen Optionen betrachtet werden, liegt in Kapitel 5 der Fokus auf mögliche technische Realisierungen.

Wie bereits bei ProfiNet IRT realisiert wurde, darf der asynchrone Datenverkehr keinen Einfluß auf die harte Echtzeitfähigkeit des Netzwerkes besitzen. Ein asynchron gesendeter Frame darf also einen Echtzeit-Frame, der in seinem Zeitslot zyklisch versendet wird, in keiner Weise verzögern. Die Echtzeit-Schedule dominiert also die Übertragungen insbesondere dann, wenn harte isochrone Echtzeit-Anforderungen der automatisierten Anlage existieren. Es müssen demnach Zeitslots für asynchrone Übertragungen verwendet wer-

den, welche keinen Einfluss auf die Echtzeit-Schedule haben. Auf diese Weise entsteht das deterministische TDMA-Zugriffsverfahren.

Da die Echtzeit-Schedules zyklisch ausgeführt wird, kann es vorkommen, dass innerhalb der Schedule einige Ports von Switches vollständig mit Echtzeit-Sendungen ausgelastet sind. Dies trifft beispielsweise auf Port 0 von Switch 1 in Abbildung 4.38 zu. Eine Möglichkeit zur Einbindung von asynchronen Übertragungen besteht darin, die berechnete Zykluszeit der Offline-Schedule mit der zulässigen Obergrenze der Zykluszeit der Produktionsanlage - die meist konstruktionsbedingt ist und aus mechanischen bzw. physikalischen Grenzen besteht - zu vergleichen. Ist die berechnete Zykluszeit geringer, so kann die verbleibende Zeit in jedem Zyklus für die Übertragung von asynchronen Frames verwendet werden. Dadurch wird ein Teil der zur Verfügung stehenden Bandbreite für asynchrone Kommunikation reserviert. Abbildung 4.40 zeigt die Schedule aus Abbildung 4.38, die um zwei weitere Zeitslots erweitert wurde. Dadurch steht jeder Port eines Switches innerhalb jedes Zyklus mindestens in zwei Zeitslots für asynchrone Übertragungen zur Verfügung.

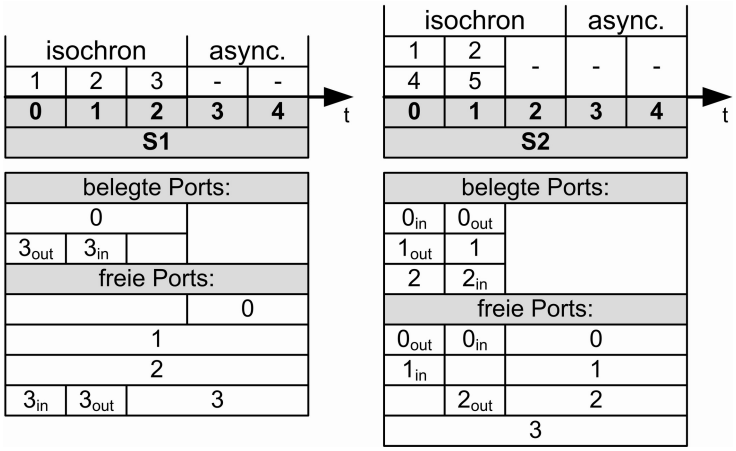


Abbildung 4.40: Hinzufügen eines Bereiches für asynchrone Übertragung

Auf diese Weise wird ein Bereich der Gesamtschedule für isochrone und ein weiterer Bereich für asynchrone Kommunikation reserviert, wie es bereits bei etablierten Lösungen wie EPL und ProfiNet der Fall ist. In diesen Lösungen wird die Trennung der beiden Phasen strikt durchgeführt. Es ist jedoch zu bedenken, dass die Größe der Echtzeit-Schedule meist aufgrund einzelner Verteiler resultiert, über welche die größte Anzahl an IRT-Übertragungen verläuft. Dadurch bleibt in der isochronen Phase ein Teil der Bandbreite bei den anderen Switches ungenutzt. Zählt man jeden Eingangs- und Ausgangsteil eines Port, so werden im Beispiel der Abbildung 4.40 in Switch 1 innerhalb der isochronen Phase 8 Portteile verwendet, während 16 ungenutzt bleiben. Gleiches gilt für Switch 2. In diesem Beispiel bleiben demnach 2/3 der Bandbreite des Netzwerkes in der isochronen Phase ungenutzt und können für andere Zwecke verwendet werden. Dabei ist zu beachten, dass jeder isochrone Zeitslot lediglich die Übertragung eines Paketes mit der Framelänge zulässt, wie sie für Echtzeitübertragungen spezifiziert wurde. In der Regel entspricht dies der minimalen Ethernet-Framelänge, die in einem $100 Mbit/s$-Ethernet ca. $7\mu s$ zur Übertragung benötigt. Diese Zeit wird auch als Basis für Kalkulationen mit ProfiNet-Switches verwendet [Pop05].

Soll hingegen ein maximal großer Ethernet-Frame unfragmentiert übertragen werden, so muss zu der Echtzeit-Schedule ein zusätzlicher Zeitslot von $123\mu s$ hinzugefügt werden. Dies entspricht 19 zusätzlichen Zeitslots für Frames minimaler Größe. Die Obergrenze der Verzögerungszeit liegt innerhalb der härtesten IAONA Echtzeitklasse bei $1ms$. Will man solch große Frames übertragen, so können maximal 129 Slots für die Echtzeit-Schedule reserviert werden, damit diese Grenze noch eingehalten wird. Dabei ist zu beachten, dass bislang keine Laufzeiten in den Switches einkalkuliert sind, also von einer idealisierten Übertragung ausgegangen wird. Eine Kalkulation unter Berücksichtigung von Laufzeiten erfolgt in Kapitel 4.8.

Um auch die freien Zeitslots innerhalb der isochronen Phase verwenden zu können, muss man den Overhead der Protokolle berücksichtigen, die oberhalb des Ethernet-Protokolls zur Anwendung kommen. Dies ist zumeist UDP/IP oder TCP/IP. Der Header eines Ethernet-Frames ist ohne VLAN-Erweiterung $26Byte$ groß, so dass bei einem minimal großen Frame von $72Byte$ ein Nutzdatenanteil von $46Byte$ verbleibt. Der Header eines IPv4-Paketes ist standardmäßig $20Byte$ groß. Daher verbleiben nach der Fragmentierung noch $26Byte$ an Nutzdaten im Ethernet-Frame, was einem Nutzdatenanteil von ca. 36% entspricht. Wird zusätzlich UDP auf der Transportschicht verwendet, so wird dem Datenstrom ein weiterer Header von $8Byte$ einmalig hinzugefügt. Ein TCP-Header benötigt statt dessen $20Byte$ ohne die Verwendung von Header-Optionen.

Bei der Übertragung eines maximal großen Ethernet-Frames verbessert sich der Nutzdatenanteil auf über 95%. Freie Bereiche solcher Größe werden in Echtzeit-Schedules jedoch nicht auftreten, so dass der Nutzdatenanteil zwischen diesen Grenzen liegen wird.

Eine Möglichkeit zur Verbesserung des Nutzdatenanteils der Ethernet-Frames besteht darin, berechnete Echtzeit-Schedules dahingehend zu optimieren, dass freie Ports nach Möglichkeit über einen kontinuierlichen Zeitraum innerhalb der isochronen Phase frei bleiben. Dadurch können dort größere asynchrone Frames übertragen werden. Dies kann durch Permutation von Zeitslots geschehen.

Neben der Modellierung ist generell die Frage zu beantworten, auf welche Weise die isochonen Übertragungen von den asynchronen Übertragungen zu trennen sind. Ziel muss es sein, dass die isochonen Übertragungen in keiner Weise von den asynchronen Übertragungen beeinflußt werden. Dies ist prinzipiell auf zwei verschiedene Weisen möglich. Die erste Möglichkeit besteht darin, dass jedes asynchron sendende Gerät Kenntnis von der Schedule besitzen und mit dem Netzwerk synchronisiert sein muss. Der Nachteil besteht darin, dass dazu eine Modifikation der asynchron sendenden Gerät durch einen Treiber oder sogar durch bestimmte Hardware[2] notwendig ist. Die zweite Möglichkeit besteht in der Modifikation der Switches dahingehend, dass sie angeschlossene asynchron sendende Geräte erkennen und deren Sendungen in die Schedule selbständig einreihen. Die erfordert zwar eine weitreichendere Modifikation der Switches, ermöglicht jedoch für die Geräte transparente Übertragungen. Aus Sicht der angeschlossenen asynchron sendenden Geräte verhält sich das Netzwerk dann wie ein herkömmliches Ethernet-Netzwerk mit verminderter Bandbreite. Eine detaillierte Diskussion der Umsetzung der Schedules wird in Kapitel 5 dieser Arbeit geführt.

[2]Dies ist beispielsweise notwendig, falls ein low-level timestamping nach IEEE 1588 zur Zeitsynchronisation erfolgen soll.

4.6 Zulassen variabler Framegrößen

Als zusätzliche Verbesserung des Nutzdatenanteils können in gewissen Grenzen variable Framegrößen für die isochrone Phase verwendet werden. Anwendungen mit harten Echtzeitanforderungen können auf diese Weise Burstübertragungen versenden, um beispielsweise von einer Master-Achse durch die Übertragung eines einzelnen, größeren Ethernet-Frames eine Vielzahl von Slave-Achsen anzusteuern, die über Hubs angeschlossen sind.

Im Folgenden wird davon ausgegangen, dass die Frames nicht wie bisher ein einheitliches Aktivitätsintervall $\hat{\alpha} = 1$ besitzen, sondern exponentiell gestaffelte Übertragungslängen. Eine Kommunikation $r \in \Gamma$ kann demnach verschiedene Übertragungslängen $\delta(r) \in \left\{2^k : k : 0, ..., N\right)$ besitzen, wobei $N \in \mathbb{N}_0$ als Konstante angesehen wird. Die Schedule für alle IRT-Übertragungen wird so entworfen, dass eine Kommunikation der Länge 2^k nur zu einer Zeit $j \cdot 2^k$ mit $j \in \mathbb{N}_0$ ausgeführt werden kann. Ein Slot der Länge n heißt *leer*, wenn er keine IRT-Übertragung der Slotlänge 2^n enthält und *rekursiv leer*, wenn er gar keine IRT-Übertragung enthält. Ein nicht leerer Slot heißt gefüllt. In der Abbildung 4.41 ist eine gestaffelte Schedule für 19 Übertragungen bis $k = 2$ skizziert. Zusätzlich ist sowohl ein leerer, als auch ein rekursiv leerer Zeitslot dargestellt.

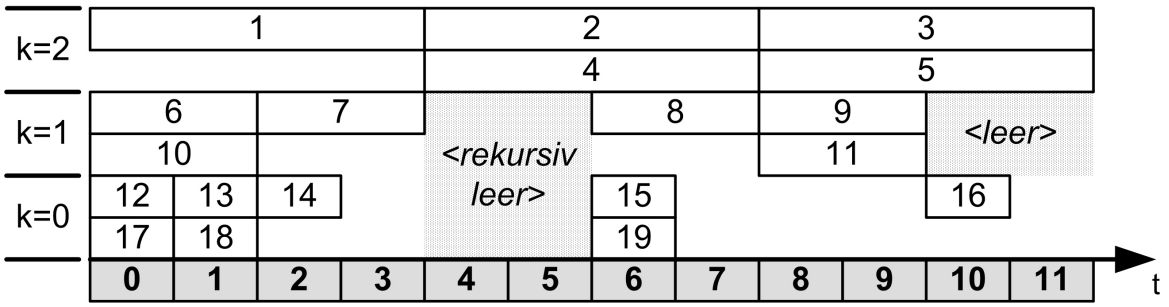


Abbildung 4.41: Gestaffelte Schedule [Now06]

Aufgrund der Staffelung wird die Berechnung der Schedule im Gegensatz zu beliebigen Framegrößen vereinfacht. Die Staffelung hat jedoch zur Folge, dass nicht in jedem Fall eine optimale Schedule gefunden wird. Die Abbildung 4.42 zeigt 4 Übertragungen, wobei die IRT-Übertragungen 1 und 2, 2 und 3 sowie 3 und 4 in Konflikt zueinander stehen. [Now06, S. 126]

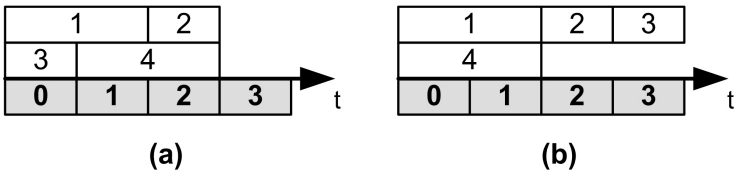


Abbildung 4.42: Optimale (a) und optimal gestaffelte Schedule (b) [Now06]

Die optimale *gestaffelte Schedule* entspricht dabei nicht der optimal möglichen Schedule und ist und ist um eine Mindestframelänge größer. Im $100Mbit/s$-Ethernet ist eine Unterteilung der Übertragungslängen in die Größen der Abbildung 4.43 sinnvoll.

Dabei entspricht $k = 0$ der minimalen Framegröße im Ethernet. In einem Netzwerk nahe der Feldebene wird die Anzahl dieser Frames überwiegen. Frames mit einer Größe

k	Byte
0	72
1	144
2	288
3	576
4	1152
5	2304

Abbildung 4.43: Gestaffelte Framegrößen

über $500Byte$ sind hingegen eher den Leitebenen zuzuordnen und deuten auf asynchrone Übertagungen. Werte größer als $k = 5$ können aufgrund der maximalen Größe eines Ethernet-Frames nicht auftreten.

Der globale Knotenkonfliktgraph beschreibt zwar alle Konflikte der IRT-Übertragungen, dessen Färbung und die daraus errechnete globale Schedule kann jedoch keine unterschiedliche Dauer für Übertragungen berücksichtigen. In Zusammenarbeit mit Nowak [Now06, S. 145ff.] wurde daher die Heuristik eines multi-dimensionalen Greedy-Algorithmus beschrieben, die in polynomieller Zeit eine gestaffelte Schedule erzeugt. Sie bezieht gestaffelte Längen von IRT-Übertragungen in die Berechnung ein und wird im Folgenden skizziert.

Der Ausgangspunkt für die Betrachtung ist ein - möglichst optimal - gefärbter globaler Knotenkonfliktgraph für die Framelänge $k = 0$, aus dem sich die globale Schedule errechnen lässt. Diese Schedule wird im Folgenden als α_0 bezeichnet. Sie kann wie in den vorherigen Kapiteln beschrieben ermittelt werden und besteht ausschließlich aus IRT-Frames, die nur eine geringe Nutzdatenmenge besitzen. Dabei entspricht jede Farbe einem Zeitslot. Die Zeitslots sind innerhalb der globalen Schedule beliebig permutierbar, da keine kausale Abhängigkeit zwischen den Übertragungen besteht. Eine solche Permutation von Zeitslots erzeugt eine neue gültige Schedule und wurde bereits zur Synchronisation lokaler Schedules eingesetzt, vgl. Kapitel 4.1.7.

Basierend auf der Schedule α_0 soll nun eine Kommunikation $r \in \Gamma$ mit doppelter Framelänge, also mit $k = 1$, hinzugefügt werden. Diese IRT-Übertragung wird mit r_1 bezeichnet. Existiert bereits ein gefüllter Slot mit doppelter Länge, in den die einzeln hinzuzufügende IRT-Übertragung r_1 konfliktfrei eingefügt werden kann, so wird die IRT-Übertragung hinzugefügt.

Existiert kein solcher Slot, so wird geprüft, ob zwei freie Slots der Länge 2^{k-1} existieren, in welche r_1 konfliktfrei eingefügt werden kann. Ist dies der Fall, so werden die untergelagerten Slots so getauscht, dass sie benachbart sind. Die Idee des Verfahrens liegt also darin, durch das Vertauschen von zwei freien Slots der Länge 2^{k-1} einen Slot der Länge k zu erzeugen, in den r_1 eingefügt werden kann.

Die Abbildung 4.44 zeigt, wie eine IRT-Übertragung mit doppelter Länge einer bestehenden Schedule durch Vertauschung untergelagerter Zeitslots hinzugefügt wird. Diese neue IRT-Übertragung 12 steht mit der bereits hinzugefügten ersten IRT-Übertragung 11 doppelter Länge sowie mit zwei IRT-Übertragungen geringerer Länge in Konflikt, nämlich mit 4 und 10. Durch die Vertauschung (a) von konfliktfreien Slots entsteht ein Bereich, der groß genug ist, um die neue IRT-Übertragung einzufügen (b).

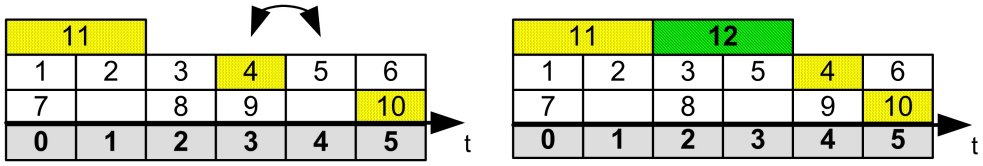


Abbildung 4.44: Tausch von Slots

Ist eine solche Vertauschung nicht möglich, so wird die IRT-Übertragung an das Ende der aktuellen Schedule eingefügt, die dadurch erweitert wird. Diese Vorgehensweise bezeichnet Nowak als *Bottom-Up Erweiterung* einer bestehenden Schedule [Now06, S. 141ff.]. Die dort beschriebene Top-Down Erweiterung wird wahrscheinlich in der Praxis nicht angewendet, da die Anzahl und die Priorität der Frames mit $k = 0$ überwiegen werden. Abschließend beschreibt Nowak einen *multidimensionalen Greedy-Algorithmus* [Now06, S. 146], mit dem ausgehend von einem beliebigen k zunächst eine Bottom-Up Erweiterung und anschließend eine Top-Down Erweiterung durchgeführt wird. Dadurch kein eine beliebige gestaffelte Framegröße als Ausgangspunkt verwendet werden.

Eine Erweiterung dieses Verfahrens besteht darin, zunächst auch eine Schedule α_1 der Übertragungen mit $k = 1$ zu erstellen. Statt einer einzelnen IRT-Übertragung r_1 werden in diesem Falle ganze Zeitslots aus α_1 nacheinander zu der bestehenden Schedule α_0 hinzugefügt.

In beiden Fällen ist es sinnvoll, neben den globalen Konfliktgraphen der einzelnen Framelängen einen weiteren globalen Konfliktgraphen zu erstellen, welcher die Konflikte aller IRT-Übertragungen beinhaltet. Anhand dieses Graphen kann dann geprüft werden, ob eine IRT-Übertragung bzw. ein weiterer Zeitslot in Konflikt mit der untergelagerten Schedule steht oder nicht.

Zur weiteren Optimierung der Heuristik skizziert Nowak [Now06, S. 148ff.] die Einführung eines Sättigungsgrades für die Vertauschung von Zeitslots in Anlehnung des DSATUR-Algorithmus zur Graphenfärbung.

Auch für den Fall, dass große Framelängen in einer Echtzeit-Schedule nicht zur Anwendung kommen, kann eine Überlegung zur Umordnung von Zeitslots sinnvoll sein. Zumeist existieren in einer industriellen Anlage nur wenige Zugangspunkte mit freien Ethernetports sowie nur wenige Anbindungen des echtzeitfähigen Netzes an das Internet bzw. das Intranet des Unternehmens. Dadurch sind die Wege von den Quellen von asynchronen Übertragungen zu deren Senken bestimmbar. Eine globale Schedule kann dahingehend optimiert werden, dass freie Portbelegungen innerhalb der isochronen Phase, die auf diesen Wegen liegen, hintereinandergeschaltet werden. Auf diese Weise erhöht sich der Nutzdatenanteil von asynchronen Frames, die innerhalb der isochronen Phase gesendet werden können, ohne die Echtzeitübertragungen zu beeinflussen.

4.7 Sendungen in Vielfachen von Produktionszyklen

Bislang wurde vereinfachend angenommen, dass jede IRT-Übertragung in jedem Zeitslot erfolgen muss. Für IRT-Übertragungen der Antriebstechnik mit höchsten Anforderungen ist dies sicherlich auch angebracht, nicht jedoch prinzipiell für jede Kommunikation der Feldebene. So ist es nicht notwendig, jeden Sensor in jedem Produktionszyklus auszulesen und

ein Datenaustausch von Steuerungen untereinander in jedem Zyklus durchzuführen. Dies würde die resultierende Echtzeit-Schedule unnötig vergrößern und die Zykluszeit drastisch erhöhen. Daher wurde bereits in etablierten Lösungen wie EPL die Multiplex-Betriebsart eingeführt, in der die Geräte in zwei Klassen unterteilt und Zeitslots von Geräten der zweiten Klasse mehrfach genutzt werden. Übertragungen in jedem Slot werden dort als *cyclic* bezeichnet, während Übertragungen, die sich Zeitslots teilen, als *prescaled* gekennzeichnet werden. In diesem Kapitel wird nun Möglichkeiten skizziert, wie man eine Multiplex-Betriebsart in das bisherige Modell integrieren kann.

Zunächst ist zu erwarten, dass die cyclic-Übertragungen mit härtesten Echtzeit-Anforderungen über wenige Verteiler verlaufen. Verlaufen sie wie in Kapitel 4.4 skizziert sogar nur über Hubs zur Minimierung der Verzögerungszeit, so kann für diese Bereiche des Netzwerks eine eigene Schedule mit geringerer Zykluszeit errechnet werden. Im Beispiel der Abbildung 4.45 sendet die Master-Achse zunächst in IRT-Übertragung 1 an ihre vier Slave-Achsen, worauf jede Slave-Achse nacheinander antwortet. Dies ist in den IRT-Übertragungen 2 bis 5 dargestellt. Da diese IRT-Übertragungen über einen Hub erfolgen, ist ihre Verzögerungszeit gering. Gleichzeitig kann jede dieser IRT-Übertragungen von allen Geräten am Hub empfangen werden, so dass ein Querverkehr ermöglicht wird. Die Frames der Slaves können also auch von den anderen Slaves ausgelesen werden. Zusätzlich dazu sendet die Master-Achse eine Statusmeldung zyklisch an die Steuerung (Kommunikation 6) und empfängt von der Steuerung ebenso zyklisch neue Sollpositionen (Kommunikation 7). Aufgrund der Halbduplexbetriebsart des Hubs stehen diese IRT-Übertragungen in Konflikt zueinander und können nicht gleichzeitig ausgeführt werden. Jede IRT-Übertragung verwendet also einen eigenen Zeitslot.

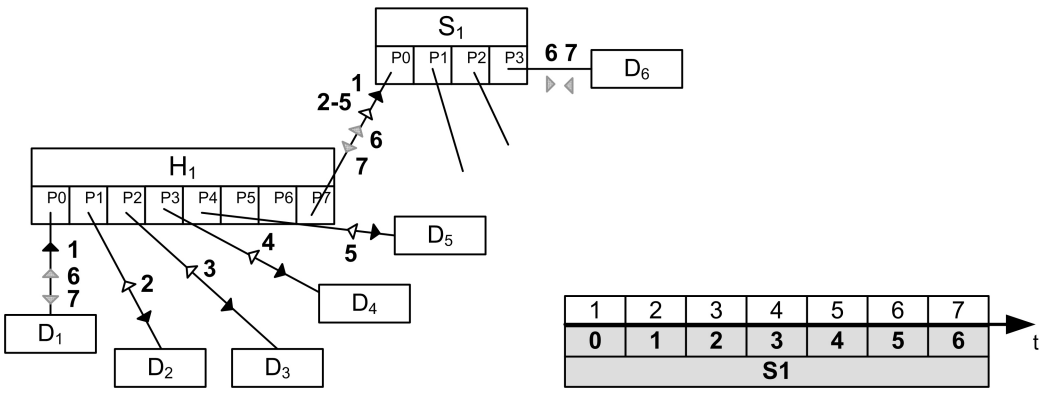


Abbildung 4.45: Master-Achse D1, 5 Slaves D2-D5 und Steuerung und Schedule

Nun ist es im realen Betrieb nicht notwendig, dass die IRT-Übertragung mit der Steuerung in jedem Zyklus erfolgen muss. Es ist ausreichend, dass die IRT-Übertragungen 6 und 7 beispielsweise in jedem achten Zyklus statt finden. Im Rahmen dieser Arbeit werden drei Wege diskutiert, wie dieser Fall in die Modellierung integriert werden kann.

4.7.1 Ersetzung mit asynchronen Übertragungen

Die erste Idee besteht darin, die nicht in jedem Zyklus verwendeten isochronen Zeitslots mit asyncronen Übertragungen zu füllen. Dadurch wird mehr Bandbreite des Netzwerkes für Standard-Ethernet Übertragungen zur Verfügung gestellt.

Wie bereits in Abbildung 4.34 dargestellt wurde, kann ein Hub mit seinen angeschlossenen Geräten und/oder Hubs als ein einzelnes Gerät betrachtet werden. Dort kann eine separate Berechnung einer lokalen Schedule erfolgen. Im Falle der Abbildung 4.45 ist der Port 0 von Switch 1 im Rahmen der Echtzeit-Schedule jeweils für 5 Zeitslots belegt. Dann erfolgt entweder der Datenaustausch mit der Steuerung oder es stehen jeweils 2 Zeitslots für asynchrone Kommunikation zur Verfügung.

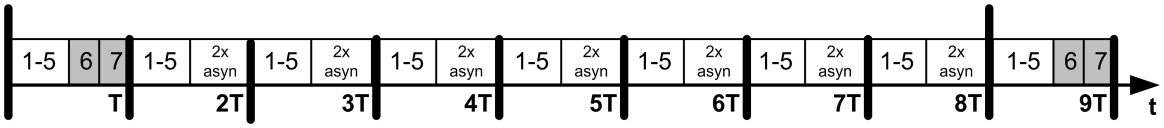


Abbildung 4.46: Multiplex-Schedule mit asynchroner Kommunikation

Eine Periodendauer T beträgt in diesem Beispiel $7 \cdot 6.72\mu s = 47.04\mu s$, wobei bislang keine Verzögerungszeiten der Verteiler berücksichtigt werden. Die Master-Achse kommuniziert also alle $47.04\mu s$ mit ihren Slaves. Die Gesamtschedule verläuft über $8T$. Die Master-Achse kommuniziert dabei alle $8 \cdot 47.04\mu s = 376.32\mu s$ mit der Steuerung, wozu 2 Zeitslots reserviert wurden mit den Übertragungen 6 und 7. In diese Kalkulation sind 14 asynchrone Slots einbezogen, welche $14 \cdot 6.72\mu s = 94.08\mu s$ und damit 1/4 der Zykluszeit benötigen.

Aus Sicht des Netzwerkes erfolgen zwei Übertragungen alle $376.32\mu s$, es müssen also alle 56 Zeitslots zwei Frames gesendet werden. Die restliche übergelagerte Schedule der Switches kann beliebig sein, solange dieses Kriterium eingehalten wird und keine sonstige Kommunikation über den Port 0 von Switch 1 verläuft. Die globale Schedule kann also wesentlich mehr Zeitslots für gleichzeitig zu übertragene Frames beinhalten. Dies funktioniert so lange problemlos, wie ein einzelner Bereich mit härtesten Echtzeitanforderungen - im Beispiel innerhalb des Hubs 1 - existiert.

In der Realität können jedoch mehrere Achsengruppen oder andere Geräte mit höchsten Echtzeitanfoderungen an das Netzwerk angeschlossen sein. In diesem Fall besteht eine mögliche Vorgehensweise darin, zunächst wie beschrieben alle einzelnen Schedules mit höchsten Echtzeitanforderungen zu berechnen. Im genannten Beispiel ergibt sich eine Periodendauer von 7 Zeitslots und eine Kommunikation in das auf Switches basierende Netzwerk alle 8 Zyklen. Resultiert aus einer anderen Achsengruppe eine andere Periodendauer, beispielsweise 8, so können diese Anforderungen bei der Berechnung der globalen Schedule kollidieren. Denn alle 56 Zyklen fallen die Übertragungen in dem auf Switches basierenden Bereich des Netzwerkes zusammen. Sind diese in Konflikt zueinander, so tritt dieser Konflikt an diesen Zeitpunkten auf. Ein Lösungsansatz besteht darin, zunächst alle lokalen Schedules der Hubs zu bilden und diese in einem zweiten Schritt zu synchronisieren. Für dieses Beispiel bedeutet dies, dass die Periodendauer des ersten Hubs auf 8 Slots erweitert wird. Dies kann durch Hinzufügen einer weiteren asynchronen Kommunikation geschehen. Als Alternative dazu können die lokalen Schedules auch als Vielfache zueinander angeordnet sein, also beispielsweise eine Schedule mit 8 und eine weitere mit 16

Zeitslots. Die kollidierenden Sendungen können dann mit einem festen Offset zueinander ausgeführt werden.

Die Zykluszeit der globalen Schedule kann in diesem Falle achtmal größer sein als die lokalen Schedules der Hubs, also 64 Zeitslots umfassen. In diesen Zeitslots sind aufgrund der Vollduplexfähigkeit der Switches parallele IRT-Übertragungen möglich, sofern diese nicht an die Achsen gerichtet sind. Die Achsen bleiben lediglich über die asynchronen Slots erreichbar.

4.7.2 Alternative Ausführung innerhalb eines Zyklus

Alternativ oder zusätzlich dazu kann eine Multiplex-Betriebsart also auf die gleiche Weise wie bei EPL eingeführt werden, sofern ausschließlich Hubs betroffen sind. Lediglich die Kommunikation der Master-Achse zu ihren Slaves muss in möglichst geringen Zeitintervallen erfolgen. Es ist auch nicht zwingend erforderlich, dass jedem Slave in jedem Zyklus die Möglichkeit zur Antwort an den Master und/oder zur Durchführung von Querverkehr eingeräumt wird. Da in diesem Beispiel aufgrund des Hubs jeder Slot lediglich aus einer Kommunikation besteht, können die jeweiligen Slots zusammengelegt werden. Die Einführung des Multiplex-Betrieb ist insbesondere bei Übertragungen über Hubs zu empfehlen, da dort keine Übertragungen parallel ausgeführt werden können, nur genau eine Übertragung zu einem Zeitpunkt statt findet und durch eine Zusammenlegung eine unmittelbare Optimierung der Schedule erfolgt. Die Abbildung 4.47 zeigt die resultierende Schedule für einen Zyklus.

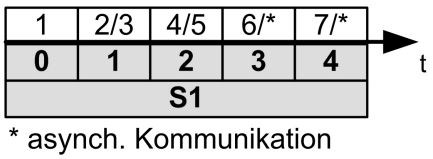


Abbildung 4.47: Gemultiplexte isochrone Übertragungen

Eine Periodendauer beträgt hier $5 \cdot 6.72\mu s = 33.06\mu s$, die Gesamtschedule verläuft über $2T$. Dadurch, dass nicht jeder Slave in jedem Zyklus sendeberechtigt ist, kann die Master-Achse alle $33.6\mu s$ zu den Slaves senden. Die Bandbreite für asynchrone Übertragungen wurde jedoch im Vergleich zur ersten Lösung reduziert. Sie wechseln sich mit der Kommunikation von und zu der Steuerung ab. Es sind bei $2T$ zwei asynchrone Übertragungen vorgesehen, die $13.44\mu s$ benötigen und damit $1/5$ der Bandbreite in Anspruch nehmen.

Eine weitere Variation besteht darin, jedem der vier Slaves lediglich in jedem vierten Zyklus die Sendeberechtigung zu gewähren und die und die Übertragungen zwischen dem Hub und Switch 1 nicht nacheinander auszuführen. Dies ist in Abbildung 4.48 dargestellt. Eine Periodendauer beträgt dann $3 \cdot 6.72\mu s = 20.16\mu s$, die Gesamtschedule verläuft über vier Perioden mit $80.64\mu s$. Dies ist zwar länger als im letzten Fall diskutiert wurde, die Master-Achse kann hier jedoch ihre Slaves alle $20.16\mu s$ aktualisieren.

In allen Fällen sollte beachtet werden, dass zur Synchronisation der globalen Schedule die längste lokale Schedule als Basis dient oder dass die lokalen Schedules in Vielfache angeordnet sind.

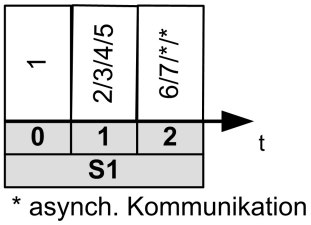


Abbildung 4.48: Optimierte Multiplex-Schedule

4.7.3 Multiplexing in geswitchten Netzwerken

Ein Multiplex-Betrieb innerhalb eines auf Switches basierenden Netzwerkes ist schwieriger zu handhaben, da in einem Zeitslot mehrere IRT-Übertragungen parallel übertragen werden. Die Zuweisung der Übertragungen zu den Zeitslots wird von den verwendeten Färbealgorithmen der Konfliktgraphen bestimmt und können daher unterschiedlich ausfallen. Die in Abbildung 4.49 nochmals dargestellte Schedule von Unicastübertragungen aus Abbildung 4.25 zeigt beispielsweise, dass 5 IRT-Übertragungen parallel ausgeführt werden.

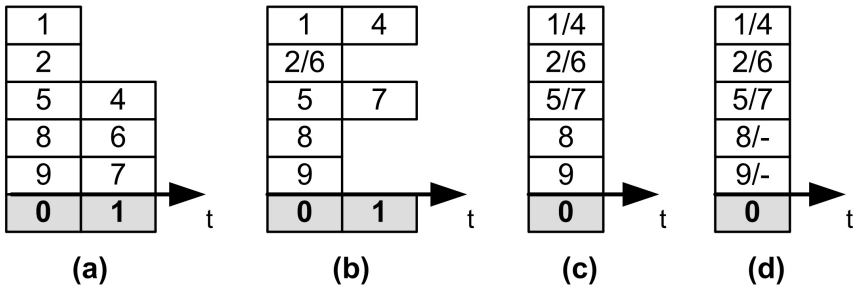


Abbildung 4.49: Vollduplex-Schedule von Unicast-Verbindungen

Angenommen, es ist ausreichend, die IRT-Übertragung 2 und 6 lediglich in jedem zweiten Zeitslot zu übertragen, so können diese jedoch nicht zwangsläufig zusammengefasst werden, s. Abbildung 4.49a. Werden beide IRT-Übertragungen abwechselnd in Slot 0 ausgeführt, so kann ein Konflikt der IRT-Übertragung 6 mit einer anderen IRT-Übertragung innerhalb dieses Zeitslots auftreten. Dies gilt auch, wenn die beiden Zeitslots vollständig zusammengefasst werden, wie es in Abbildung 4.49c dargestellt ist. Es ist immer noch möglich, dass die IRT-Übertragungen 8 und 9, die in jedem Zeitslot ausgeführt werden, mit einer der IRT-Übertragungen aus Slot 1 in Konflikt stehen. Eine Zusammenfassung ist also nur möglich, wenn die in den Zeitslot zu integrierende IRT-Übertragung ausschließlich mit der IRT-Übertragung in Konflikt steht, mit der sie abwechselnd ausgeführt wird. Dies lässt sich bei Unicastübertragungen im Vollduplexmodus leicht an den bipartiten Kantenkonfliktgraphen ablesen.

Eine weitere einfache Möglichkeit besteht auf den ersten Blick darin, die beiden Zeitslots vollständig alternieren zu lassen, wie es in der Abbildung 4.49d dargestellt wird. Hier besteht jedoch das Problem darin, dass in einer anderen lokalen Schedule zu diesem Zeitslot weitere IRT-Übertragungen hinzukommen können, die dann ebenfalls alterniert werden müssten.

Ein Multiplexing lässt sich jedoch durchführen, indem man zunächst die Schedules der Übertragungen berechnet, die in jedem Zeitslot ausgeführt werden müssen und alle Multiplex-Übertragungen von dieser Schedule ausnimmt. Im Anschluss daran können alle IRT-Übertragungen eingebunden werden, die in jedem zweiten Zeitslot ausgeführt werden sollen. Die Zeitslots können dann - wie in Abbildung 4.49d skizziert - global vereinigt werden. Die neue Schedule wird dabei zu der ersten Schedule addiert. Auf die gleiche Weise kann auch ein mehrfaches Multiplexing der IRT-Übertragungen realisiert werden.

Bei dieser Vorgehensweise ist es unter bestimmten Bedingungen möglich, dass die resultierende Schedule mit Multiplexing länger ist als die Schedule, in der jede IRT-Übertragung in jedem Zeitslot stattfindet. Dies kommt beispielsweise dann vor, wenn das Multiplexing keinen großen Anteil an der Gesamt-Kommunikation darstellt und die Multiplex-Übertragungen problemlos in die zuvor erstellte Schedule integriert werden könnten. In der Praxis ist es allerdings wahrscheinlicher, dass eine Vielzahl von IRT-Übertragungen nicht in jedem Zeitslot ausgeführt werden muss. Dies gilt insbesondere für Übertragungen, die nicht dem direkten Umfeld der Antriebsregelung zuzuordnen sind. Dabei handelt es sich unter anderem um Abfragen einzelner Sensoren, die in hoher Zahl in der industriellen Anlage integriert sind, dem Statusabgleich dezentraler Steuerungen sowie den zeitlich unkritischen Soll- und Istwertübertragungen. Bei dieser Art der Übertragungen ist üblicherweise auch keine Isochronität zwingend gefordert.

4.8 Berücksichtigung von Verzögerungszeiten

Die Modellierung ist bislang vereinfachend davon ausgegangen, dass die Verteiler die Übertragungen nicht verzögern. Dies ist in der Praxis nicht der Fall. Bereits in Kapitel 2.2.4.2 wurden typische Verzögerungszeiten von Standard-Ethernet Verteilern aufgezeigt, in Kapitel 3.1 typische Verzögerungen T_V von existierenden Lösungen im Umfeld des echtzeitfähigen Ethernets. Die Tabelle 4.50 stellt diese Zeiten nochmal dar.

Art des Verteilers	Verzögerungszeit T_V [µs]
Store-and-Forward Switch	16,00
Cut-Through Switch	10,00
ProfiNet Switch	3,00
Hub	0,40
EtherCAT-Verteiler	0,06

Abbildung 4.50: Typische Verzögerungszeiten von Verteilern

Es ist zu beachten, dass diese Zahlen lediglich Richtwerte heutiger Geräte darstellen. Sie entsprechen einer Übertragung eines Ethernet-Frames geringer Größe und wurden aufgerundet. Die reale Verzögerung eines store-and-forward Switches ist beispielsweise abhängig von der Framegröße. Ein Frame mit einer Größe von $1046 Byte$ verzögerte sich in einem Switch um $90\mu s$. Während die Angaben in der Literatur für einen cut-through Switch zwischen $10\mu s$ und $40\mu s$ liegen, wurden in der Praxis weniger als $10\mu s$ gemessen. Ein Hub erfüllt noch die IEEE-Norm bei einer Verzögerung von $0.92\mu s$, in der Praxis liegt sie bei $0.4\mu s$. Die Verzögerungen eines ProfiNet Switch und eines Gerätes einer EtherCAT-Linie

sind als relativ genaue Werte anzunehmen, da sie bereits unter harten Echtzeitbedingungen eingesetzt werden. Ein Ethernet-Kabel mit $100m$ Länge verzögert zusätzlich um ca. $0.5\mu s$, die jeweils zu der Verzögerung eines Switches addiert werden muss.

Modelliert man eine Linientopologie (vgl. Abbildung 4.33) mit 10 Verteilern, so ergibt sich eine Verzögerungszeit eines Frames vom sendenden Gerät am Anfang der Linie bis zum letzten empfangenen Gerät am Ende der Linie von $30\mu s$ bei der Verwendung von ProfiNet Switches, $4\mu s$ bei Hubs und $0.6\mu s$ bei der Verwendung von EtherCAT. Dazu müssen die durch die Verzögerungen der Kabel resultierenden Bitlaufzeiten addiert werden, die im Folgenden vernachlässigt werden. Die Verzögerungszeiten besitzen insbesondere dann einen großen Einfluss auf die erstellte Schedule, wenn von einer minimalen Framegröße ausgegangen wurde, die ca. $7\mu s$ für ihre Übertragung benötigt. Grundlegende Formeln für die Berechnung von Verzögerungszeiten wurden bereits von Popp [Pop05] für ProfiNet IRT vorgegeben.

Um mögliche Integrationen von Verzögerungszeiten in das Modell zu diskutieren, wird im weiteren Verlauf dieses Kapitels das folgende Beispiel eines auf Switches basierenden Vollduplexnetzwerkes mit Unicastübertragungen verwendet. Zur besseren Berechenbarkeit wird angenommen, dass alle Verteiler die gleiche Verzögerungszeit T_V besitzen.

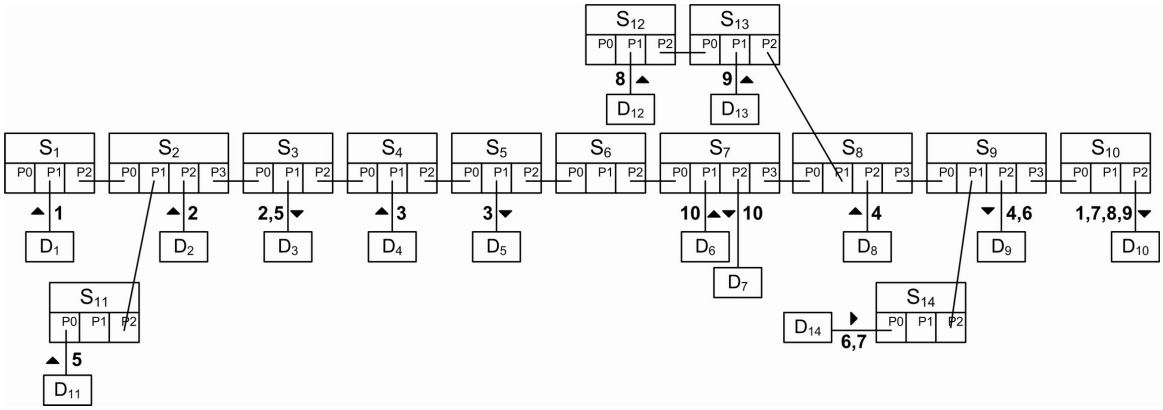


Abbildung 4.51: Verzögerungszeiten anhand eines Beispiels

Ein einfacher Ansatz besteht darin, zunächst die globale Schedule zu erstellen. Dies kann über den globalen Knotenkonfliktgraphen in Abbildung 4.52 geschehen, der greedy-gefärbt wird. Daraus resultiert die Schedule, die zunächst keine Verzögerungszeiten berücksichtigt.

Geht man von einer minimalen Framelänge aus, so beträgt die ideale Zykluszeit $4{\cdot}7\mu s = 28\mu s$. Die Verzögerungszeiten der Kabel werden im folgenden Beispiel vernachlässigt; die Dauer der IRT-Übertragungen von $6.72\mu s$ auf $7\mu s$ aufgerundet. Die IRT-Übertragungen verlaufen über 10 Verteiler mit einer jeweilgen Verzögerungszeit T_V. Basierend auf ProfiNet-Switches mit $T_V = 3\mu s$ ergibt sich eine maximale Gesamtverzögerungszeit von $T_{V,Ges} = 30\mu s$, die für die IRT-Übertragung 1 auch zutrifft, da diese über das gesamte Netzwerk verläuft. Dies bedeutet beispielsweise, dass die IRT-Übertragung 1 zu Beginn des Zyklus startet und erst nach $24\mu s$ bei Port 3 von Switch 8 weitergeleitet wird. Die IRT-Übertragung 9 startet zu Beginn des vierten Zeitslots, also bei $21\mu s$. Sie wird $2 \cdot T_V$ später, also nach $6\mu s$, am Port 3 von Switch 8 weitergeleitet. Dies entspricht einer absoluten Zeit von $27\mu s$. Damit kollidiert die IRT-Übertragung 9 mit der IRT-Übertragung

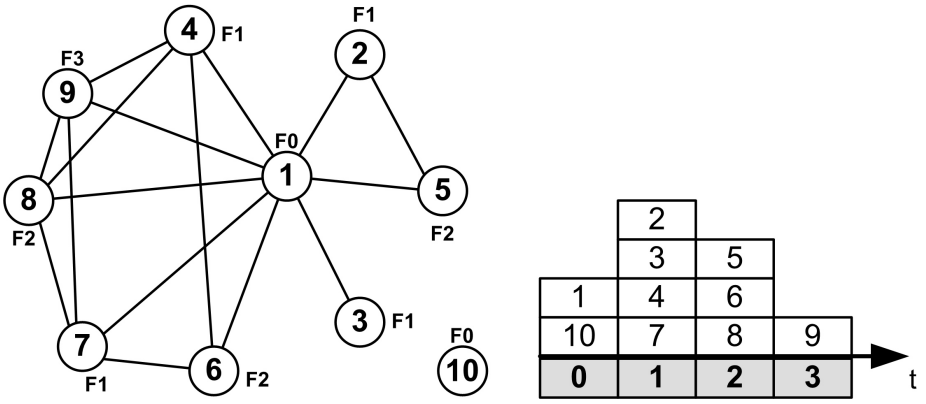


Abbildung 4.52: Globaler Knotenkonfliktgraph und resultierende Schedule

1, die gerade übertragen wird und deren Übertragung $7\mu s$ dauert. Als Resultat wird die IRT-Übertragung 9 in dem Puffer des Switches verzögert, obwohl die Schedule eingehalten wird.

Die Einbeziehung der Verzögerungszeit in die Schedule kann als erster Ansatz durch die Ausweitung der Zeitslots geschehen, so dass die Übertragungen nicht mehr in andere Zeitslots überlappen. Erweitert man jeden Slot pauschal um die maximale Verzögerungszeit im Netzwerk zuzüglich der Übertragungszeit $T_U = 7\mu s$ des Frames, so würde jeder Zeitslot $T_{V,Ges} + T_U = 37\mu s$ lang sein. Die Zykluszeit wird auf diese Weise von $28\mu s$ ohne Berücksichtigung der Verzögerungen auf $4 \cdot 37\mu s = 148\mu s$ erhöht.

4.8.1 Reduktion der Slotlänge

Dies lässt sich jedoch optimieren, indem man nicht pauschal den längsten theoretischen Übertragungsweg in jedem Zeitslot verwendet, sondern lediglich den längsten Übertragungsweg innerhalb eines Slots. Dies ist leicht möglich, da die betreffenden Switches in jeder Kommunikationslinie als Pfadangabe gespeichert werden. Im Beispiel der Abbildung 4.53 bedeutet dies, dass lediglich der erste Zeitslot maximal erweitert wird, da dort nur die IRT-Übertragung 1 über die maximale Ausdehnung des Netzwerks verläuft. Daraus ergibt sich bereits eine reduzierte Gesamtzykluszeit von $(10 + 3 + 5 + 4) \cdot T_V + 4 \cdot T_U = 94\mu s$.

In Abbildung 4.53 ist der Verlauf der Übertragungen über das Netzwerk dargestellt, wobei die Switches einzeln betrachtet werden. Auf der Zeitskala werden die Übertragungen mit ihren IDs dargestellt. Die benötigte Zeit für eine Übertragung setzt sich jeweils aus der Weiterleitung des Frames (weißer Bereich mit Kommunikations-ID) und der Verzögerungszeit des Switches (grauer Bereich) zusammen. Zum Vergleich ist oben links in der Abbildung die bislang berechnete Schedule ohne Berücksichtigung der Verzögerungszeiten dargestellt.

Zusätzlich werden unabhängige IRT-Übertragungen dargestellt, die verschiedene Ports eines Switches verwenden, beispielsweise die IRT-Übertragungen 4 und 7 in Slot 1 an Switch 9. Treten nebenläufige IRT-Übertragungen auf, so sind die Eingangs- und Ausgangsports der jeweiligen Switches mit aufgezeichnet worden. Beispielsweise verläuft die IRT-Übertragung 4 in Slot 0 im Switch 9 vom Eingangsport 0 zum Ausgangsport 2.

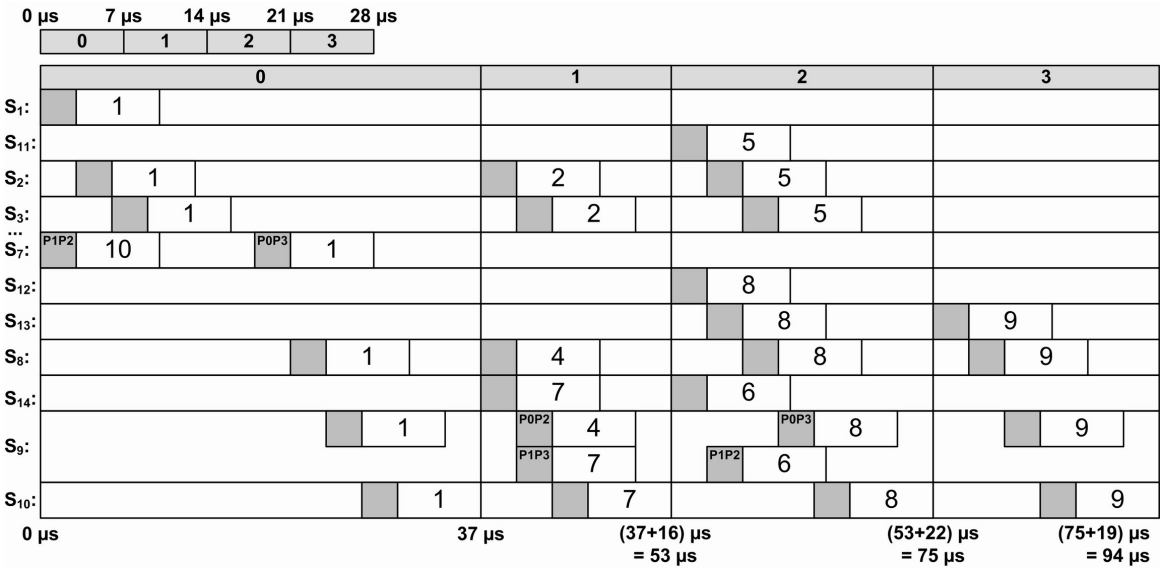


Abbildung 4.53: Schedule unter Berücksichtigung der Verzögerungszeiten

Auffallend sind die Slots 0 und 3, über die nur wenige IRT-Übertragungen verlaufen und die einen großen Anteil der Gesamtschedule bilden. Neben der Optimierung der Färbung, welche die Anzahl der Slots reduziert, ist demnach eine Optimierung der resultierenden Schedule unter Berücksichtigung der Verzögerungszeiten - insbesondere bei der Verwendung von Switches - sinnvoll.

Die Kommunikationslinie 1 verläuft über Switch 1 über die maximale Länge des Netzes bis zu Switch 10. In Switch 7 verläuft die IRT-Übertragung 10 parallel zu der IRT-Übertragung 1. Dies ist möglich, da beide Übertragungen über disjunkte Ports verlaufen.

Aufgrund der großen Anzahl von freien Bereichen lässt sich erahnen, dass diese Schedule weiter optimiert werden kann. Einerseits könnten diese freien Bereiche für asynchrone Übertragungen von kleinen Frames verwendet werden, die jeweils nur bis zum nächsten Switch weitergereicht und dort gespeichert werden. Ist der betreffende Ausgangsport des Switches für einen Zeitraum nicht mit Echtzeit-Kommunikation belastet, so kann der asynchrone Frame weiter geleitet werden. Andererseits besteht die Möglichkeit, weitere Heuristiken anzuwenden, um die Schedule zu optimieren, so dass eine kleinere Zykluszeit resultiert. Diese Heuristiken werden im Folgenden skizziert.

4.8.2 Vereinigung von Zeitslots

Bereits zur Synchronisation von unabhängigen Schedules im Halbduplexbetrieb sowie zur Optimierung einer gestaffelten Schedule wurde das Prinzip der Umsortierung von Zeitslots verwendet. Die erste mögliche Heuristik besteht darin, lediglich die Slots mit einer geringen Anzahl an IRT-Übertragungen paarweise zu untersuchen; im Beispiel sind dies die Slots 0 und 3. Ist dabei der eine Slot besonders lang und der andere besonders kurz, so ist die Wahrscheinlichkeit hoch, dass die beiden Slots vollständig zusammengefasst werden können, obwohl die dort enthaltenen Übertragungen prinzipiell in Konflikt zueinander stehen.

In Abbildung 4.54 wird der Slot 3 direkt hinter Slot 0 einsortiert und überprüft, ob eine Zusammenlegung erfolgen kann. Ob zwei Kommunikationslinien in Konflikt stehen oder nicht, kann dem Konfliktgraphen entnommen werden. Nimmt man an, dass die Verzögerungszeit der Switches für jede IRT-Übertragung konstant ist, so muss beim Vergleich von zwei Kommunikationslinien lediglich jeweils ein Switch betrachtet werden. Im Beispiel kann IRT-Übertragung 9 direkt hinter IRT-Übertragung 1 ausgeführt werden, wie es im ersten Schritt dargestellt ist. Eine weitere Vorverlegung von $1\mu s$ bis $10\mu s$ führt zu einem Konflikt zwischen den beiden Übertragungen.

Es ist jedoch eine Vorverlegung von IRT-Übertragung 9 vor die IRT-Übertragung 1 möglich, so dass eine weitere Optimierung der Zykluszeit resultiert. Durch die zweite Optimierung wird der Zeitslot 3 aufgehoben, wodurch die Schedule um weitere $19\mu s$ verkürzt wird.

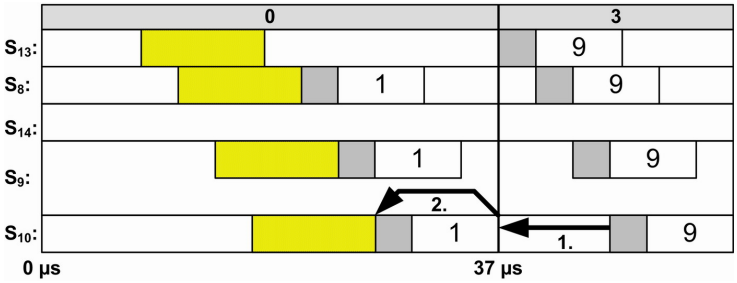


Abbildung 4.54: Vereinigung von zwei Zeitslots

In diesem Fall ist also für die zeitliche Entzerrung der Konflikte nicht zwingend eine Zuweisung verschiedener Zeitslots notwendig. Statt dessen wird die zeitliche Entzerrung bereits durch die unterschiedlichen Verzögerungszeiten vorgenommen. Zu Beginn eines neuen Zyklus sind lediglich die Geräte zu synchronisieren, was ähnlich wie bei EPL durch einen als Broadcast versendeten SoC-Frame geschehen kann.

Tritt bei der Suche nach einem freien zeitlichen Bereich ein Konflikt auf, so kann der Algorithmus die Anzahl an μs fortschreiten, wie der Switch mit der jeweiligen Übertragung ausgelastet ist, mit der die gerade betrachtete Übertragung in Konflikt steht. So kann im dargestellten Beispiel zwischen Schritt 1 und 2 die zeitliche Distanz von $10\mu s$ bei einer Suchiteration überbrückt werden. Diese Zeit setzt sich zusammen aus den benötigten $7\mu s$ für die Weiterleitung des Frames zuzüglich $3\mu s$ für die Verzögerungszeit des betreffenden Switches 10. Nach dem in Abbildung 4.54 dargestellten zweiten Schritt besteht an Switch 10 kein Konflikt mehr, so dass die IRT-Übertragung 9 unmittelbar vor der IRT-Übertragung 1 eingefügt werden kann. Es ist zu bedenken, dass vor der endgültigen Einfügung der IRT-Übertragung 9 in die Gesamtschedule geprüft werden muss, ob diese einen Konflikt an einem anderen Switch - im Beispiel an Switch 13, 8, 14 oder 9 - auslöst. Ist dies der Fall, so muss im weiteren Verlauf des Algorithmus die Übertragung an diesem konfliktauslösenden Switch betrachtet werden. Dies geschieht so lange, bis bei allen Switches ein freier zeitlicher Bereich gefunden wurde oder der Beginn der Schedule erreicht ist. Im zweiten Fall kann die IRT-Übertragung nicht vorverlegt werden.

Es ist zu erwarten, dass diese vorgestellte Heuristik in polynomieller Zeit abgearbeitet werden kann, da keine Permutationen von Lösungsmöglichkeiten betrachtet werden.

4.8.3 List-Scheduling

Eine weitere Optimierung der Schedule besteht darin, stets den frühest möglichen Startzeitpunkt für eine IRT-Übertragung zu ermitteln. Die IRT-Übertragung 2 steht beispielsweise nur mit den Übertragungen 1 und 5 in Konflikt. Dies kann dem globalen Knotenkonfliktgraphen entnommen werden. Aus der Schedule der Abbildung 4.53 lässt sich entnehmen, dass diese IRT-Übertragung 2 lediglich über die Switches 2 und 3 verläuft. Vor dem Start dieser IRT-Übertragung existiert eine große Zeitspanne ohne Kommunikation in diesen beiden Switches. Die IRT-Übertragung 2 kann somit vorverlegt werden, bis ihr Startzeitpunkt an den Endzeitpunkt einer IRT-Übertragung stößt, mit der die IRT-Übertragung 2 in Konflikt steht. Ebenso kann auch die IRT-Übertragung 5 vollständig in den ersten Zeitslot vorverlegt werden können. Es ist also naheliegend, dass die Heuristik, eine IRT-Übertragung stets zu ihrem frühest möglichen Zeitpunkt auszuführen, zu einer Optimierung der Schedule führt. Die Grenzen der Zeitslots werden dabei aufgehoben, so dass man von einer einzigen IRT-Phase ohne weitere Unterteilung ausgehen kann. Getaktet werden kann diese Phase jeweils zu Beginn eines neuen Zyklus.

Die Einordnung der Übertragungen nach ihrem frühest möglichen Zeitpunkt ist mit einem List-Scheduling bzw. einer First-Fit Vorgehensweise zu vergleichen, das aufgrund der im Vorfeld bekannten IRT-Übertragungen offline durchgeführt werden kann. Die bereits beschriebene vollständige Auflösung der Zeitslots hat zur Folge, dass eine Färbung des Konfliktgraphen nicht mehr benötigt wird. Die Erstellung des globalen Knotenkonfliktgraphen ist dennoch sinnvoll, um die Konflikte zwischen einzelnen Kommunikationslinien bzw. Kommunikationsbäumen zu erkennen. Im Folgenden wird beschrieben, wie die Schedule aus den gegebenen IRT-Übertragungen und dem globalen Knotenkonfliktgraphen erstellt werden kann.

Ähnlich wie bei der Graphenfärbung werden zunächst die IRT-Übertragungen in eine Reihenfolge vorsortiert. Entsprechend dieser Reihenfolge werden die IRT-Übertragungen nacheinander direkt in die Schedule eingefügt unter der Prämisse der möglichst frühen Sendung. Es wird also vom Beginn der Schedule bei $0\mu s$ untersucht, ob die gerade betrachtete IRT-Übertragung eingefügt werden kann oder nicht. Ist dies nicht der Fall, so wird eine erneute Untersuchung unmittelbar nach Beendigung der Kommunikation, mit welcher die betrachtete Kommunikation in Konflikt steht, durchgeführt. Dies geschieht so lange über alle Switches, die von der betrachteten IRT-Übertragung betroffen sind, bis ein freier Bereich innerhalb der Schedule gefunden wird. Diese Vorgehensweise ist mit dem Greedy-Algorithmus der Graphenfärbung vergleichbar. Wie bereits beschrieben wurde, existert beim Greedy-Algorithmus der Graphenfärbung mindestens eine Vorsortierung, welche zu einem optimalen Ergebnis führt. Daher ist anzunehmen, dass eine solche optimale Vorsortierung auch bei der Anwendung des List-Schedulings existiert. Da die Graphenfärbung in diesem Fall der Berücksichtigung der Verzögerungszeiten von der Optimierung durch das List-Scheduling abgelöst wird, besteht die Aufgabe darin, eine optimale Vorsortierung der IRT-Übertragungen zu ermitteln.

Im Gegensatz zu einer zufälligen Anordnung der Übertragungen können die Übertragungen beispielsweise nach ihrer Länge sortiert werden, wobei die längste Übertragung zuerst in die Schedule eingefügt wird. Bei Betrachtung des Beispiels aus Abbildung 4.53 ist ersichtlich, dass das späte Einfügen der IRT-Übertragung 1 - welche über die Switches 1 bis 10 verläuft - mit hoher Wahrscheinlichkeit nicht zu einer optimalen Schedule führt.

Das Ermitteln der Anzahl an Hops einer IRT-Übertragung ist leicht durchzuführen, da die Anzahl der von der IRT-Übertragung betroffenen Switches in der Datenstruktur der Kommunikationslinie bereits festgehalten ist (vgl. Kapitel 4.1.2). Es handelt sich hierbei um eine statische Vorsortierung unter der Annahme, dass lange IRT-Übertragungen schwer in die Schedule zu integrieren sind.

Die Anzahl an Hops ist jedoch nicht direkt für eine schwierige Integration in die Schedule verantwortlich. Präziser formuliert geht man implizit davon aus, dass eine IRT-Übertragung, welche über viele Switches verläuft, auch gleichzeitig ein hohes Potential an Konflikten beeinhaltet. Die Anzahl der Konflikte einer IRT-Übertragung kann jedoch leicht anhand des Grades $d(v)$ des Knotens $v \in V$, der die IRT-Übertragung im globalen Knotenkonfliktgraphen $G = (V, E)$ repräsentiert, ermittelt werden. Es ist also sinnvoller, die Anzahl der Konflikte von jeder IRT-Übertragung aus dem Knotenkonfliktgraphen zu ermitteln und die Reihenfolge der IRT-Übertragungen dementsprechend anzuordnen. Auch hierbei handelt es sich um eine statische Sortierung.

In Abbildung 4.55 wird zunächt die Anzahl der Konflikte jeder IRT-Übertragung anhand des globalen Knotenkonfliktgraphen ermittelt. Die Anzahl ist jeweils an den Knoten, welche die jeweilige Übertragung repräsentiert, verzeichnet. Im Anschluss daran werden die Übertragungen anhand der Anzahl der Konflikte nacheinander in die Schedule eingefügt, wobei mit der IRT-Übertragung mit den meisten Konflikten begonnen wird. Besitzen mehrere IRT-Übertragungen die gleiche Anzahl an Konflikten, so wurden sie anhand ihrer ID eingeordnet, wobei mit der geringsten ID begonnen wurde. Die dargestellte resultierende Schedule liegt bei $47\mu s$ im Vergleich zu $148\mu s$ des ersten beschriebenen Ansatz. Die Verzögerungszeiten verlängern die idealisierte Schedule, die $28\mu s$ lang ist, um 68% bei der Anwendung des List-Schedulings. Der erste beschriebene Ansatz hingegen hat die Verzögerungszeiten pauschal auf jeden Zeitslot addiert und die Schedule dabei um 529% erhöht.

Eine weitere Optimierung der Vorsortierung kann darin bestehen, lediglich die Anzahl der Konflikte der Übertragungen zu zählen, welche noch nicht in die Schedule integriert worden sind und die Knoten der bereits eingefügten Übertragungen mit den inzidenten Kanten aus dem Konfliktgraphen zu entfernen. Dies entspricht einer dynamischen Sortierung und ist mit dem Sättigungsgrad des DSATUR-Algorithmus der Graphenfärbung zu vergleichen, der in der Praxis gute Ergebnisse erzeugt.

Eine weitere interessante Eigenschaft des List-Schedulings besteht darin, dass es unabhängig von der Framelänge operieren kann. Die Framelänge kann also für jeden Frame beliebig sein, da im Falle eines Konfliktes ein weiterer Versuch zur Einsortierung nach Beendigung des in Konflikt stehenden Frames unternommen wird. Die in Kapitel 4.6 vorgestellte gestaffelte Framelänge ist damit obsolet. Ebenso kann die Verzögerungszeit bei Anwendung des List-Schedulings für jeden Switch unterschiedlich sein, da sie algorithmisch zur benötigten Zeit für das Weiterleiten des Frames addiert wird. Wichtig ist lediglich, dass die Verzögerungszeit jedes Switches für alle Frames möglichst konstant ist, also über einen geringen Jitter verfügt. Betrachtet man zusätzlich die Jitter, so müssen diese als Worst-Case Sicherheitsabstände in das List-Scheduling integriert werden.

Ebenso können bei der Anwendung des List-Schedulings neben den Unicastübertragungen, welche zu Kommunikationslinien führen, auch Multicast- und Broadcastübertra-

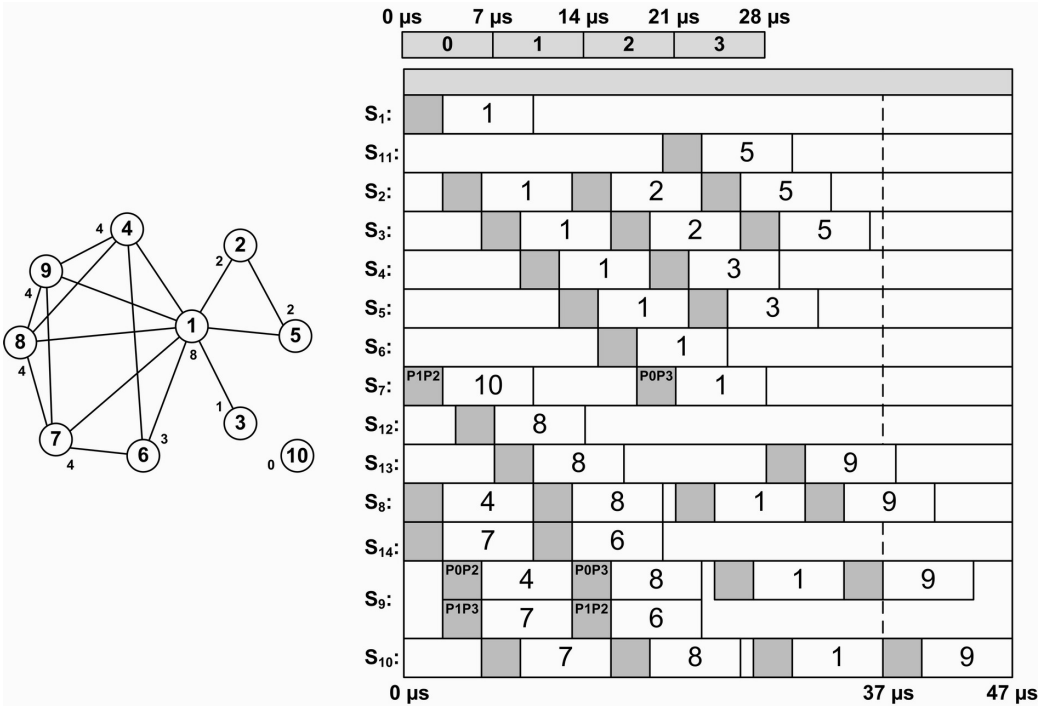


Abbildung 4.55: Optimierte Schedule mit Verzögerungen

gungen berücksichtigt werden. Dabei erhöhen sich lediglich die Anzahl der betroffenen Switches sowie die Anzahl der Konflikte im globalen Knotenkonfliktgraphen.

Werden an den Rändern des Netzwerkes Hubs eingesetzt, so können diese jeweils wie ein einzelnes Gerät betrachtet werden. Die hardwarebedingten Jitter und die möglichen Synchronisationsabweichungen können auch hier als worst-case Berechnung in die Optimierung der Schedule einbezogen werden. Im Vergleich zu den Verzögerungen sollten diese jedoch gering sein, zumal etablierte Protokolle wie IEEE 1588 unter Verwendung eines Hardwarebausteins sehr genaue Synchronisierungen ermöglichen. Für die Synchronisierung kann ein zusätzlicher fester Zeitbereich zu Beginn jedes Zyklus reserviert werden; eine grundlegende Synchronisierung kann in der Hochlaufphase der Anlage erfolgen.

Ebenso ist es möglich, den Datenaustausch über andere etablierte Lösungen wie EtherCAT in die Modellierung einzubeziehen. Der umlaufende Frame verursacht geringe Verzögerungszeiten an den FMMU-Anbindungen der einzelnen Geräte (s. Kapitel 3.1.3.2). Das EtherCAT-Subnetz besitzt über den Master eine einzelne Anbindung nach außen, über die Übertragungen von/zu anderen Geräten ermöglicht werden. Diese Übertragungen können auf die gleiche Weise berechnet und ggf. optimiert werden, wie es in diesem Kapitel vorgestellt wurde.

4.9 Zusammenfassung

Die Modellierung des Netzwerkes und der Konflikte hat im Falle der idealisierten Halbduplexübertragung ergeben, dass die Switches unabhängig voneinander betrachtet werden können und dass die erzeugten lokalen Schedules durch Vertauschen ganzer Zeitslots

synchronisierbar sind. Der Konfliktgraph kann Kreise ungerader Länge enthalten. Eine exakte Färbung ist daher NP-vollständig, jedoch hat bereits eine Greedy-Färbung akzeptable Resultate geliefert. Bei der Färbung kann auf eine Vielzahl bekannter Heuristiken zurückgegriffen werden, die im Allgemeinen gute Resultate liefern.

Aus der idealisierten Vollduplexübertragung entstehen bipartite Kantenkonfliktgraphen, die in poynominieller Zeit exakt färbbar, jedoch nicht synchronisierbar sind. Hier wurde skizziert, dass man die Farben von bereits gefärbten Kommunikationslinien in einer lokalen Schedule auf andere lokale Schedules übertragen kann.

Während IRT-Broadcastübertragungen leicht in das Modell integriert werden können, da diese stets einen Zeitslot exklusiv verwenden, gilt dies für IRT-Multicastübertragungen nicht. Diese können an einem Switch parallel ausführbar sein, an einem anderen Switch jedoch in Konflikt zueinander stehen. Daher können die Switches nicht mehr unabhängig voneinander betachtet werden. Dazu kann ein globaler Konfliktgraph erstellt werden, wobei sich der Knotenkonfliktgraph besonders eignet.

Hubs sind aufgrund ihrer geringen Verzögerungszeit für Anwendungen mit harten Echtzeitanforderungen interessant, lassen jedoch keine parallelen IRT-Übertragungen konfliktfrei zu. Es wurde gezeigt, auf welche Weise Hubs in die Modellierung des Netzwerkes integriert werden können. Eine Möglichkeit besteht darin, das gesamte Netzwerk in den Halbduplexbetrieb umzuschalten und die Schedules der Switches unabhängig zu betrachten. Der Kantenkonfliktgraph kann in diesem Fall Schlingen enthalten.

Die restliche Bandbreite des Netzes kann dann für asynchrone Datenübertragungen verwendet werden. Diese Methode eignet sich besonders dann, wenn nur eine geringe Anzahl an Switches im Netzwerk vorhanden ist. Alternativ dazu kann wiederum mit einem globalen Konfliktgraphen gearbeitet werden. Die Hubs mit ihren angeschlossenden Geräten werden dabei als ein einziges Gerät betrachtet. Auf die gleiche Weise können auch andere Lösungen wie EtherCAT in die Modellierung integriert werden.

Asynchrone Datenübertragung kann - wie es bereits bei existierenden Lösungen geschieht - durch zusätzliche Zeitslots hinzugefügt werden. Es ist jedoch auch denkbar, dass asynchrone Frames in der isochronen Phase übertragen werden, sofern die entsprechenden Ports der Switches nicht unmittelbar mit Echtzeitübertragungen beschäftigt sind. Die Frames können dabei Hop-by-Hop von einem Switch sum nächsten übertragen und dort jeweils zwischengespeichert werden, bis der betreffende Ausgangsport für eine Frame-Übertragung frei ist. Eine Optimierung der Echtzeitschedule kann dahingehend erfolgen, dass Slots mit identischen freien Ports hintereinander ausgeführt werden, damit größere asynchrone Frames ohne Fragmentierung übertragen werden können. Dies kann durch die Permutation der Zeitslots in der globalen Schedule geschehen.

Da bei asynchronen Übertragungen meist große und bei isochronen Übertragungen meist kleine Frames (vgl. Abbildung 1.2) versendet werden, wurden variable Framegrößen in die Modellierung einbezogen. Dazu wurden bis zu 5 Slotgrößen definiert, die sich in ihrer Größe jeweils um eine Zweierpotenz unterscheiden. Dadurch können Übertragungen unterschiedlicher Länge geschachtelt werden. In der resultierenden gestaffelten Schedule können kleinere Slots jeweils so permutiert werden, dass längere Übertragungen konfliktfrei darüber gelagert werden können, sofern dies möglich ist. Dies wurde als Bottom-Up Erweiterung bezeichnet.

Aufgrund dessen, dass lediglich im besonders zeitkritischen Umfeld der Antriebstechnik Frames in jedem Produktionszyklus versendet werden, wurden Sendungen in Vielfachen von Produktionszyklen diskutiert. So müssen beispielsweise Temperaturen und Druck-Werte nicht in jedem Zyklus ausgelesen werden. Um diese Anforderung zu erfüllen, wurden zwei prinzipielle Wege dargelegt. Der erste Weg führt entweder eine IRT-Übertragung aus und abwechselnd dazu eine asynchrone Übertragung von gleicher Länge. Der zweite Weg beschreibt die alternative Ausführung von zwei IRT-Übertragungen. Während das Multiplexing bei Hubs leicht einzuführen ist, ist dies bei Switches schwieriger. Als mögliche Lösung wurde vorgeschlagen, zunächst die Schedule der IRT-Übertragungen, die in jedem Zeitslot ausgeführt werden, zu erstellen. Im Anschluss daran wird die Schedule der Übertragungen aus jedem zweiten Zeitslot berechnet usw. Die einzelnen Schedules werden abschließend hintereinander ausgeführt.

Um die Realitätsnähe der Modellierung zu erhöhen, wurden die Verzögerungszeiten der Switches in das Modell einbezogen, welche die Schedule erheblich verlängern können. Es wurden Heuristiken vorgestellt, die diese Erweiterung minimieren und die Grenzen der Zeitslots aufheben können. Auch wenn die Graphenfärbung bei der Anwendung des List-Schedulings obsolet wird, ist es dennoch sinnvoll, zunächst den globalen Knotenkonfliktgraphen zu erstellen. Denn dadurch wird die Ausführung einer Heuristik für eine geeignete Vorsortierung der IRT-Übertragungen erleichtert, welche zu einer Optimierung der Schedule beiträgt. Zusätzlich zu der Einbindung unterschiedlicher Verzögerungszeiten in die Modellierung ermöglicht das List-Scheduling ebenso die problemlose Einbeziehung von Multicast-Übertragungen und ist daher vielversprechend für eine detaillierte Betrachtung im Rahmen zukünftiger Forschungsarbeiten.

Zusammenfassend ist zu sagen, dass das entwickelte Modell von den gängigen Technologien wie Ethernet PowerLink, ProfiNet und EtherCAT abstrahiert und diese im Rahmen eines formalen Frameworks als spezielle Lösungen integriert. Es ist möglich, komplexe Netzwerke zu modellieren und aufgrund der Anforderungen der konkreten Anlage - minimale Verzögerungszeiten bis hin zu paralleler Kommunikation - zu konzeptionieren. Bei diesem Entwurf sind konkrete Technologien ebenso wie strenge Subnetzbildung zunächst untergeordnet. Diese können ausgewählt werden, sobald die resultierende Zykluszeit aus der Modellierung für die Anlage zufriedenstellend ist.

Der wissenschaftliche Beitrag des in diesem Kapitel vorgestellten, formal fundierten Frameworks besteht in der allgemeinen Modellierung von echtzeitfähigen Ethernet-Netzwerken in der Automatisierungstechnik. Diese Modellierung abstrahiert von einer konkreten technologischen Implementierung und bietet Lösungsansätze von idealisierten Übertragungen und Netzwerken (s. Kapitel 4.1 sowie 4.2) bis hin zu realitätsnahen Anwendungen (s. Kapitel 4.7 ff.). Dazu wurde in jedem Unterkapitel ein Katalog von Verfahren[3] entwickelt, wie man ausgehend von einer gegebenen Netzwerkinfrastruktur und IRT-Übertragungen Schedules erstellen kann, mit denen ein deterministischer Medienzugang ermöglicht wird. Mit Hilfe dieser Verfahren kann in Abhängigkeit der konkreten Echtzeitanforderungen einer automatisierten Anlage ein individueller Kompromiss zwischen Kompatibilität zu Standard-Ethernet und Echtzeitfähigkeit sowie zwischen asynchronen und isochronen Übertragungen ermittelt werden.

[3]vgl. beispielsweise Abbildungen 4.16, 4.31, 4.32, 4.44, 4.55

Kapitel 5

Technische Realisierbarkeit

In diesem Kapitel werden Szenarien diskutiert, in welcher Weise die Schedules, die in der formalen Modellierung beschrieben wurden, umgesetzt werden können. Zunächst wird der Prototyp eines Simulators vorgestellt, mit dessen Hilfe das in Kapitel 4 beschriebene formale Framework umgesetzt werden kann.

Im Kapitel 5.2 werden mögliche Orte untersucht, an denen die Schedules aktiv werden. Die Wahl des Ortes hat dabei eine direkte Auswirkung auf das Verhalten des Netzwerks bezüglich seiner Echtzeitfähigkeit, aber auch bezüglich des Verhaltens gegenüber asynchronen Sendungen und Geräten, die keine Kenntnis der Schedules besitzen, wie übliche PCs oder Laptops mit Ethernet-Schnittstelle. Die Frage lautet dabei, ob sich das Netzwerk gegenüber diesen Geräten transparent verhält und ob asynchrone Sendungen dieser Geräte das Echtzeitverhalten des Netzwerkes beeinflussen oder nicht. Falls asynchrone Sendungen das Echtzeitverhalten beeinflussen würden, so müssen die sendenden Geräte Kenntnis der Schedule, z. B. durch den Einsatz eines eigenen Treibers, besitzen.

Die Integration der Schedules in den Switches, wie es bei Siemens ProfiNet angewendet wurde, ist ein vielversprechender Ansatz. Im Kapitel 5.3 werden erste Forschungsergebnisse zur Implementierung einer Schedule in einem Switch vorgestellt, da die Umsetzung eines ProfiNet-Switches - beispielsweise im ERTEC 400 - proprietär ist. Diese ersten Ergebnisse skizzieren Möglichkeiten, aber auch technische Probleme und deren Lösungsansätze der Einbindung von Schedules in die Switches zur Erreichung der Echtzeitfähigkeit.

5.1 Simulation des Frameworks

Um die Anwendung der formalen Modellierung zu erleichtern, wurden im Rahmen dieser Arbeit zwei Vorgehensweisen untersucht. Der erste Ansatz besteht in der Eigenentwicklung eines Netzwerksimulators, während der zweite Ansatz die Integration der in der Modellierung vorgestellten algorithmischen Abläufe (vgl. Abbildungen 4.16, 4.31 und 4.32) in einen vorhandenen Netzwerksimulator beschreibt.

5.1.1 NetSim und der Graphical Schedule Manager (GSM)

Das Ziel eines eigenen Netzwerksimulators besteht darin, eine Netzwerkinfrastruktur und IRT-Übertragungen vorgeben zu können und daraus Schedules für die Switches zu erzeu-

gen. Die notwendigen Algorithmen, welche sequentiell angewendet werden, sollen dabei modular ausführbar sein. Das Ergebnis ist eine objektorientierte prototypische Anwendung [Sch07b], die aus zwei Komponenten besteht und deren Struktur in Abbildung 5.1 skizziert ist.

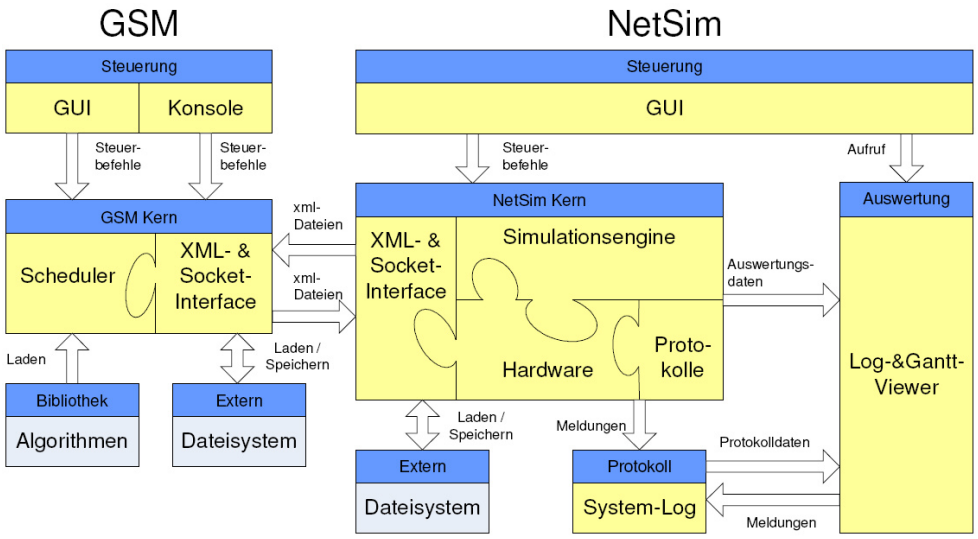


Abbildung 5.1: Komponenten des Netzwerksimulators

Der Simulator NetSim ist betriebssystemunabhängig in Java implementiert und beinhaltet die Benutzerschnittstelle. Von dort aus kann die Netzwerkinfrastruktur aufgebaut werden, die Hubs, store-and-forward und cut-through Switches, RT-Switches mit einer vorgegebenen Schedule, nRT- und RT-Geräte sowie Netzwerkkabel enthalten kann. Jede Hardware-Komponente ist parametierbar. So können beispielsweise Verzögerungszeiten definiert werden. Durch die objektorientierte Struktur können weitere Geräte hinzugefügt werden.

Im Anschluss an die Parametierung der Infrastruktur können Kommunikationslinien definiert werden, wobei das Quell- und das Zielgerät angegeben werden muss. Es können bisher Unicastübertragungen definiert werden, die zu jedem Produktionszyklus ausgeführt werden.

Da neben isochronen Echtzeitdaten auch asynchroner Verkehr simuliert werden soll, wurden Klassen zur Protokollverwaltung eingeführt und darauf aufbauend bislang Ethernet II und IEEE 802.3 Frames sowie die aufbauenden Protokolle ARP, IP und ICMP entsprechend der RFCs modular realisiert. Zusätzlich dazu wurde exemplarisch ein eigenes Echtzeitprotokoll REAL entwickelt, welches auf IEEE 802.3 Frames basiert.

Ein solches Netzwerk kann für eine vorgegebene Zeit in der ereignisorientierten Simulationsengine simuliert werden. Die Simulation basiert auf einer Verhaltensbeschreibung der Hardware. So werden die Frames als Objekte über die Netzwerkkomponenten transportiert, indem Referenzen auf die Frame-Objekte weitergereicht werden. Bei jeder Referenzübergabe wird ein Ereignis ausgelöst, welches jeder betroffenen Komponente mitgeteilt wird.

Die Ergebnisse der Simulation werden aufgezeichnet und und sind im Log-Viewer visualisierbar. Des Weiteren kann der Verlauf der Frames in einem Frame-Log dargestellt

werden, der sich an die Ethereal-Visualisierung anlehnt (s. Abbildung 5.2). Dieser Verlauf zeigt im Gegensatz zu Ethereal nicht nur die Protokollierung eines Frames an einer Ethernet-Schnittstelle, sondern den gesamten Verlauf des Frames im Netzwerk. In einem Gantt-Viewer kann bislang die Schedule jedes einzelnen Switches eingesehen werden.

Für eine zukünftige Betrachtung ist die Erweiterung des Gantt-Viewers zu fokussieren, um den Verlauf von IRT-Frames über das gesamte Netzwerk incl. seiner Verzögerungszeiten darstellen zu können. Damit kann in Zukunft eine Darstellung wie in Abbildung 4.53 ermöglicht werden, die zur Optimierung der globalen Schedule unter Berücksichtigung der Verzögerungszeiten beitragen kann.

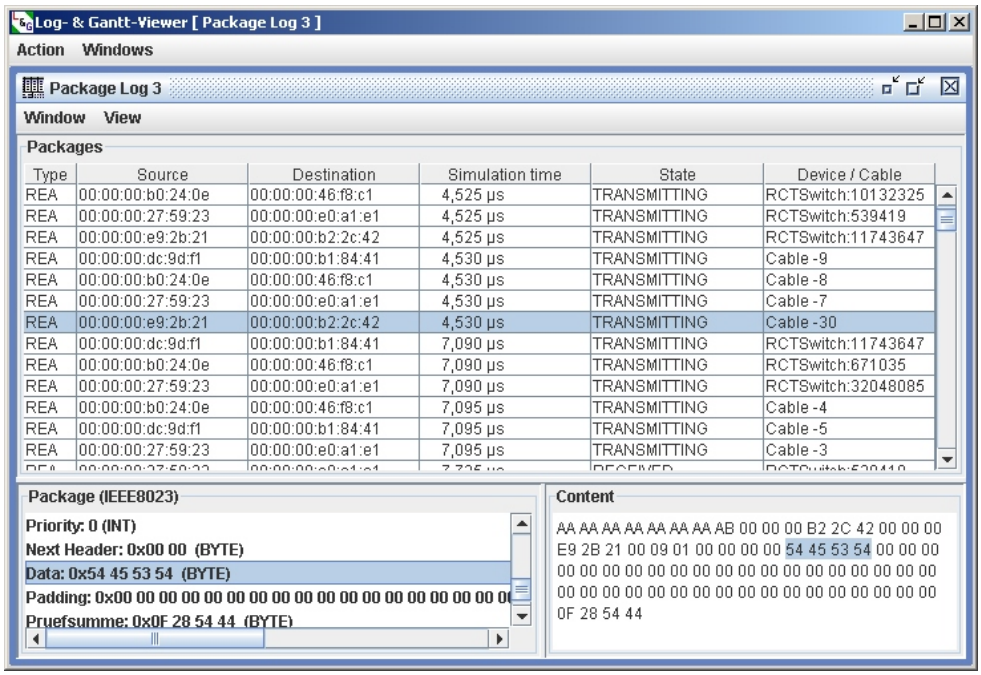


Abbildung 5.2: Ausgabe des Frame-Log

Bei der zweiten Komponente des Simulation, GSM, handelt es sich um ein Framework zur Verwaltung der Algorithmen, mit denen die Schedules erstellt werden. Die Zeit, welche jede Bibliothek für die Berechnung benötigt, kann gemessen und für Optimierungszwecke der Algorithmen ausgewertet werden. Ein Teil der in der Modellierung vorgestellten Vorgehensweisen (s. Kapitel 4.1) wurde im Rahmen einer Machbarkeitsanalyse als GSM-Bibliotheken erfolgreich implementiert. Jeder Algorithmus, beispielsweise die Errechnung der Kommunikationslinien aus den gegebenen Quell- und Zielports sowie der Netzwerkinfrastruktur oder die Greedy-Färbung eines Graphen, wurde dabei als eigene Bibliothek[1] umgesetzt. Jeder Bibliothek wird bei ihrem Aufruf ein Datencontainer-Objekt übergeben, auf das sie begrenzt zugreift. So werden notwendige Parameter aus dem Container ausgelesen und Ergebnisse in vorgegebene Datenstrukturen abgelegt. Diese Ergebnisse können dann von einer anschließend aufgerufenen Bibliothek weiterverwendet werden.

Das Netzwerk und die Kommunikationslinien werden als XML-Dateien verwaltet und können als Datenströme mit GSM über Socketverbindungen ausgetauscht werden. GSM

[1] als DLL-Datei für Windows bzw. SO-Datei für Linux

ist zur effizienten Realisierung der Algorithmen in C++ unter Verwendung von QT 4.0.1 [Tro05] plattformunabhängig entwickelt und besteht neben der Benutzeroberfläche[2] im Wesentlichen aus einem XML-Parser, der Socket-Schnittstelle, dem Datencontainer und der Bibliotheksverwaltung. GSM kann über XML-Steuerströme von NetSim ferngesteuert werden.

5.1.2 Einbindung der Algorithmen in OMNeT++

Nahezu parallel zu der Entwicklung von NetSim wurde eine Integration des formalen Frameworks in den weit verbreiteten Netzwerksimulator *OMNeT++* [Omn08] geprüft [Wes06]. Zunächst wurde in C++ ein Tool entwickelt, mit dem große Netzwerkstrukturen sowie Kommunikationslinien automatisch generiert werden können. So kann beispielsweise vorwiegend eine Baum- oder eine Linientopologie erzeugt werden. Bei der Generierung der Kommunikationslinien ist zwischen einer überwiegenden Master/Slave- oder Publisher/Subscriber-Infrastruktur zu wählen. Alternativ dazu können auch dezentrale IRT-Übertragungen als Vorgabe für den Zufallsgenerator der Kommunikationslinien vorgegeben werden. Die erzeugten XML-Dateien beinhalten die erzeugte Netzwerk-Infrastruktur sowie die Kommunikationslinien und können von GSM importiert werden. GSM berechnet auf dieser Basis die Schedules, welche dann wiederum als XML-Dateien vorliegen. Eine Simulation der in Kapitel 4 vorgestellten Modellierung in OMNeT++ kann im Rahmen von zukünftigen Forschungstätigkeiten durchgeführt werden.

Abbildung 5.3 zeigt die Darstellung eines einfachen Ethernet-Netzwerkes, welches automatisch generiert worden ist. Die Geräte 9 und 21 stellen dabei asynchron sendende Datenquellen dar. OMNeT++ verfügt bereits über einen vollständig implementierten TCP/IP-Protokollstapel, der in die Simulation integriert werden kann.

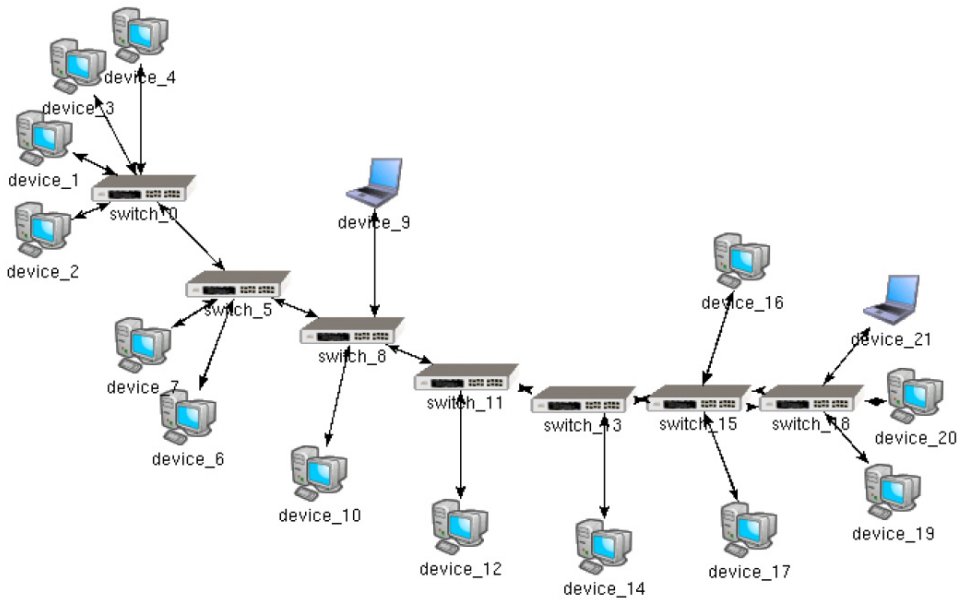


Abbildung 5.3: Netzwerkdarstellung in OMNeT++ [Wes06]

[2]Alternativ dazu ist eine Konsolenbedienung möglich.

Um nun die Netzwerkinfrastruktur, die Kommunikationslinien und die Schedules der Switches in OMNeT++ zu laden, wurde ein Konverter der XML-Dateien zu Netzwerkbeschreibungsdateien[3] für OMNeT++ entwickelt. Diese Dateien können bereits in das Simulationsframework OMNeT++ geladen werden.

Als zukünftige Weiterentwicklung ist für die Integration der Schedule eine Verhaltensbeschreibung für einen modifizierten Switch in OMNeT++ zu entwickeln. Ein solcher Switch wird in Kapitel 5.3 vorgestellt. Zusätzlich ist es wünschenswert, zentrale Schedules zur Simulation eines (Sub-)Netzwerkes auf der Basis von Hubs zu generieren. Dadurch sind Ansätze wie EPL (vgl. Kapitel 3.1.2.1) simulierbar. Ebenso ist die Simulation einer gemischten Intfastruktur aus Hubs und Switches wünschenswert, wie sie in Kapitel 4.4 vorgestellt wurde.

5.1.3 Zusammenfassung

Die algorithmische Umsetzung des formalen Frameworks bis zu Kapitel 4.2 als GSM-Bibliotheken wurde erfolgreich durchgeführt und mit exemplarischen kleinen Netzwerken getestet [Sch07b]. Die erzeugten Schedules können in die von NetSim bereitgestellten RT-Switches geladen werden. Die Simulation erlaubt die Übertragung sowohl von IRT-Frames, als auch von asynchronen Frames bis zur IP/ICMP-Protokollebene. Asynchrone Sendungen müssen im Vorfeld unter Angabe der Quelle, des Ziels und des Sendezeitpunktes definiert werden.

Da es sich um eine prototypische Implementierung handelt, bleibt an dieser Stelle Bedarf für eine Weiterentwicklung von NetSim und GSM. Insbesondere ist die Umsetzung des List-Schedulings (s. Kapitel 4.8.3) unter Berücksichtigung der Verzögerungszeiten von Interesse. Mit diesem Werkzeug können dann industrietypische Anlagenkonfigurationen simuliert und entsprechende Schedules kalkuliert werden, welche mit gängigen Technologien verglichen werden können. Ebenso kann ein Prototyp des in Kapitel 5.3 vorgestellten modifizierten Switches als Hardware-Komponente in die Simulation integriert werden. Die bisherige Verhaltensbeschreibung des RT-Switches kann zwar Schedules aufnehmen, berücksichtigt jedoch bislang keine Verzögerungszeiten. Des Weiteren ist eine Simulation der Synchronisierung der lokalen Schedules, z. B. unter Verwendung des IEEE 1588 Protokolls, im Rahmen einer Weiterentwicklung sinnvoll.

Diese Integration kann alternativ dazu in OMNeT++ erfolgen. Der Vorteil besteht hier in der Verwendung eines etablierten Simulators. Im Umfeld dieser Arbeit wurde gezeigt, dass die erstellte Software zur Kalkulation der Schedules GSM mit OMNeT++ zusammenarbeitet und OMNeT++ die von GSM berechneten Schedules als NED-Datei importieren kann.

Als Raum für weitere Forschungsarbeiten ist auch bei OMNeT++ die Integration des in Kapitel 5.3 beschriebenen modifizierten Switches zu nennen. Ebenso können zentrale Scheduler, die beispielsweise von EPL als Managing Node (vgl. Kapitel 3.1.2.1) bezeichnet werden, als Simulationskomponenten in OMNeT++ hinzugefügt werden. Die gleiche Vorgehensweise ist für ProfiNet-Switches (s. Kapitel 3.1.3.1) und SERCOS III- bzw. EtherCAT-Netzwerke (s. Kapitel 3.1.3.2 bzw. 3.1.3.3) denkbar. Auf diese Weise ist es möglich,

[3] von OMNeT++ als Network Definition Dateien (NED) bezeichnet

die bestehenden Technologien über die Simulation mit der formalen Modellierung zu verbinden.

Zur Integration von asynchronen Sendungen bietet OMNeT++ das INET-Framework, s. [Wes06]. Dadurch ist es im Gegensatz zur Weiterentwicklung von NetSim nicht notwendig, den Protokollstapel von TCP, UDP sowie der Anwendungsprotokolle selbst zu entwickeln. Das Ziel der beiden Weiterentwicklungen von OMNeT++ kann darin bestehen, eine nahtlose Integration von gängigen Lösungen zur Realisierung von echtzeitfähigem Ethernet über die formale Modellierung bis hin zu Ethernet-Netzwerken der Leitebene zu fokussieren.

5.2 Szenarien zur Umsetzung der Schedules

Wie bereits in den vorangegangenen Kapiteln erwähnt, geht das Modell von einer baumförmigen Infrastruktur des Netzwerkes aus, welches kompatibel zum Ethernet-Standard nach IEEE 802.3 sein soll. Die Einführung von Schedules ersetzt das etablierte CSMA/CD mit seiner Exponential Backoff Strategie um ein TDMA, um eine für die Echtzeitfähigkeit notwendige deterministische Datenübertragung einzuführen.

Die Modellierung basiert darauf, dass die Sendezeitpunkte der Kommunikationslinien bzw. -bäume in jedem Zyklus bekannt sind. In der Praxis kann jedoch eine Vielzahl einfacher Geräte wie Lichtschranken oder Endschalter zum Einsatz kommen. Muss jedes dieser Geräte eine eigene lokale Schedule verwalten und synchronisieren, so werden die Kosten für diese Geräte unverhältnismässig steigen. Der Einsatz von Geräten, die nicht aktiv senden können und statt dessen lediglich auf Anfragen reagieren, reduziert hingegen die Kosten der Anlage. Andererseits stellt der Poll Frame eine zusätzliche Kommunikation dar, die wiederum die Reaktionszeit des Systems erhöht.

Dabei können zwei Arten des *Pollings* unterschieden werden. Bei der ersten Art wird das Polling direkt von einer Steuerung applikationsbezogen abgesendet mit der Aufforderung an einen Sensor, einen Messwert zu erfassen und diesen dann zurückzusenden. Dabei ist zu beachten, dass die Erfassung eine gewisse, je nach Sensor variable Zeit in Anspruch nehmen kann. Diese Zeit sowie deren Schwankung ist in der Schedule als Sicherheitsabstand zu berücksichtigen.

Die Kontrolle der zweiten Art des Pollings liegt bei dem Netzwerk selbst. Hier kann beispielsweise ein Temperatursensor einen Messwert erfassen und vorhalten. Der Poll Frame veranlasst den Sensor dann lediglich zum Versenden des Messwertes. Bei dieser Art des Pollings kann der Sensor unmittelbar auf die Anfrage reagieren, so dass die Verzögerung und der Jitter der Verzögerung minimiert wird. Der Messwert selbst kann jedoch unter Umständen älter sein. In diesem Falle kann zusammen mit dem Messwert ein Zeitstempel der Erfassung in den Antwort-Frame übertragen werden.

In beiden Fällen des Pollings ist zusätzlich zu berücksichtigen, dass der Poll Frame vor dem entsprechenden Datenframe zu senden ist. Dadurch entsteht eine kausale Abhängigkeit zwischen zwei Sendungen. Soll sowohl die Anfrage, als auch die Antwort im gleichen Produktionszyklus ausgeführt werden, so muss bei Anwendung der Graphenfärbung der Anfrage ein geringerer Farbwert zugewiesen werden als der Antwort, um früher ausgeführt zu werden. Dazu müssen jedoch die Färbungsalgorithmen modifiziert werden, deren Komplexität sich erhöht. Wird das List-Scheduling (s. Kapitel 4.8.3) angewendet, welches

die IRT-Übertragungen über alle Switches betrachtet, so sind die beiden Kommunikationslinien mit entsprechendem Abstand zueinander zu vereinigen. Bei der Suche nach dem frühest möglichen Startzeitpunkt müssen dann beide Kommunikationslinien gemeinsam verschoben werden. Als Alternative dazu können die Poll- und Datenframes in verschiedenen Zyklen ausgeführt werden. In diesem Falle kann wie in Kapitel 4.7 beschrieben vorgegangen werden.

Der Ort, an dem das Polling durchgeführt wird, kann zusätzlich dazu variiert werden. Damit werden alternative Umsetzungen der erstellten Schedules ermöglicht, welche Auswirkungen die jeweiligen Lösungen besitzen. Diese Alternativen werden in diesem Kapitel diskutiert. Abbildung 5.4 zeigt eine Übersicht der vier ausgearbeiteten Varianten, die im Folgenden vorgestellt werden.

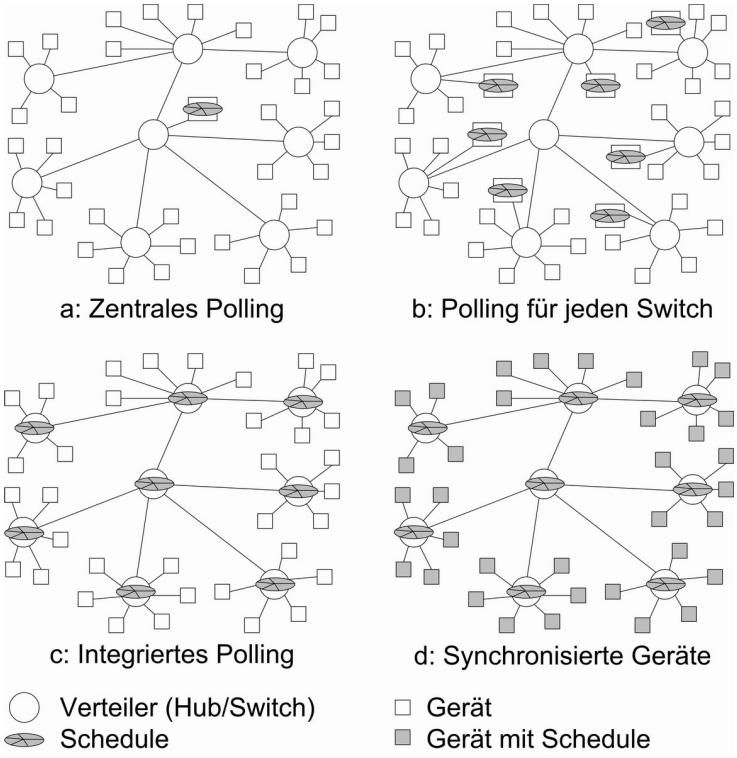


Abbildung 5.4: Vier Wege zur Umsetzung der Schedules [DW06]

5.2.1 Zentrales Polling

Abbildung 5.4a zeigt die Realisierung im Rahmen einer einzigen Schedule, die zentral von einem speziellen Gerät verwaltet wird. Dieses Gerät wird im Folgenden als Scheduler bezeichnet. Der Scheduler wird wie ein herkömmliches Gerät an einen Verteiler angeschlossen. Die gesamte Hardware des Netzwerkes - die Geräte, Kabel und Netzwerkkarten - kann aus im Handel erhältlichen COTS-Komponenten bestehen. Die Geräte dürfen jedoch keine Frames selbständig versenden. Statt dessen werden die Sendungen ausschließlich vom Scheduler bestimmt, indem dieser eine Sendeberechtigung - ähnlich eines Tokens - an die Geräte übergibt. Im Anschluss daran können die Sender jeweils einen Ethernet-Frame von

fester Größe an einen beliebigen Empfänger oder als Broadcast an alle Geräte absetzen. Eine Synchronisierung kann wie bei Ethernet PowerLink über einen Broadcast-Frame des Schedulers bei Zyklusbeginn erfolgen.

Da alle Geräte beim zentralen Polling dem Scheduler folgen, ist das Hinzufügen asynchron sendender Geräte, beispielsweise eines Laptops ohne zusätzlichen Echtzeit-Treiber, nicht gestattet. Eine asynchrone Sendung würde mit IRT-Frames kollidieren und damit die Echtzeitfähigkeit des Netzwerkes zerstören. Es handelt sich also hier um geschlossene echtzeitfähige Subnetze. In der Praxis lässt sich das zentrale Polling realisieren, indem man die entsprechenden Verteiler fest in die Schaltschränke einbaut und keine offenen, nach außen zugängliche Ports gestattet.

5.2.1.1 Verwendung von Hubs

Um die Verzögerungszeit zwischen dem Absenden der Sendeberechtigung und deren Empfang sowie die Verzögerung der Datensendung zu minimieren, empfiehlt sich die Verwendung von Hubs. Zur Eliminierung des nicht-deterministischen Verhaltens ist bei der Verwendung von Hubs nur eine Sendung zu einem Zeitpunkt möglich. Die Bandbreite des Netzes wird dadurch nicht effizient genutzt.

Die Erstellung einer Schedule für diesen Fall ist trivial, da alle Kommunikationsanforderungen einfach hintereinander geschaltet werden können. Aus der Summe der benötigten Übertragungszeiten ergibt sich dann die Zykluszeit. Der Aufbau von Konfliktgraphen, deren Färbung und der Aufbau von synchronen Schedules entfällt also bei dieser Art der Umsetzung. 5.5a basiert auf dem Beispiel in Abbildung 4.1, fügt einen Scheduler an einen freien Port ein und geht davon aus, dass es sich bei den Verteilern um Hubs handelt. Abbildung 5.5b zeigt die auf Abbildung 4.15 basierende Schedule, die durch den Scheduler verwaltet wird.

Mit dem Synchronisationsframe oder auch als separate Sendung kann der Scheduler einem einzelnen Gerät oder einer Gruppe von Geräten in einer vorkonfigurierten Folge asynchrone Sendungen gewähren. Diese Sendungen werden im Anschluss an die isochrone Phase gestartet und müssen vor dem Start der nächsten isochronen Phase beendet sein. Die zur Verfügung gestellte Zeit für asynchrone Übertragungen muss im Vorfeld konfiguriert werden.

Fordert der Scheduler ein Gerät aus seiner unmittelbaren Nachbarschaft zum Senden auf, so empfängt dieses Gerät den Poll Frame früher als ein Gerät in großer Entfernung zum Scheduler. Die Position des Schedulers kann zur Optimierung der Zykluszeit beitragen, indem er in die Nähe der Geräte mit IRT-Echtzeitanforderungen plaziert wird. Um die Differenzen der Übertragungszeiten zu beschränken, kann auch eine maximale Grösse des Subnetzes vorgesehen werden. Bei Ethernet PowerLink dürfen beispielsweise nicht mehr als zehn Hubs kaskadiert werden.

Ethernet PowerLink und verwandte Ansätze stellen eine weit verbreitete Implementierung dieser Art der Umsetzung von Schedules dar. Die Ideen von EPL und ähnliche Ansätze wie TCnet und EPA sind also als Teillösungen innerhalb der Modellierung des in dieser Arbeit erstellten Frameworks anzusehen. Bei EPL im bislang realisierten Protected Mode ist eine Verwendung von Geräten, welche nicht dem Protokoll des Schedulers gehorchen, verboten. Denn solche Geräte würden durch ihren CSMA/CD-Algorithmus die

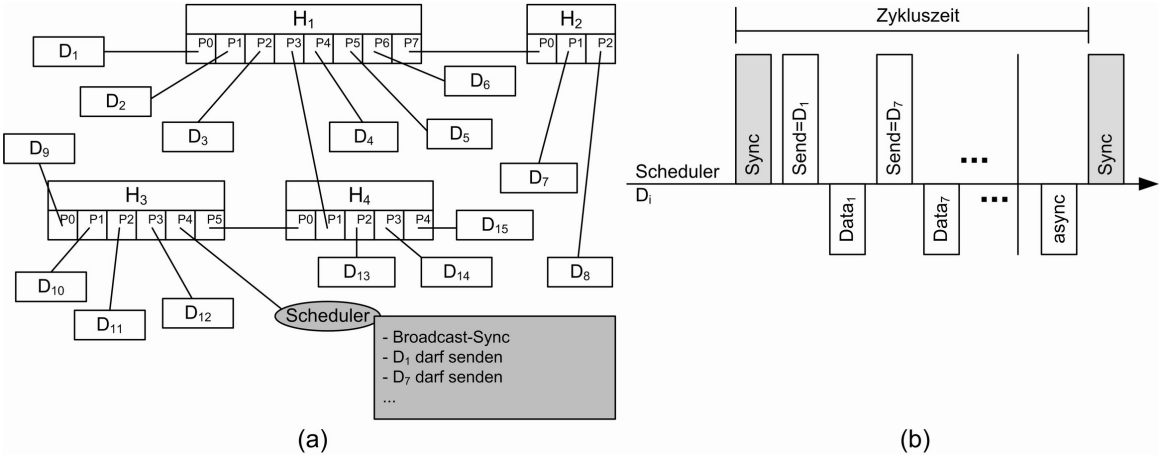


Abbildung 5.5: Zentraler Scheduler - Netzwerkinfrastruktur (a) und Schedule (b)

in Abbildung 5.4b dargestellte Schedule unterwandern. Der Scheduler wird bei EPL als Managing Node bezeichnet.

Dieser Ansatz der Umsetzung der Schedules erlaubt also lediglich echtzeitfähige Geräte im Netzwerk und führt damit zur Bildung von Subnetzen, bei denen der Scheduler als Manager fungiert. Wie bei EPL kann ein Subnetz auch hier über den Scheduler mit einem äußeren, nicht-deterministisch agierenden Netz verbunden werden. Der Scheduler besitzt dann die Funktionalität eines Routers bzw. einer Bridge. Gleichzeitig stellt diese Umsetzung einen Master/Slave-Ansatz dar, der bereits weit verbreitet ist.

EPL ermöglicht die Übertragung von asynchronen Daten in einer separaten Phase. Die Unterteilung in eine isochrone und asynchrone Phase stellt bei dieser Form der Realisierung die einzige Möglichkeit zur Übertragung von nicht-echtzeitfähigen Daten dar. Ein Problem stellt die Framegröße in der asynchronen Phase dar, welches in Kapitel 5.2.6 beschrieben wird.

Die Größe der Datenframes kann in beiden Phasen beliebig sein, da eine feste Unterteilung zur leichteren Berechnung von Schedules entfällt. Große Frames erhöhen jedoch die Zykluszeit. Zur Einhaltung der Kompatibilität sollten die Größenbeschränkung eines Ethernet-Frames eingehalten werden. Aus Sicht der Automatisierungstechnik wird man hier die minimale Framegröße wählen.

5.2.1.2 Verwendung von Switches

Um parallelen Datenverkehr zuzulassen und damit die Bandbreite des Netzes besser auszunutzen, können statt Hubs Standard-Switches eingesetzt werden. Die Poll Frames des Schedulers werden weiterhin als Broadcast versendet und können dadurch mehrere Geräte gleichzeitig zur Sendung auffordern. Die Poll Frames verzögern sich jedoch stärker als im vorherigen Fall aufgrund der erhöhten Durchlaufzeit der Switches.

Die Broadcastsendung kann im Daten-Teil des Frames eine Liste mit MAC-Adressen der Geräte enthalten, die im folgenden Zeitschlitz gleichzeitig senden dürfen. Um die Broadcastsendung zu verkürzen, ist statt der $6Byte$ großen vollständigen MAC-Adresse eine Knoten-ID wie bei EPL vorzuziehen. Diese ID kann in der Konfigurationsphase des Netzes vergeben werden.

Während bei der Verwendung von Hubs der jeweilige Sender in seinem Zeitslot zu jedem anderen Gerät des Subnetzes kommunizieren kann, müssen in diesem Fall auch die jeweiligen Empfänger der Übertragungen im Vorfeld feststehen, damit eine gleichzeitige konfliktfreie Abarbeitung der Übertragungen erfolgen kann. Es können dann alle Geräte gleichzeitig im Vollduplexmodus senden, die sich in der Abbildung 4.25 dargestellten globalen Schedule in einem Zeitslot befinden und damit parallel ausführbar sind.

Abbildung 5.6a zeigt das vorherige Beispiel der Netzwerkinfrastruktur unter Verwendung von Switches. Die Schedule besteht aus den Sendeanforderungen und den parallel ausführbaren IRT-Übertragungen (5.6b).

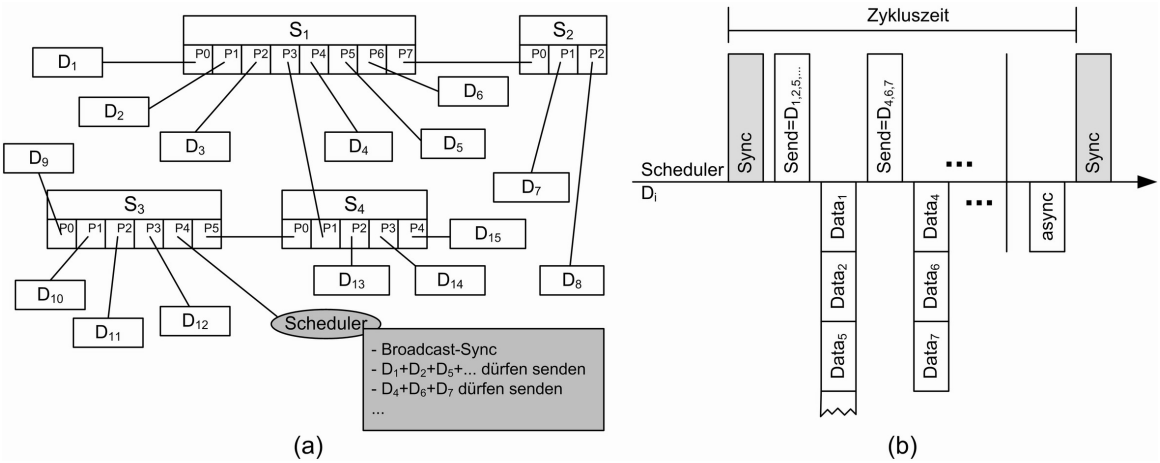


Abbildung 5.6: Zentraler Scheduler und Switches

Der entscheidende Nachteil beim Einsatz von COTS-Switches liegt in der erhöhten Verzögerungszeit. Store-and-forward Switches kommen im harten Echtzeitumfeld nicht in Betracht. Bei cut-through Switches, welche eintreffende Frames lediglich bis zur MAC-Adresse des Ziels auslesen, sind die Verzögerungszeiten in der Praxis noch annehmbar.

Im Gegensatz zu Hubs können sich gerade low-cost Switches in ihrem Verhalten bei der Weiterleitung von Frames und dem Versenden von Broadcasts stärker unterscheiden. Für die Erfüllung von IRT-Echtzeitanforderungen muss gewährleistet sein, dass die Switches ihre Weiterleitungstabellen bezüglich der RT-Geräte nach der Hochlaufphase nicht verwerfen. Des Weiteren sind die Intervalle zur Aktualisierung der Weiterleitungstabelle nicht genormt. Eine gesendete Broadcast-Meldung nach dem Löschen existierender Einträge in der Tabelle zerstört in der isochronen Phase die Echtzeitfähigkeit des Netzes. Konfigurierbare Switches hingegen verfügen meist über die Möglichkeit der Parametrierung einer statischen Weiterleitungstabelle.

Auch beim Einsatz des STP-Protokolls muss darauf geachtet werden, dass die Aktualisierung des Spanning Trees nicht in einen Echtzeit-Zyklus fällt. Denn auch dadurch kann die Weiterleitung von echtzeitkritischen Frames für einen gewissen Zeitraum nicht gewährleistet werden, so dass die gesamte Anlage betroffen ist. Generell existieren in dem vorgegebenen Netz keine alternativen Routen, da es sich bereits um eine baumförmige Infrastruktur handelt. Um die Echtzeitfähigkeit nicht zu verlieren, darf ein STP-Algorithmus allenfalls in der Konfigurationsphase ausgeführt werden und ist im laufenden Betrieb

ebenso zu verhindern wie das Hot-Potato Verfahren. Solche Verfahren müssen also für den laufenden Betrieb in den Switches abschaltbar sein.

Bei einem Test gängiger Switches [Sch07b] ist aufgefallen, dass diese bei einer hohen Auslastung Frames verwerfen. In den Tests konnte diese Auslastung jedoch erst durch mehrfache Socket-Verbindungen jedes angeschlossenen Gerätes zu vielen anderen Geräten mit hoher Datenrate erreicht werden. Generell muss ein Switch mit n Ports im Vollduplexbetrieb nicht zwingend n gleichzeitige Verbindungen zulassen, selbst wenn diese Übertragungen keinen Konflikt auslösen. Eine volle Ausnutzung der theoretischen Bandbreite kann also nicht vorausgesetzt werden. Beide Arten von Switches können jedoch als Modelle in die in Kapitel 5.1 vorgestellte Simulation integriert werden.

Ein weiterer Nachteil gegenüber der Verwendung von Hubs liegt darin, dass der Scheduler keine Informationen über den Status des gerade versendeten Datenframes erhält, da er selbst lediglich bei Broadcastsendungen als Empfänger fungiert. Dadurch ist der Scheduler nicht in der Lage, auf zu lange Frames oder auf Frames, die irregulär gesendet wurden, zu reagieren. Irreguläre Sendungen können beispielsweise durch Geräte entstehen, die keine Kenntnis von dem Echtzeitprotokoll besitzen. Der Scheduler muss sich demnach auf die korrekte Umsetzung seiner Schedule durch die Geräte verlassen können. Versendet man statt dessen alle Frames als Broadcast, so kann sich jedoch wiederum nur eine Sendung auf dem Netz befinden mit dem Nachteil der erhöhten Verzögerung von Switches. Durch die Nachteile ist es unwahrscheinlich, dass ein zentraler Scheduler in Kombination mit COTS-Switches eine Lösung für IRT-Echtzeitanforderungen darstellt. Solche Lösungen können jedoch bei weichen Echtzeit-Bedingungen in Frage kommen, wenn zusätzlich die von den Switches unterstützte VLAN-Priorisierung zum Einsatz kommt. Damit ist zumindest die bevorzugte Behandlung von Frames mit höheren Echtzeitanforderungen gegenüber asynchronen Sendungen gewährleistet. Ist der Anteil der hochprioren Frames gering (vgl. Kapitel 2.2.1.4), so stehen bereits effiziente Lösungen wie Ethernet/IP zur Verfügung.

Die Vorteile der zentralen Schedule liegen generell in der leichten Berechenbarkeit, dem Einsatz von Standard-Komponenten und der hohen Synchronisation des Netzes durch die Broadcastsendungen des Schedulers. Generell problematisch ist hingegen die Ausführung von asynchronen Übertragungen. Diese dürfen generell nur dann erfolgen, wenn der Scheduler ein Gerät dazu berechtigt. Asynchrone Übertragungen werden dabei in den etablierten Standards in eine eigene, asynchrone Phase ausgelagert. In diesem Fall muss sichergestellt werden, dass keine asynchronen Sendungen mehr zu Beginn der nächsten isochronen Phase ausgeführt werden.

5.2.2 Polling für jeden Switch

Abbildung 5.4b zeigt die Verwendung eines Schedulers für jeden Switch. Der Einsatz von Hubs in Kombination mit einer dezentralen Schedule ist nicht sinnvoll, da sie nur eine gleichzeitige Kommunikation über das gesamte Netzwerk zulassen.

Jeder Scheduler besitzt in diesem Fall lediglich die lokale Schedule für die sendenden Geräte an seinem Switch. Auf diese Weise werden die Verzögerungen der Poll Frames im Netzwerk vereinheitlicht, während die zeitliche Verzögerung des Empfangs der Poll Frames

im vorherigen Kapitel von der Nähe des Schedulers zu den Geräten abhängt. Durch die Verkürzung des Pollings kann die Zykluszeit verringert werden.

Dezentrale Scheduler in Verbindung mit Standard-Switches bringen jedoch eine Reihe von Problemen mit sich. Zunächst ist eine höhere Anzahl von Zusatzgeräten mit höheren Kosten und Verkabelungsaufwand notwendig. Zusätzlich belegen diese Geräte jeweils einen freien Port eines Switches und damit Kapazitäten des Netzwerkes.

Außerdem müssen die Scheduler untereinander synchronisiert werden, damit ihre Schedules nicht auseinander driften und Switch-übergreifender Verkehr nicht zu Konflikten durch Pufferung von Frames führt. Die Synchronisation kann jedoch durch regelmäßige Broadcast-Meldungen realisiert werden. Zusätzlich kann ein präziser Uhrenabgleich hardwareseitig durch einen in die Scheduler integrierten IEEE 1588-Synchronisationsbaustein erfolgen, der bereits bei existierenden Lösungen zur Anwendung kommt.

Problematischer ist hingegen die Ansteuerung der Geräte durch den jeweiligen Scheduler. Broadcastsendungen zum Pollen der Sender an einem Switch können nicht durchgeführt werden, da sich diese auch auf die anderen Switches auswirken. Unicastsendungen des Poll Signals hätten zur Folge, dass an einem Switch zu einem Zeitpunkt nur genau ein Gerät aktiv zum Senden aufgefordert werden kann. Dadurch wird der Vorteil eines Switches, gleichzeitg mehrere Sendungen parallel zu verarbeiten, unwirksam. Die einzige Lösung besteht darin, alle an einem Switch angeschlossenen Geräte zu einer Ethernet-Multicast Gruppe zusammenzufügen. Der zuständige Scheduler für diesen Switch kann dann für jeden Zeitslot einen Poll Frame für diese Geräte versenden. In Kapitel 2.2.3.2 wurde beschrieben, dass Switches zur Interpretation von Ethernet-Multicastadressen das Verfahren des IGMP-Snoopings verwenden. Dabei handelt es sich jedoch um Schicht-3-Switches, welche eine detailliertere Interpretation der Frames vornehmen und daher sowohl die Verzögerungszeit erhöhen, als auch kostenintensiver sind. Eine ausschließliche Interpretation von Ethernet-Multicastsendungen auf der OSI-Schicht 2 ist mit herkömmlichen Switches kaum durchführbar.

Des Weiteren bleiben die in Kapitel 2.2.4.2 diskutierten Nachteile der Standard-Switches, insbesondere die erhöhte Verzögerungszeit gegenüber Hubs, bestehen. Geräte, die nicht echtzeitfähig sind und nicht dem Protokoll des Schedulers folgen, zerstören auch in diesem Ansatz die Echtzeitfähigkeit des Netzwerkes.

Das Polling der Switches stellt jedoch ein Problem in der Modellierung dar. Zunächst kann ein Poll Frame als eine eigene Kommunikationslinie betrachtet werden, die von einem Vertreiler unmittelbar vor einem Blatt des Baumes zu diesem Blatt verläuft und dabei keine weiteren Verteiler überquert. Eine solche Kommunikation führt bei der Betrachtung des globalen Knotenkonfliktgraphen zu einer Lösung, in welcher der Poll Frame eine eigene Kommunikations-ID erhält. Das Problem liegt darin, dass der folgende Datenframe des passiven Gerätes zu dem Ziel der Poll Anfrage kausal abhängig von dem Poll Frame ist, da sich dieser zweite Frame unmittelbar nach dem Polling in der Schedule befinden muss, also im unmittelbar folgenden Zeitslot.

Eine Möglichkeit zur Lösung dieses Problems besteht darin, zunächst die Poll Frames zu ignorieren und statt dessen nur die Datenframes zu betrachten und zu schedulen. Im Anschluss daran kann vor jeden Zeitslot ein zusätzlicher Zeitslot für das Polling plaziert werden. Dieser Slot muss keine Verzögerungszeiten über mehrere Switches berücksichtigen und kann daher entsprechend kurz sein. Im Anschluss daran kann die Berücksichtigung

der gesamten Verzögerungen in der Schedule erfolgen. Dadurch können die entsprechenden Pollings zum jeweils frühest möglichen Startzeitpunkt vorverlegt werden.

Ein weiterer Ansatz, der die Verzögerungszeiten einbezieht, besteht in der Verwendung des List-Schedulings, s. Kapitel 4.8.3. Innerhalb des globalen Knotenkonfliktgraphen kann der Poll Frame separat von dem Antwort-Frame betrachtet werden, da es sich um zwei getrennte Übertragungen handelt. Dementsprechend können die jeweiligen Konflikte als Kanten im Graphen modelliert werden. Durch die Reaktionszeit des betreffenden Gerätes ist die Zeit zwischen dem Empfang des Poll Frames und dem Absenden der Antwort konstant. In der Datenstruktur der List-Schedule können diese beiden Frames daher als eine Struktur angesehen werden, die nur als Gesamtheit zeitlich verschoben werden kann. Bei einer Verschiebung der Struktur ist dabei an jedem Switch eine Konfliktprüfung - sowohl des Poll Frames, als auch der Antwort - anhand des Knotenkonfliktgraphen vorzunehmen.

5.2.3 Integriertes Polling

Die einzige Möglichkeit, im laufenden Betrieb nicht-echtzeitfähige Geräte hinzuzuschalten und die bislang beschriebenen Nachteile der Standard-Switches aufzuheben, besteht in der Integration der Schedules in die Switches selbst, siehe Abbildung 5.4c und d. Ein solcher modifizierter Switch muss so konfiguriert werden können, dass die Adresstabellen der angeschlossenen Geräte mit Echtzeitanforderungen statisch sind. Zusätzlich können Datenframes von Protokollen unterbunden werden, die den laufenden Echtzeitbetrieb beeinflussen. Spanning-Tree Algorithmen im laufenen Betrieb der Anlage können ebenso ausgeschlossen werden wie andere versendete Broadcast-Meldungen. Diese asynchronen Frames können zwischengespeichert und in zuvor definierten freien Zeiträumen des jeweiligen Ausgangsports weitergeleitet werden.

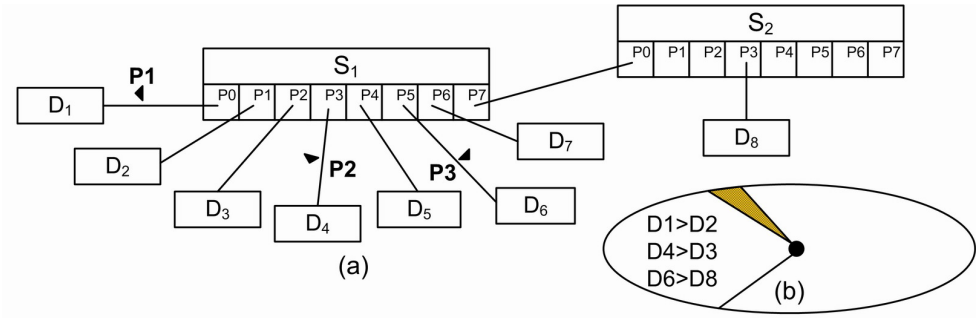
Um dies zu realisieren, muss die lokale Schedule - beim Start der Anlage - in den Switch geladen werden können. Im Anschluss daran müssen die lokalen Schedules der Switches synchronisiert werden. Eine grundlegende Synchronisation kann, wie bereits in etablierten Verfahren, in der Hochlaufphase erfolgen. Im Anschluss daran kann die Synchronisation über einen zyklisch versendeten Broadcast-Frame erfolgen. Da die Switching-Hardware und die Verdrahtung der Switches untereinander konstant bleibt und Konflikte anhand der Schedules aufgehoben werden, ist eine sehr präzise Synchronisation anzunehmen.

In der Konfiguration kann auch festgehalten werden, ob an dem jeweiligen Port ein anderer Switch oder ein Endgerät angeschlossen ist. Im zweiten Fall kann ein Switch anhand seiner Schedule auch selbständig die direkt angeschlossenen Endgeräte pollen, die über keine eigene Schedule verfügen und damit nicht aktiv senden können. In dem Poll Frame ist die Zieladresse im Datenteil enthalten, an die das Endgerät seine Daten versendet.

Unmittelbar nach dem Absetzen des Poll Frames kann der Switch den entsprechenden Quell- und Zielport neu verschalten, so dass der Datenframe des Gerätes unmittelbar weitergeleitet werden kann, siehe Kapitel 5.3.

Ebenso ist vorstellbar, dass ein solcher Switch zeitgleich mehrere Poll Frames an verschiedene Ports sendet. Dies ist dann sinnvoll, wenn entsprechend der Schedule die folgenden Sendungen der Geräte unabhängig voneinander sind.

Abbildung 5.7a zeigt zunächst das Absetzen der Sendeanfragen eines Switches an seine angeschlossenen Geräte entsprechend seiner Schedule. Die Zeit für das Absetzen der Anfrage im Rahmen eines minimal großen Ethernet-Frames bis zur Interpretation des Frames durch die jeweiligen Geräte und schließlich der Antwort muss in der Schedule berücksichtigt sein, wie es im Teil b der Abbildung skizziert wird. Diese Zeit ist jedoch im gesamten Netzwerk relativ konstant, da die Poll-Frames nicht über mehrere Switches verlaufen. Variabel sind die Kabellängen sowie die Antwortzeiten der Geräte. Abbildung 5.7c zeigt eine Liste der Poll-Frames, die von Switch 1 an die betroffenen Geräte abgesetzt werden.



Sende-Anfrage (Poll)	Quell-MAC	Ziel-MAC	Daten
P1	S1	D1	sende an D2
P2	S1	D4	sende an D3
P3	S1	D6	sende an D8

(c)

Abbildung 5.7: Sendeanfragen eines Switches

Sobald die Anfragen gesendet wurden und gerade von den Geräten interpretiert werden, kann der Switch mit der entsprechenden Umschaltung seiner Ports beginnen. Im Beispiel wird der Eingang von Port 0 mit dem Ausgang von Port 1, der Eingang von Port 3 mit dem Ausgang von Port 2 sowie der Eingang von Port 5 mit dem Ausgang von Port 7 verschaltet. Dies ist in Abbildung 5.8 skizziert.

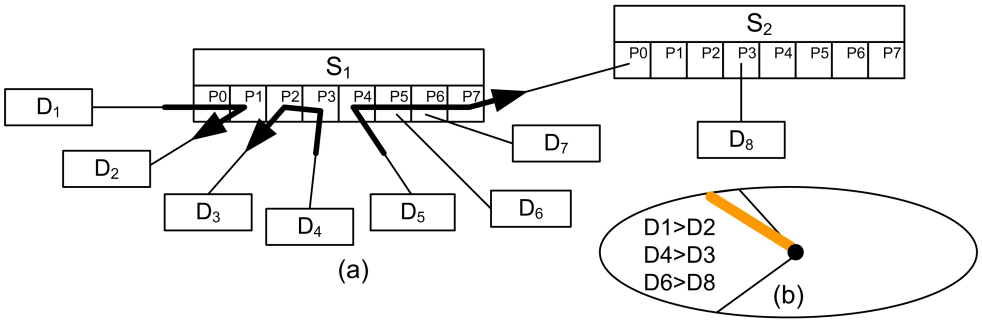


Abbildung 5.8: Durchschalten der Ports

Die Ports müssen durchgeschaltet sein, wenn die Sendung der einzelnen Geräte beginnt. Diese Sendungen können dann mit minimaler Verzögerung zu den Zieladressen weitergeleitet werden, die Frames müssen von den Switches aufgrund der Schedule nicht mehr

interpretiert werden. Die berechnete Zeit der Weiterleitung ist in der Schedule festgehalten und in Abbildung 5.9 dargestellt. Für diese Zeit bleiben die entsprechenden Ports durchgeschaltet.

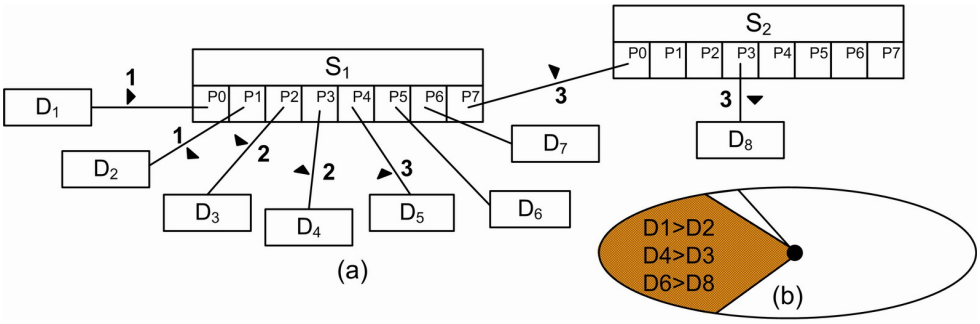


Abbildung 5.9: Weiterleitung der Daten-Frames

Der entscheidende Vorteil dieser Lösung ist, dass der Echtzeit-Frame bei diesem Ansatz in keiner Weise mehr analysiert werden muss, da die Ziel-Adresse dem Switch durch seine Schedule bereits bekannt ist. Dadurch sind ähnliche Verzögerungszeiten wie bei einem Hub zu erwarten mit dem weiteren Vorteil, dass dieser Switch das Paket ausschließlich an den richtigen Ausgangsport weiter sendet und die Weiterleitung der Frames an jedem Hop kontrollierbar ist. Es wird also paralleler Echtzeit-Verkehr ohne Auftreten von Konflikten ermöglicht, wobei asynchrone Frames separat behandelt werden können (s. Kapitel 5.2.6).

Zusätzlich ist es möglich, Zustände einzelner Sensoren und Aktoren asynchron auszulesen. Mit einem Laptop kann beispielsweise ein asynchroner Poll Frame abgesetzt werden, der in freien Zeitslots zum betreffenden Sensor oder Aktor gelangt. Dies muss jedoch in genügend großem Abstand zu regulären Übertragungen durch die Echtzeit-Schedule erfolgen, so dass diese nicht beeinträchtigt wird. Die Kontrolle der Weiterleitung von asynchronen Anfragen kann in den modifizierten Switches integriert werden, welche diese Anfragen ggf. in einer eigenen Warteschlange zwischenspeichern und nur dann zum nächten Hop weiterleiten, wenn keine Konflikte mit der Echtzeitschedule zu erwarten sind. Der Sensor bzw. Aktor antwortet dann ebenso asynchron - ggf. mit einem integrierten Zeitstempel im Datenframe. Der Frame wird im ersten Switch zwischengespeichert und in freien Zeitslots bis zu diesem Laptop weiter gereicht.

Jedoch sind auch in dieser Lösung die Nachteile aufzuzeigen. Statt preisgünstige Standard-Switches zu verwenden, wird die Anlage von neu zu entwickelnden Switches durchzogen. Auch wenn dadurch der Scheduler eingespart wird, ist davon auszugehen, dass diese Lösung kostenintensiver ist.

Im Gegensatz zu den geringen Vorteilen der dezentralen Scheduler im Vergleich zu einem zentralen Scheduler kann auf der Basis dieses Ansatzes jedoch ein sehr effizientes echtzeitfähiges Netzwerk aufgebaut werden:

- Es werden ausschließlich Ethernet-kompatible Frames versendet und die Verdrahtung bleibt ebenfalls standardkonform.
- Die Switches weisen eine ähnliche Performance wie Hubs auf.
- Die modifizierten Switches basieren zu einem Großteil auf Standard-Hardware.
- An jedem freien Port können nicht-echtzeitfähige Geräte angeschlossen werden, welche ihre Daten transparent wie durch ein herkömmliches Ethernet versenden.
- Daten von Sensoren und Aktoren können auch asynchron abgefragt werden.

Die Idee der Integration einer Schedule in den Switches ähnelt dem ProfiNet-Ansatz. Deren Switches führen jedoch kein aktives Pollen der angeschlossenen Geräte durch, sondern setzen aktiv sendende synchronisierte Geräte voraus, wie im folgenden Kaptitel 5.2.4 beschrieben wird. Zusätzlich dazu sehen ProfiNet-Switches eigene Phasen für asynchrone Übertragungen vor. In Kapitel 4.5 wurde jedoch herausgearbeitet, dass je nach Struktur des Netzes und dem Verlauf der Übertragungen große Bereiche der IRT-Phase ungenutzt bleiben. Diese Bereiche können für Hop-zu-Hop Weiterleitung von asynchronen Übertragungen verwendet werden.

5.2.4 Synchronisierte Geräte

Alternativ zu einem Polling können Geräte auch aktiv senden, sofern sie eine lokale Schedule besitzen, die mit der Schedule des Netzes synchronisiert ist. Der Vorteil liegt darin, dass auf diese Weise die Poll Frames entfallen und dadurch die resultierende Zykluszeit verkürzt werden kann. Der Switch sendet dabei keine eigenen Frames und schaltet zyklisch seine Ports entsprechend der Schedule. Die Ports, welche zum aktuellen Zeitpunkt aufgrund der Schedule nicht mit echtzeitkritschen Daten belegt sind, können asynchrone Frames bearbeiten (s. Kapitel 5.2.6). Ein möglicher Aufbau eines solchen Switches sowie eine Analyse der Realisierbarkeit der Port-Umschaltung erfolgt in Kapitel 5.3. Die Integration der Schedule und der Synchronisation erfordert jedoch intelligentere und damit kostenintensivere Geräte. Dies wirkt sich insbesondere bei einer großen Anzahl an Sensoren und Aktoren aus.

Das Polling und das selbständige Senden von Daten lässt sich kombinieren. Da die Switches in der Konfigurationsphase die lokalen Schedules erhalten, kann in diesem Schritt auch die Konfiguration der angeschlossenen Geräte am Switch erfolgen. Geräte mit höchsten Echtzeitanforderungen, beispielsweise Antriebe, könnten als aktiv sendende Geräte markiert werden, bei denen der Switch lediglich die Umschaltung der Ports durchführt. Andere Geräte mit etwas geringeren Anforderungen, die zumeist in größerer Zahl in einer Anlage vorhanden sind, können von ihren Switches gepollt werden. Auf diese Weise lässt sich Kostenersparnis mit hohen Echtzeitanforderungen kombinieren.

5.2.5 Gegenüberstellung der Ansätze

Die in Kapitel 4 beschriebene Modellierung lässt sich unmittelbar auf die in Kapitel 5.2.4 beschriebene Umsetzung der Schedules in den Switches mit synchronisierten Geräten an-

wenden. IRT-Frames werden zyklisch und selbständig von den Geräten versendet. In Kapitel 4.8 wurde auch beschrieben, in welcher Weise die Verzögerungszeiten kalkuliert werden können. Dazu ist zu Beginn jedes Zyklus eine Zeit für die Synchronisation der Geräte zu addieren. Ebenso ist eine Zeit für die Varianz der Verzögerungszeit als Sicherheit zu der Schedule hinzuzufügen. Durch den Einsatz von modifizierten Switches, die beispielsweise über eine hardwareunterstützte Synchronisation nach IEEE 1588 verfügen, sind genaue Abschätzungen der Verzögerungen und Jitter zu erwarten. Diese Art der Umsetzung entspricht Siemens ProfiNet IRT.

Um passive Geräte, die nur auf Anfrage senden, einbeziehen zu können, wurde die Möglichkeit des Pollens diskutiert. Erfolgt ein ausschließliches Polling unter Verwendung von Standard-Hubs und/oder Switches, so können Geräte ohne Kenntnis des Polling-Protokolls nicht zu dem Netzwerk hinzugefügt werden, da deren asynchrone Sendungen die Echtzeitfähigkeit beeinflussen würden. Das Polling kann direkt von dem Switch durchgeführt werden, an dem das jeweilige Gerät angeschlossen ist. Eine solche Umsetzung wird bislang in der Praxis ebensowenig angewendet wie eine Kombination aus Polling der Switches und aktivem Senden der Geräte.

Die zweite Gruppe der Lösungen, welche das Framework umfasst (s. Kapitel 4.4), sind zentrale Schedules unter Verwendung von Hubs. Diese geschlossenen Subnetze verkürzen im Gegensatz zu Switches die Verzögerungszeit, lassen jedoch keine parallelen Übertragungen zu. Solche Subnetze eignen sich für Master/Slave-Strukturen der Automatisierungstechnik, die über ein Gateway von außerhalb des Netzes angesprochen werden können. Switches hingegen eignen sich für dezentrale Anlagen mit möglichst gleichverteilter Kommunikation.

Bei der Modellierung des Frameworks hat sich eine weitere Klasse von Lösungen ergeben, die aus modifizierten Switches als Übergang zum herkömmlichen Ethernet sowie aus geschlossenen Teilnetzen mit Hubs besteht, s. Kapitel 4.4. Die geschlossenen Teilnetze zeichnen sich aufgrund der Hubs durch eine verringerte Verzögerungszeit im Gegensatz zu Switches aus und ist daher bei der Verschaltung von Antrieben zu empfehlen. Gleichzeitig können durch die Broadcastsendungen mehrere Slave-Achsen aktualisiert werden. Diese Teilnetze sind dann im Schaltschrank der Anlage fest zu verdrahten, so dass keine externen Geräte angeschlossen werden können. Damit wird der Nachteil von Hubs, dass asynchrone Sendungen den IRT-Betrieb der laufenden Anlage gefährden, vermieden. Über das Netzwerk mit modifizierten Switches (vgl. Kapitel 3.1.3.1 und 5.3) können dann sowohl IRT-Frames, als auch asynchrone Frames übertragen werden. Die zur Verfügung stehende Bandbreite ist entsprechend der Echtzeitanforderungen der Anlage aufzuteilen. Somit wird ein nahtloser Übergang zu den Standard-Ethernet Netzwerken auf den Leitebenen geschaffen.

Rundsendeframes mit einer zentralen Schedule im Master - siehe SERCOS oder EtherCAT - können in der Modellierung nur bedingt betrachtet werden, da ihre Ringstruktur nicht nachgebildet werden kann. Dazu ist eine Erweiterung der Modellierung notwendig. Es ist jedoch möglich, lediglich die ein- und ausgehenden Frames aus diesen geschlossenen Netzen zu betrachten und die Verzögerungszeiten anzupassen. Dadurch werden SERCOS- oder EtherCAT-Subnetze wie einzelne Geräte behandelt, wie es bei der gemeinsamen Verwendung von Hubs und Switches vorgestellt wurde.

5.2.6 Einbindung von asynchronen Daten

Eine weitere Frage besteht darin, welche technischen Möglichkeiten zum Hinzufügen von asynchron versendeten Datenframes bestehen. In den meisten Fällen wird eine einheitliche Framegröße für den Echtzeitbetrieb verwendet, die zusätzlich sehr klein ist bzw. der minimalen Ethernet-Framelänge entspricht.

Reserviert man hier die Zeit für die Übertragung eines maximalen Ethernet-Frames, so wird die Zykluszeit erheblich verlängert. Andererseits ist ansonsten eine Übertragung, welche dem Ethernet-Standard entspricht, nicht möglich. Werden ausschließlich Hubs mit einem zentralen Scheduler verwendet, so müssen die asynchron sendenden Geräte dem Scheduler gehorchen. Dazu kann eine Zeit innerhalb der Schedule eingeplant werden, in der asynchrone Daten übertragen werden können. Wie bei EPL kann der Scheduler dazu einzelnen Geräten in jedem Zyklus eine Berechtigung erteilen. Eim Problem besteht in der maximalen Länge eines Ethernet-Frames, der die Echtzeit-Schedule stark ausdehnen würde. Um dieses Problem zu vermeiden, lässt sich die MTU des Netzwerkes herabsetzen, so dass übergelagerte Protokolle wie IP die Frames kleiner segmentieren. Durch die verstärkte Fragmentierung und Reassemblierung der Nutzdaten asynchroner Sendungen erhöht sich jedoch der übertragene Overhead. Anhand der MTU kann man also zwischen asynchronem Datendurchsatz und benötigter Zykluszeit abwägen.

Die gleiche Problemstellung ergibt sich bei der Verwendung von modifizierten Switches. Der notwendige Aufbau eines solchen Switches wird in Kapitel 5.3.5 diskutiert. In dieser Lösung können nicht-echtzeitfähige Geräte im laufenden Betrieb der Anlage hinzugefügt werden, die keine Kenntnis von der Echtzeitfähigkeit des Netzwerkes besitzen müssen. Der Grund liegt darin, dass asynchrone Frames unabhängig von den IRT-Frames in jedem Switch verwaltet werden können. Hier wurde in Kapitel 4.6 ein Verfahren vorgestellt, in dem man mehrere Slotgrößen zulassen kann. Bei der Verwendung des List-Scheduling sind sogar beliebig große Frames planbar. Zusätzlich dazu wurde gezeigt, dass unter Umständen ein großer Anteil der Bandbreite im isochronen Echtzeitbetrieb ungenutzt bleibt. Diese in der Schedule oft fragmentierten Anteile lassen sich durch Vertauschung der Zeitslots aneinander fügen, so dass möglichst große freie Zeitbereiche der einzelnen Ports entstehen. Dadurch können größere asynchrone Frames Hop-by-Hop übertragen werden. Die Bildung eines reinen Echtzeit-Subnetzes wird also aufgehoben.

Bei jedem Port eines modifizierten Switches kann es sich handeln um

1. einen IRT-Port, falls ein IRT-Gerät angeschlossen ist,

2. einen nRT-Port, falls ein herkömmlicher Ethernet-Switch oder ein asynchron sendendes Gerät angeschlossen ist oder bei Anlagenstart kein Gerät angeschlossen ist, oder

3. einen gemischt betriebenen Port, falls ein weiterer modifizierter Switch angeschlossen ist.

Im ersten Fall werden die zu sendenden bzw. zu empfangenden IRT-Frames durch die Schedule verwaltet. Dennoch können auch von einem IRT-Gerät asynchrone Sendungen emfangen und gesendet werden. Dies geschieht, wenn das IRT-Gerät - beispielsweise eine Master-Achse - asynchron durch ein anderes Gerät abgefragt wird. Bei diesem anderen

Gerät kann es sich beispielsweise um ein Programmiergerät handeln, über das der Zustand der Achse abgefragt wird. Dieses Gerät muss Kenntnis des Echtzeit-Protokolls besitzen, mit dem das IRT-Gerät abgefragt werden kann. Der Switch muss dann diese asynchrone Anfrage so an das IRT-Gerät weiterleiten, dass der Echtzeitbetrieb nicht gefährdet ist.

Im zweiten Fall werden über diesen Port keine IRT-Frames weitergeleitet. Der Switch verhält sich an diesem Port wie ein herkömmlicher Ethernet-Switch.

Ist an dem betreffenden Port ein weiterer modifizierter Switch angeschlossen, so kann sowohl IRT- als auch nRT-Verkehr weitergeleitet werden. Da alle modifizierten Switches über eine synchronisierte Schedule verfügen, werden nRT-Übertragungen nur dann weitergeleitet, falls zwischen diesen Switches zur Zeit keine IRT-Übertragung in der Schedule eingeplant ist.

Empfängt ein modifizierter Switch einen asynchron versendeten Frame, so kann er diesen erkennen, da er an einem Port eintrifft, an dem zu diesem Zeitpunkt kein ankommender IRT-Frame in der Schedule vorgesehen ist. In diesem Fall kann der Switch den Datenframe zunächst zwischenspeichern und dessen Ziel-Adresse auslesen. Anhand dieser Adresse kann der zugehörige Ausgangsport ermittelt werden. Ist die Adresse nicht in der Weiterleitungstabelle vorhanden, so kann der Frame an einen Port oder an mehrere Ports ausgegeben werden (Hot-Potato-Verfahren), sofern diese nicht gerade durch die Schedule mit Echtzeit-Daten belegt sind. Ist der Ausgangsport bekannt, so kann anhand der vorliegenden Schedule entschieden werden, zu welchem Zeitpunkt der Frame weitergeleitet werden darf.

Dabei kann unter Umständen eine Fragmentierung des Frames durch den Switch durchgeführt werden, falls der freie Zeitslot an dem Ausgangsport zu klein ist. Dies kann nur durch ein proprietäres Protokoll innerhalb der echtzeitfähigen Switches erfolgen, wenn davon ausgegangen wird, dass die asynchron sendenden Geräte keine Kenntnis über Zusatzprotokolle besitzen dürfen. Durch die Fragmentierung können asynchrone Frames in der isochronen Phase Hop-by-Hop weitergeleitet werden. Erreichen diese Segmente den letzten modifizierten Switch an der Grenze des Netzwerkes zu Standard-Switches und/oder nichtechtzeitfähigen Geräten, so können die Segmente an diesem Switch gesammelt und reassembliert werden. Die Grenzen des echtzeitfähigen Netzes sind dadurch transparent. Ebenso können die asynchronen Frames transparent von den Geräten weitergeleitet werden, so dass die Geräte keine Kenntnis über zusätzliche Protokolle besitzen müssen. Herkömmliche PCs und/oder Laptops sind dadurch in das echtzeitfähige Netzwerk zur Laufzeit integrierbar, ohne den Betrieb der automatisierten Anlage zu stören.

Innerhalb der Zeitslots für asynchrone Daten können Frames mit weichen Echtzeitanforderungen - wie bereits bei ProfiNet SRT - mit VLAN-Headern priorisiert werden. Auf diese Weise wird eine fließende vertikale Integration innerhalb der Pyramide der Automatisierungstechnik ermöglicht.

5.3 Skizzierung eines echtzeitfähigen Switches

Konventionelle Switches analysieren zunächst die MAC-Adresse des Ziels eines Frames und leiten ihn dementsprechend an den passenden Ausgangsport weiter. Die durch die Analyse bedingte Verzögerung durch die Interpretation des Frames ist aufgrund der Schedule nicht mehr notwendig, da die Weiterleitung durch den Zeitpunkt des Eintreffens eines Frames

an einem bestimmten Port im Vorfeld bekannt ist. Die Weiterleitung kann also direkt erfolgen. Die erstellten Schedules, welche in die Switches integriert werden, setzen jedoch eine Möglichkeit zur variablen Verschaltung der Eingangs- und Ausgangsports zum Zwecke der Weiterleitung voraus. Um die Verzögerung der Frames gering zu halten, muss diese Umschaltung in der Protokollhierarchie möglichst nah am Medium erfolgen. In diesem Kapitel wird untersucht, auf welche Weise diese Umschaltung möglich ist.

Abbildung 5.10 stellt die Realisierung der beiden unteren Schichten des OSI-Schichtenmodells für $100Mbit/s$-Ethernet dar. Der Zugriff auf das physikalische Übertragungsmedium, bei dem es sich um ein 4-adriges Twisted-Pair Kupferkabel bei 100BaseTx oder um Glasfaserkabel bei 100BaseFx handelt, erfolgt über einen Physical Attachment Layer (PHY), der aus drei Unterschichten besteht. Dabei handelt es sich unter anderem um den Physical Coding Sublayer (PCS) zur Codierung und Decodierung der Daten-Nibbles in 4B/5B-Codegruppen. Des Weiteren generiert der PCS die Carrier-Sense und Collision-Detect Signale [Hei98], die im Halbduplexbetrieb von Bedeutung sind und von der Sicherungsschicht ausgewertet werden. Der zweite Teilbereich des PHY besteht aus dem Physical Media Attachment (PMA), der zusammen mit dem dritten Teil, dem Physical Media Dependent (PMD) den Tranceiver bildet. Dieser Tranceiver besitzt einen medienspezifischen Anschluss, der als Media Dependent Interface (MDI) bezeichnet wird. Der verwendete Tranceiver ist abhängig vom Übertragungsmedium und dient als Bindeglied zwischen dem Medium und dem Gerät [Hei98].

In der PMD-Teilschicht werden physikalischen Eigenschaften beschrieben, Übertragungsparameter - beispielsweise die Übertragungsgeschwindigkeit - definiert und die passende Signalisierung zum Übertragungsmedium realisiert. Das PMA ist hauptsächlich zuständig für die Sende- und Empfangsfunktion, die Autonegotiation sowie für die Kollisionserkennung auf dem Medium. Der Physical Attachment Layer ist auf Netzwerkkarten und auch in Switches auf einem Baustein implementiert, der ebenfalls als PHY bezeichnet wird und direkt mit dem Ethernetport verbunden ist.

Es ist nicht zu empfehlen, das Scheduling auf der Ebene des MDI oder des PHY anzusetzen, da die angeschlossen Geräte bei einer Umschaltung kurzzeitig ihren Link verlieren würden. Dies hat zur Folge, dass unmittelbar nach der Umschaltung der Prozess der Autonegotiation eingeleitet wird, um die Übertragungsgeschwindigkeit und das Duplexverfahren neu zu erkennen und zu konfigurieren. Die dadurch entstehende Verzögerung ist zu vermeiden.

Zu der höheren Schicht kommuniziert der PHY-Baustein über die Verwendung einer standardisierten Schnittstelle, dem *Media Independent Interface* (MII) [Fra95]. Diese Schnittstelle ist unabhängig vom Übertragungsmedium und wird im folgenden Kapitel genauer betrachtet. Die Kommunikation erfolgt über eine Zwischenschicht direkt unterhalb der Sicherungsschicht, die als Reconciliation Layer bezeichnet wird. Die Aufgabe dieser Schicht besteht darin, die Dienstprimitive des Physical Line Signaling (PLS), die von der Media Access Control stammen, in Signale umzuwandeln, die dem MII entsprechen. Diese Dienstprimitive lauten PLS_DATA.request für das Absetzen von Daten der MAC zum PHY, PLS_DATA.indicate für den Empfang von Daten, PLS_SIGNAL.indicate für eine Kollisionserkennung sowie PLS_CARRIER.indicate für die Überwachung des Übertragungsmediums.

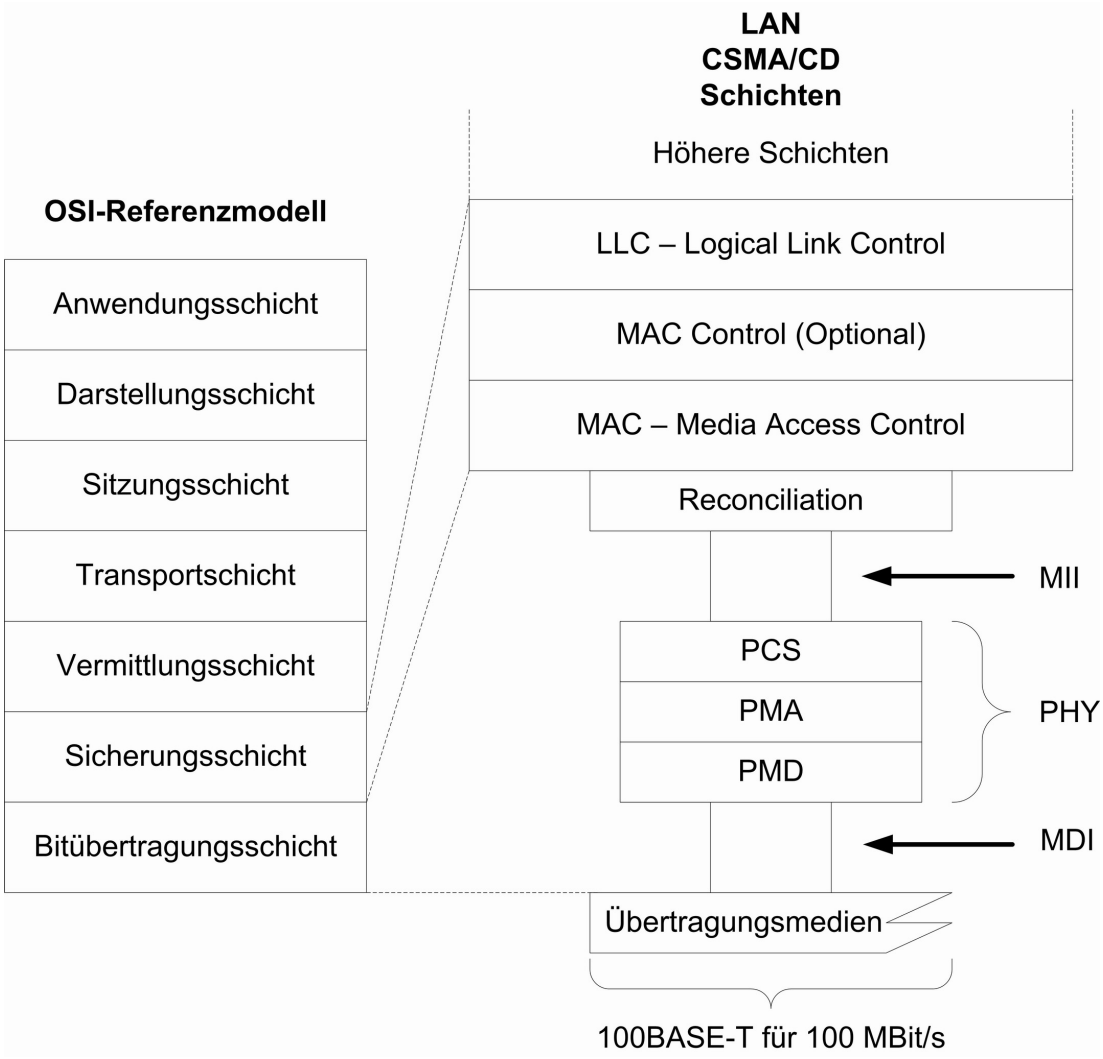


Abbildung 5.10: Realisierung von Ethernet auf OSI-Schicht 1 und 2 [Hij07]

Der Reconciliation Layer ist zusammen mit der MAC und der Logical Link Control (LLC) auf einem weiteren Baustein integriert, der insgesamt als MAC bezeichnet wird. Heutzutage werden in einem Switch mehrere MAC-Bausteine und die gesamte Switching-Logik in einen einzelnen Chip integriert, der über einen gemeinsamen Speicher verfügt, um die Frames an die Ausgangsports weiterzuleiten. Dieser Chip ist neben der Weiterleitung der Frames anhand der Zieladresse auch für die Kontrolle der Prüfsumme verantwortlich, sofern es sich um einen store-and-forward Switch handelt. Des Weiteren verwaltet dieser Chip die Weiterleitungstabellen und die Warteschlangen für ausgehende Frames. Er ist ebenso verantwortlich für die Ausführung von Spanning-Tree-Protokollen und die bevorzugte Weiterleitung von VLAN-priorisierten Frames.

5.3.1 Das Media-Independent-Interface (MII)

Das MII bildet einen hardware- und medienunabhängigen Standard, der von nahezu allen Herstellern von Ethernet-Geräten zwischen der physikalischen Schicht und der Siche-

rungsschicht eingehalten wird. Im Folgenden werden die wichtigsten Signale des MII kurz erläutert, die auf TTL-Signalpegeln basieren, vgl. [Hei98].

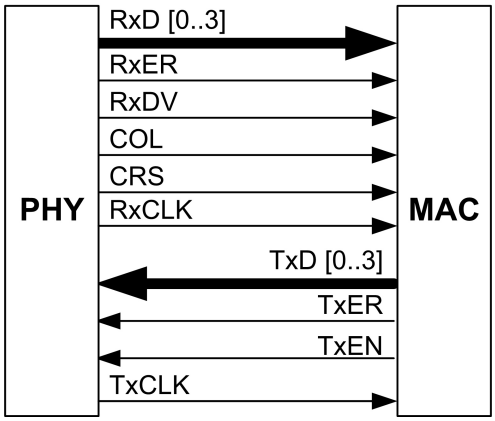


Abbildung 5.11: Die wichtigsten Signale des MII

Die empfangenen (RxD) und die zu sendenden Daten (TxD) werden jeweils in Nibbles synchron zu ihren Taktsignalen RxCLK bzw. TxCLK übertragen. Hierbei fällt auf, dass beide Takte von dem PHY vorgegeben werden. Beim Senden von Daten muss also der MAC die Daten synchron zum vorgegebenen TxCLK-Signal bereitstellen. Wird das RxER-Signal für eine oder mehrere Takte gesetzt, so bedeutet dies, dass fehlerhafte Daten empfangen wurden. Das gleiche gilt für das TxER-Signal für Sendungen vom MAC zum PHY. Das RxDV-Signal wird gesetzt, sobald der PHY das erste Nibble der Präambel erkannt hat und als Daten bereitstellt. Es bleibt so lange gesetzt, bis das letzte empfangene Nibble eines Frames ausgelesen wurde. Für sendende Daten ist das Signal TxEN vorgesehen, welches ein gültiges zu sendendes Nibble signalisiert. Das Signal zur Erkennung einer Kollision COL hat im Vollduplexbetrieb ebenso wenig Bedeutung wie das Carrier-Sense Signal CRS. Das Verhalten der Signale ist nach IEEE 802.3 im Vollduplexbetrieb nicht genormt [IEEE05].

Um die Verzögerungszeit eines übertragenen Datenframes zu ermitteln, sind zunächst die Verzögerungen des empfangenen und des sendenden PHY zu addieren. Die Zeit zum Empfangen eines Frames wird exemplarisch an dem L80227 Ethernet PHY der Firma LSI Logic [Lsi02] in der Abbildung 5.12 sowie 5.13 dargestellt. Bei den Signalen $TPI+/-$ handelt es sich um die Übertragung auf dem Medium, dessen Daten nach der Leseverzögerung as TxD-Nibbles zur Verfügung stehen. Das MII ist mit $25MHz$ getaktet. Von Interesse ist die Zeit $t33$, welche die Zeitspanne des Frame-Empfangs zu dem Setzen der Leitung RxDV beinhaltet. Sie beträgt bei der $100Mbit/s$-Übertragung maximal $240ns$. Dazu ist die Verzögerung der RxCLK zu RxDV und RxD zu addieren, die als $t37$ bezeichnet ist und $8ns$ beträgt.

Dieser PHY-Baustein ist also nach maximal $240ns/60ns = 6$ übertragenen Nibbles synchronisiert und setzt das Signal RxDV. Dies bedeutet, dass bei Verwendung dieses Bausteins maximal $3Byte$ der Präambel eines Ethernet-Frames durch die Synchronisationsphase verloren gehen können.

Auf die gleiche Weise kann die Verzögerung der TxD-Signale des MII zu den analogen Signalen der Übertragungsleitung bestimmt werden. Das Transmit Propagation Delay ist in der Abbildung 5.13 als $t23$ bezeichnet und beträgt maximal $140ns$. Die Verzögerungszeit

auf der physikalischen Schicht liegt bei Verwendung dieses Bausteins unter $400ns$ und ähnelt der Verzögerung eines Hubs.

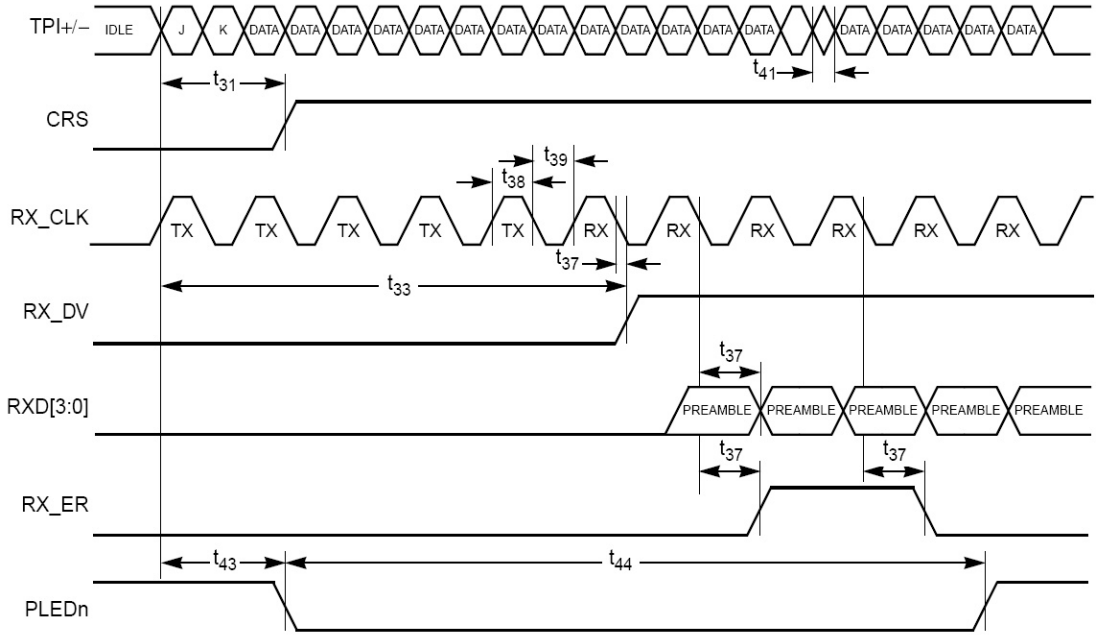


Abbildung 5.12: MII-Verzögerungszeiten bezüglich des Sendens [Lsi02]

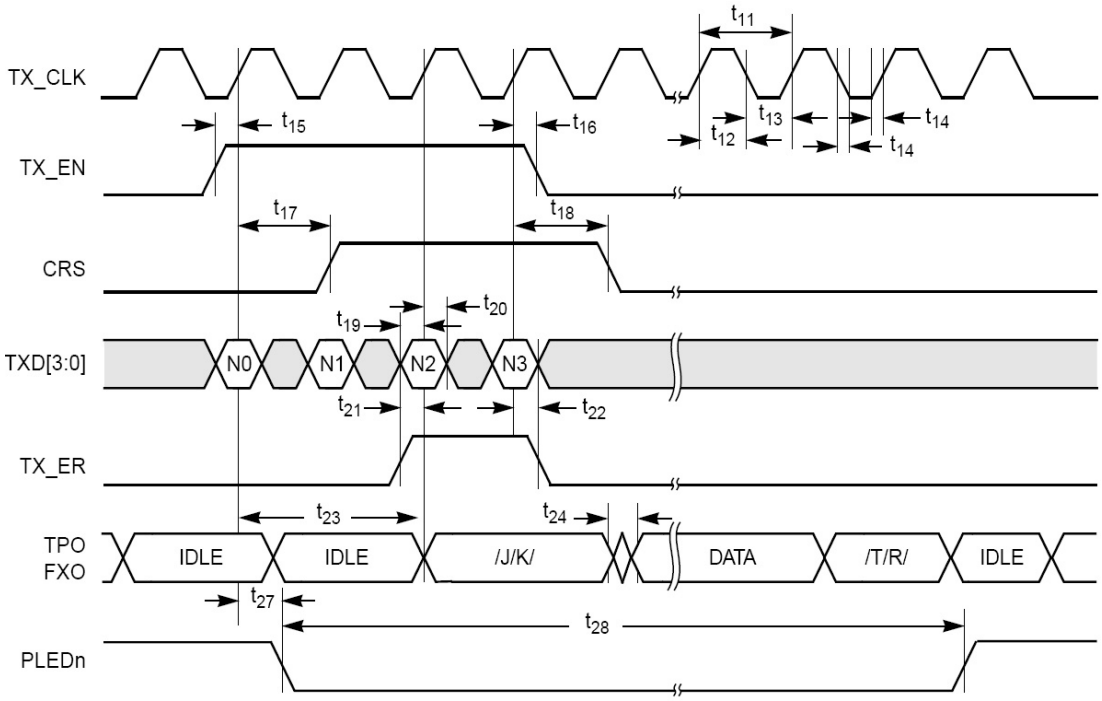


Abbildung 5.13: MII-Verzögerungszeiten bezüglich des Empfangens [Lsi02]

5.3.2 Das Reduced Media-Independent-Interface (RMII)

Neben der MII-Spezifikation existiert eine weitere Schnittstelle, welche zwischen der physikalischen Schicht und der Sicherungsschicht des Ethernets verbreitet ist. Im Gegensatz zum MII, bei dem die Daten in Nibbles weitergereicht werden, werden beim RMII in jedem Takt zwei Datenbits übertragen. Auf diese Weise werden jeweils 2 Leitungen für das Senden und 2 Leitungen für den Datenempfang eingespart. Diese Reduzierung hat jedoch zur Folge, dass die Taktung des RMII verdoppelt werden muss. Anstatt mit einer internen Taktung zu arbeiten und diesen Takt als Ausgangssignal zur Verfügung zu stellen, erlaubt das RMII eine externe Taktung mit $50MHz$ (REF_CLK). Damit kann ein einzelnes Taktsignal für mehrere PHY-Bausteine mit RMII-Schnittstelle bereitgestellt werden, die bei einer dynamischen Verschaltung der PHYs einen zentralen Takt bilden.

Die Bedeutung der Leitung Receive Error (RxER) ist identisch zu der Bedeutung im MII, die Leitung Carrier Sense / Data Valid (CRS_DV) fasst die beiden Leitungen CRS und RxDV des MII durch Multiplexing auf alternierende Takte zusammen.

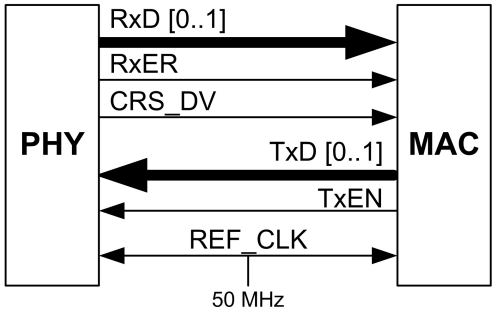


Abbildung 5.14: Die wichtigsten Signale des RMII [RMI98]

Die Möglichkeit, zwei PHY-Bausteine direkt über die RMII-Schnittstelle zu verbinden, wird bereits von dem Ethernet Transceiver DP83848 Single $10/100Mb/s$ von [Nat06] National Semiconductor berücksichtigt. Das Ziel ist es hierbei, die Funktion eines Repeaters zu gewährleisten. Abbildung 5.15 zeigt die Crosslink-Verschaltung von zwei DP83848-Bausteinen durch Kreuzung der Sende- und Empfangsleitungen auf MII-Ebene.

Zu beachten ist dabei, dass das Signal RxDV nicht zum RMII-Standard gehört und bei einer Verschaltung von zwei DP83848 Bausteinen direkt mit TxEN verbunden wird. Für eine Verwendung als herkömmlicher PHY-Baustein wird RxDV nicht verwendet.

Um die Verzögerung eines Frames zu ermitteln, sind die Zeiten für den Empfang und das erneute Versenden der betreffenden PHY-Bausteine zu addieren. Die Timing-Diagramme sind in den Abbildungen 5.16 und 5.17 dargestellt und ergeben eine Verzögerung für diesen Baustein von $T2.27.5 + T2.26.4 = (38 + 17)bits \cdot 40ns = 2.2\mu s$.

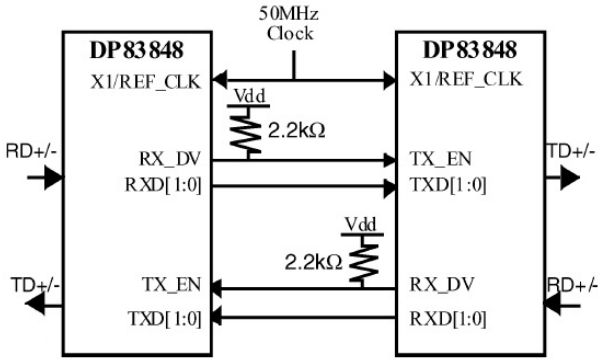
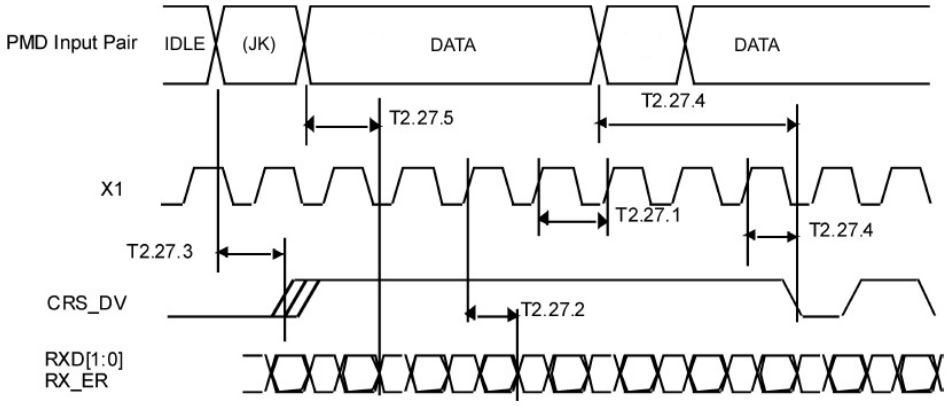
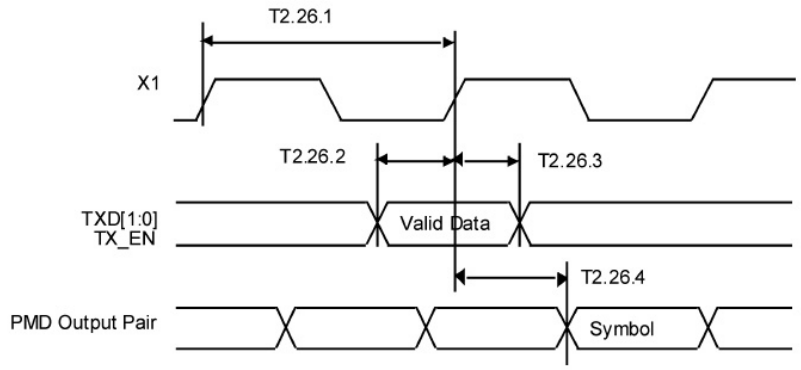


Abbildung 5.15: Aufbau eines Repeaters mittels 2 DP83848-Bausteinen [RMI98]



Parameter	Description	Notes	Min	Typ	Max	Units
T2.27.1	X1 Clock Period	50 MHz Reference Clock		20		ns
T2.27.2	RXD[1:0], CRS_DV and RX_ER output delay from X1 rising		2		14	ns
T2.27.3	CRS ON delay	From JK symbol on PMD Receive Pair to initial assertion of CRS_DV		18.5		bits
T2.27.4	CRS OFF delay	From TR symbol on PMD Receive Pair to initial deassertion of CRS_DV		27		bits
T2.27.5	RXD[1:0] and RX_ER latency	From symbol on Receive Pair. Elasticity buffer set to default value		38		bits

Abbildung 5.16: RMII-Verzögerungszeiten bezüglich des Sendens [RMI98]



Parameter	Description	Notes	Min	Typ	Max	Units
T2.26.1	X1 Clock Period	50 MHz Reference Clock		20		ns
T2.26.2	TXD[1:0], TX_EN, Data Setup to X1 rising		4			ns
T2.26.3	TXD[1:0], TX_EN, Data Hold from X1 rising		2			ns
T2.26.4	X1 Clock to PMD Output Pair Latency	From X1 Rising edge to first bit of symbol		17		bits

Abbildung 5.17: RMII-Verzögerungszeiten bezüglich des Empfangens [RMI98]

5.3.3 Die Erstellung einer Real-Time Crossbar (RTC)

Die Idee zur Implementierung einer Verschaltung von Ports am MII bzw. RMII liegt nahe. Einerseits handelt es sich um eine standardisierte Schnittstelle, die einfach aufgebaut ist. Andererseits wird die physikalische Schicht gekapselt, so dass eine Implementierung erfolgen kann, die unabhängig vom Übertragungsmedium ist. Aus Hardwaresicht können prinzipiell handelsübliche PHY- und MAC-Bausteine eingesetzt werden, zwischen die der zu entwickelnde Baustein integriert wird. Dieser besitzt MII- oder RMII-Anschlüsse sowohl zur physikalischen Schicht, als auch zur Sicherungsschicht. Abbildung 5.18 skizziert die zu erstellende *Real-Time Crossbar* (RTC).

Um unabhängige Vollduplexübertragungen zu gewährleisten, müssen die sendenden und empfangenden Teile jedes Ethernetports unabhängig geschaltet werden können. Die Idee der RTC liegt also in dynamischen Crosslink-Verbindungen der (R)MII-Schnittstellen oberhalb der physikalischen Schicht. Diese Crosslink-Verbindungen schalten entsprechend der Schedule, die offline berechnet und in die Switches in der Hochlaufphase der Anlage geladen wurde. Nach einer Synchronisation der lokalen Schedules, welche ebenfalls in der Hochlaufphase durchgeführt werden kann, sind die Switches betriebsbereit.

Die Abbildung zeigt beispielhaft, dass vom eingehenden Port 1 direkt zum ausgehenden Port 0 des Switches weitergeleitet wird. Die Weiterleitung erfolgt also zwischen Schicht 1 und 2 des OSI-Modells ohne eine Interpretation des Ethernet-Frames. Die Verschaltung der Ports erfolgt entsprechend der Schedule, die in den Switch geladen wurde. Werden Ports in einem Zeitslot nicht verwendet, so werden diese über die RTC mit dem MAC der Switching-Engine verbunden. Treffen in dieser Zeit Frames ein, so muss es sich um asynchron versendete Frames handeln, die von der Switching Engine verarbeitet werden können. Dabei ist es sinnvoll, einen store-and-forward Switch zu verwenden, der Frames speichern kann und über genügend große Sendepuffer verfügt. Wird der betreffende Ausgangsport durch die Schedule freigegeben, so kann der asynchrone Frame weitergeleitet

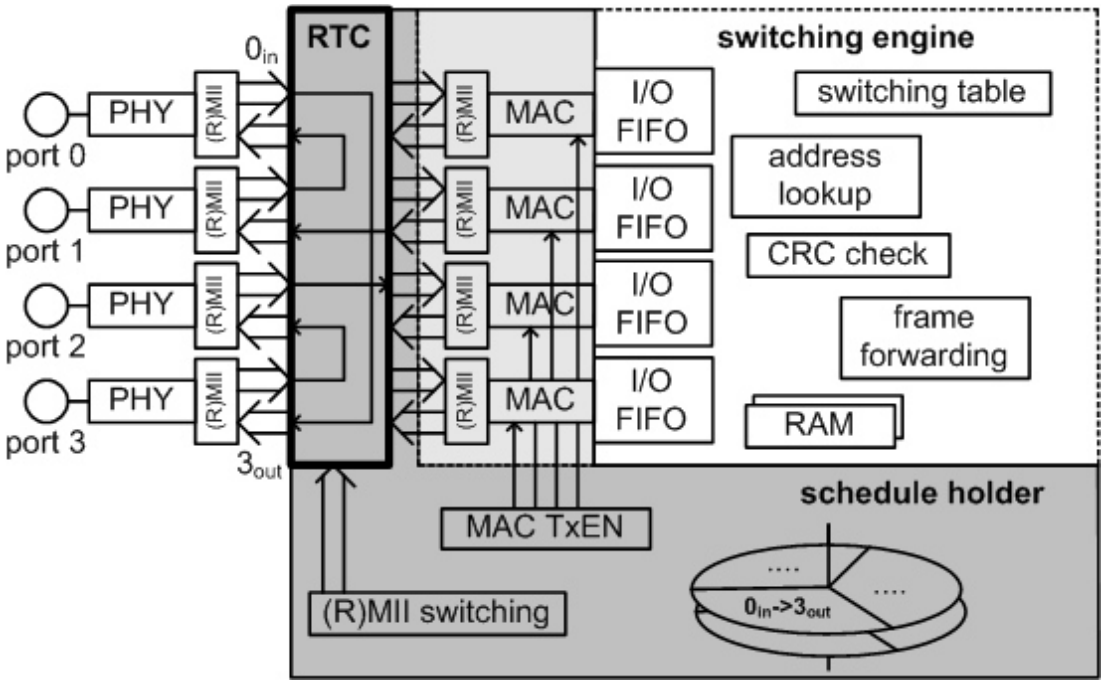


Abbildung 5.18: Integration der Schedule durch eine RTC [DW07]

werden. Handelt es sich bei einem Port um einen nRT-Port (vgl. Kapitel 5.2.6), so bleibt dieser Port zu jedem Zeitpunkt mit dem MAC-Bausteins des Switches verbunden, wird also nicht zu einem anderen Port verschaltet.

Als technische Umsetzung kann entweder eine eigene Switching Engine erstellt oder eine vorhandene genutzt werden. Kommt eine eigene Engine zur Anwendung, so können die asynchronen Frames von der Engine unter Verwendung eines eigenen Protokolls fragmentiert bzw. am letzten echtzeitfähigen Switch im Netzwerk reassembliert werden, um auch innerhalb der isochonen Phase Daten weiterleiten zu können.

Wird eine vorhandene Switching Engine verwendet, so entsteht das Problem, die asynchronen Frames aus den Sendepuffern an den dazugehörigen PHY-Baustein zu übertragen. Da die MAC-Bausteine üblicherweise fest mit den PHY-Bausteinen verdrahtet sind, wird der Inhalt eines Sendepuffers direkt an den zu sendenden PHY übergeben. Im Halbduplexbetrieb wird durch Setzen des COL-Signals die Weiterleitung verhindert und das Exponential-Backoff Verfahren in der Switching Engine ausgelöst. Auch das CRS-Signal, welches das Medium als belegt kennzeichnet, verhindert im Halbduplexbetrieb die Weiterleitung. Beide Signale können im Vollduplexbetrieb ignoriert werden, s. Kapitel 5.3.1.

Das Problem besteht darin, dass die PHY-Bausteine im IRT-Betrieb direkt miteinander verbunden sind, um eine unmittelbare Weiterleitung der echtzeitkritischen Frames zu ermöglichen. In diesem Betrieb sind die (R)MII-Anschlüsse des sendenden MAC mit keinem PHY-Baustein verbunden. Dies kann im Vollduplexbetrieb dazu führen, dass die in den Sendepuffern befindlichen Frames verworfen werden. Eine Möglichkeit, das Verwerfen nach IEEE-Norm [IEEE05] zu verhindern, besteht nicht. Um dieses Problem zu lösen, wurden bereits in [DW07] einige Ansätze diskutiert, die im Folgenden beschrieben werden.

Der erste Ansatz besteht darin, einen MAC-Baustein zu verwenden, der auch im Vollduplexmodus beim Senden das CRS-Signal interpretiert und solange nicht sendet, bis das

Medium als „frei“ gekennzeichnet ist. Die Steuerung des CRS-Signals wird von der Steuerung der RTC, dem „Schedule Holder“[4], übernommen. Bevor die Leitungen TxD, TxER und TxEN für den IRT-Betrieb von dem zugehörigen MAC-Baustein abgekoppelt werden, wird zunächst das Medium über das CRS-Signal als „belegt“ markiert. Steht der sendende Teil des PHY-Bausteins wieder für asynchrone Frames zur Verfügung, werden zunächst die betreffenden Leitungen durch die RTC wieder mit dem MAC-Baustein verbunden und im Anschluss daran das Medium als „frei“ markiert.

Um mit allen MAC-Bausteinen kompatibel zu sein, besteht die zweite Möglichkeit darin, den Switch durch Verwendung der Autonegotiation im Halbduplex-Modus zu betreiben. Dadurch wird sichergestellt, dass der MAC-Baustein auf das CRS-Signal reagiert. Dies muss natürlich bei der Berechnung der Schedule berücksichtigt werden. Der Nachteil dieser Lösung besteht darin, dass die zur Verfügung stehende Bandbreite des Netzwerkes reduziert wird.

Als weitere Alternative ist die Modifikation der MAC-Bausteine zu nennen. In die VHDL-Beschreibung eines vorhandenen MAC-Bausteins kann ohne größeren technischen Aufwand eine weitere Leitung aufgenommen werden, die man als MAC-TxEN bezeichnen kann. Der VHDL-Code von MAC-Bausteinen kann als Bibliothek erworben werden [GI06] und ist teilweise unter GNU GPL frei erhältlich. Die Modifikation besteht darin, dass der MAC-Baustein zu sendende asynchrone Frames nur bei gesetztem MAC-TxEN Signal weiterleitet und die Frames ansonsten in der Ausgangswarteschlange des Switches verbleiben.

Um das Weiterleiten von asynchronen Daten mit Sicherheit zu verhindern, kann als vierter Ansatz eine eigene Switching Engine verwendet werden. Diese eigene Engine muss MAC-Bausteine beinhalten, welche auf das CRS-Signal auch im Vollduplexbetrieb reagieren, indem sie die zu sendenden Frames in den Ausgangspuffern erhalten. Dies bedingt jedoch die nahezu vollständige Neuentwicklung eines Switches und führt zu einer Lösung, welche dem ERTEC-Switch des ProfiNet-Ansatzes entspricht.

5.3.4 Technische Schwierigkeiten

Im Fokus dieser Arbeit steht die Untersuchung, ob eine dynamische Verschaltung der MII-Schnittstellen an den Ausgängen der PHY-Bausteine möglich ist. Zu diesem Zweck wurden als erster Test die MII-Schnittstellen von PHY-Bausteinen direkt miteinander gekreuzt verbunden. Dazu wurden zwei 78Q2123-DB MicroPHY MII Evaluation Boards der Firma Teridian [Ter05] verwendet, die das MII nach außen über einen Stecker zugänglich machen. Abbildung 5.19 zeigt den PHY-Baustein mit RJ45-Buchse auf der rechten Seite und der herausgeführten MII-Schnittstelle als 40-poligen Stecker auf der linken Seite.

Das erste Problem besteht darin, dass die PHY-Bausteine intern getaktet sind und sowohl den Sende-, als auch den Empfangstakt lediglich als Ausgangssignale am MII bereitstellen, s. Abbildung 5.11. Dies bedeutet, dass die MII-Leitungen zwischen zwei PHY-Bausteinen nicht problemlos gekreuzt werden können. Statt dessen muss die Weiterleitung eines Frames, der gerade empfangen wird, mit dem Sendetakt des ausgehenden PHY-Bausteins synchronisiert werden. Dabei ist sowohl die Phasenlage, als auch die Drift zwischen

[4]Dabei kann es sich um einen eigenen FPGA- oder ASIC-Baustein handeln. Alternativ dazu kann auch der Schedule Holder und die RTC in einem einzelnen Chip integriert werden.

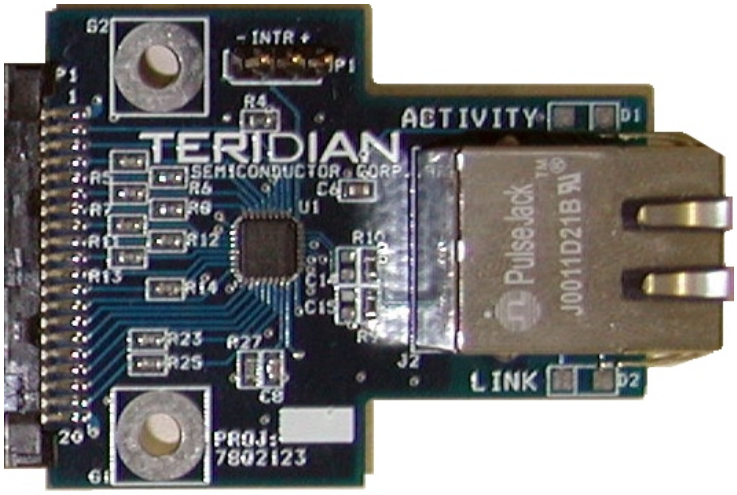


Abbildung 5.19: Evaluationboard eines Teridian-PHYs

den beiden Takten, die nach IEEE 802.3 Norm [IEEE05] bis zu $100ppm$ bei $100Mbit/s$-Ethernet betragen kann, zu berücksichtigen. Bis die Synchronisation erfolgt ist, müssen einige Nibbles des Frames, die bereits empfangen wurden, zwischengespeichert werden. Eine Schaltung, welche dies leistet, wird im folgenden Kapitel vorgestellt.

Ein weiteres Problem liegt darin, dass der PHY-Baustein beim Empfang der Präambel eine gewisse Zeit benötigt, um seinen Takt mit dem eintreffenden Frame zu synchronisieren. Der empfangene PHY-Baustein benötigt also einige Takte, um den eintreffenden Frame zu erkennen. Dies entspricht dem Zweck der Präambel. Die Synchronisierung muss lediglich vor dem SFD angeschlossen sein. Erst nach der Synchronisation werden die Daten an das MII weitergegeben und das RxDV-Signal gesetzt. Dies wird in Abbildung 5.20 dargestellt.

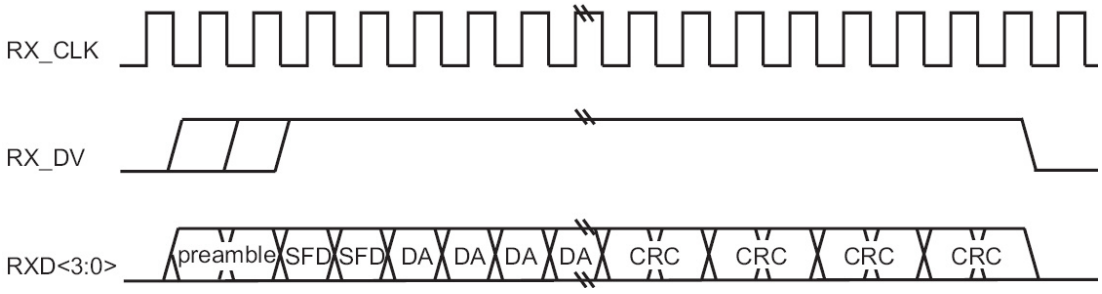


Abbildung 5.20: Beziehung zwischen den Signalen RxDV und RxD

Dadurch ist es möglich, dass beim Empfang der Daten an der MII-Schnittstelle maximal $7Byte$ der Präambel fehlen und der Frame mit dem SFD beginnt. Bevor dieser Frame weitergeleitet wird, ist jedoch die Präambel zu vervollständigen, da ansonsten bei einer Weiterleitung über mehrere Switches evtl. der SFD nicht mehr erkannt werden kann. In diesem Fall wird der Frame als „defekt“ interpretiert. Zur Lösung dieses Problems wird im folgenden Kapitel ein Ansatz beschrieben, der eine mögliche Umsetzung mit entsprechender Hardware skizziert.

5.3.5 Lösungsansätze

In einem ersten Schritt muss die Weiterleitung eines empfangenen Frames synchron zum Taktsignal des sendenden PHY erfolgen. Zu diesem Zweck werden die empfangenen Nibbles zunächst mit dem Takt des emfangenden PHYs in einen FIFO-Speicher geschrieben, der wie auch im ProfiNet IRT Switch (s. Kapitel 3.1.3.1) als Dual-Port RAM [Hij07] ausgelegt ist. Der Ringspeicher wird dann mit dem Takt des sendenden PHYs wieder ausgelesen. Dabei ist zu beachten, dass der Ringspeicher weder überläuft noch leer wird. Ist der Sender schneller als der Empfänger, so müssen vor Beginn des Sendens einige Nibbles des Ethernet-Frames im Zwischenspeicher vorgehalten werden. Ist der Sender langsamer, so muss der Puffer die entsprechende Anzahl an Nibbles vom Empfänger aufnehmen können. Da man jedoch von kleinen Framelängen bei IRT-Frames ausgeht, ist dieser Effekt sehr begrenzt.

Ein im $100Mbit/s$-Ethernet verwendeter Taktgeber für MII arbeitet mit $25MHz$. Dies bedeutet, dass bei einer Drift von $100ppm$, die vom Standard als Obergrenze vorgegeben wird, eine obere Frequenz $f_o = 25.002500MHz$ und eine untere Frequenz $f_u = 24.997500MHz$ toleriert wird. Dadurch dauert ein Takt maximal $t_o = 40.004ns$ und minimal $t_u = 39.996ns$ lange an. Ein minimal großer Ethernet-Frame ist incl. Präambel $72Byte$ groß und benötigt 144 Takte, da die Datenübertragung am MII in Nibbles erfolgt. Diese Übertragung benötigt mit t_u eine Zeit von $5760.576ns$ und mit t_o eine Zeit von $5759.424ns$. Die Drift liegt also deutlich unterhalb eines Taktes.

Als Sicherheit genügt es also, den Zwischenspeicher so zu wählen und zu füllen, dass der Füllstand stets um 2 Nibbles schwanken kann: Ein Takt für die Anpassung der Phasenlage zwischen RxCLK des sendenden PHYs und TxCLK des empfangenden PHYs sowie ein Takt für den Ausgleich der Drift.

Die Signalleitungen bei großen, asynchron versendeten Frames werden vom empfangenden PHY-Baustein auf den MAC-Baustein durch eine direkte Verbindung weitergeschaltet. Wird ein solcher Frame vom Switch weiterversendet, so erfolgt eine direkte Verbindung des MAC-Bausteins zu dem sendenden PHY. Die Synchronisation ist demnach lediglich für IRT-Frames aktiv und wird lediglich bei der Verschaltung der PHY-Bausteine eingesetzt.

Unabhängig von der Synchronisation können Teile der Präambel bei dem empfangenen Frame fehlen. Um wieder eine vollständige Präambel zu senden, wird der binäre Code einer vollständigen Präambel im FPGA gespeichert. Wird das erste Nibble der Präabel vom FPGA empfangen, so wird unmittelbar mit dem Senden des ersten Nibbles der vollständigen Präambel begonnen.

Ein empfangender Frame wird von einem PHY dann noch als gültig erkannt, wenn der SFD als erstes gültiges Byte erkannt wird. Die 14 Nibbles der Präambel können vollständig zur Synchronisation des empfangenden PHY-Bausteins verwendet werden. Alternativ dazu kann auch eine völlige Synchronität zum empfangenden Frame bestehen, so dass kein Nibble der Präambel für die Synchronisation verwendet wird. Daraus resultiert eine Untergrenze für die Empfangssynchronisierung von 0 Takten und eine Obergrenze von 14 Takten. Zusammen mit der Anpassung der Phasenlage und mit dem Drift-Ausgleich resultiert eine Gesamtverzögerung zwischen 1 und 16 Takten. Dies entspricht $0.08\mu s$ bzw. $0.56\mu s$. Die Obergrenze der Empfangssynchronisierung des exemplarisch ausgewählten L80227 Ethernet PHY-Baustein der Firma LSI Logic (vgl. Kapitel 5.3.1) liegt bei 6 Takten und damit zwischen $0.08\mu s$ und $0.32\mu s$. Dazu addieren sich die Verzögerungen des empfangenden und weiterleitenden PHY-Bausteins, die $0.24\mu s + 0.14\mu s = 0.38\mu s$ betragen. Die Verzöge-

rung des FPGA-Bausteins, der die Umschaltung realisiert und die Schedule behinhaltet, kalkuliert Hijazi [Hij07] mit $0.08\mu s$. Die Gesamtverzögerungszeit eines Switches liegt bei ca. $0.8\mu s$ und ist damit deutlich geringer als die Verzögerung eines ProfiNet-Switches mit $3.0\mu s/Switch$.

Der an der MII-Schnittstelle eingefügte *RTC-Core* besteht im Wesentlichen aus einem Speicher für die geladene Schedule, der Crossbar zum dynamischen Verschalten der MII-Kanäle sowie aus dem Modul RTC-Sync. Dieses Modul vervollständigt die Präambel und synchronisiert den Sende- mit dem Empfangstakt. Abbildung 5.21 zeigt den Aufbau des RTC-Cores.

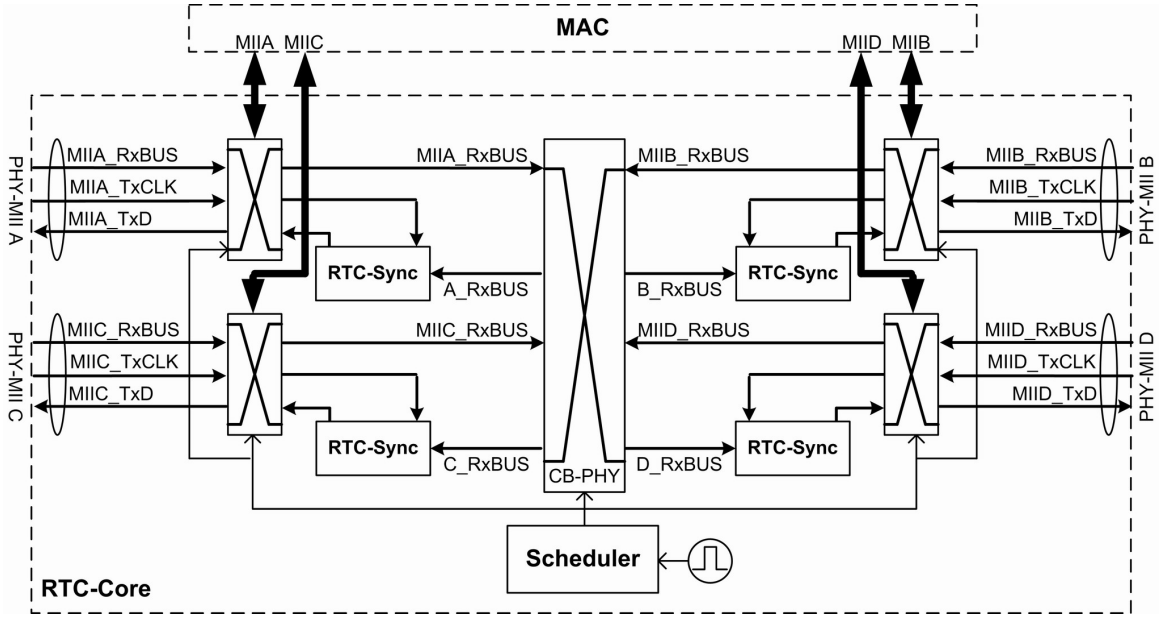


Abbildung 5.21: Blockschaltbild des RTC-Core Moduls [Hij07]

Die Signale RxD, RxDV, RxCLK und RxER sind als RxBUS zusammengefasst. Ebenso sind TxD, TxEN und TxER als TxBUS vereinigt. Die vier Crossbars an den Schnittstellen des RTC-Cores zu den PHY-Bausteinen erlauben es, die Ports direkt zur Switching Engine durchzuschalten, um asynchrone Frames zu den MAC-Bausteinen der Switching Enging bzw. zu der Kreuzung CB-PHY oberhalb der physikalischen Ebene weiterzuleiten.

Im zweiten Fall handelt es sich um einen IRT-Frame. Die Umschaltung der Crossbars wird vom Scheduler vorgegeben. Werden die PHY-Bausteine direkt gekreutzt, so verlaufen die Signale über das RTC-Sync Modul, welches in Abbildung 5.22 näher betrachtet wird. Dieses Modul ist auf der jeweiligen Sendeseite des CB-PHY plaziert. Der Grund dafür liegt darin, dass auf diese Weise bei der Umschaltung des RxBUS durch die CB-PHY Crossbar der Empfangstakt RxCLK einschwingen kann, bevor das Eingangssignal RxDV gesetzt wird. Gilt $RxDV = 0$ an dem Eingang des RTC-Sync, so ist das Modul abgeschaltet und der entsprechende PHY-Baustein sendet nicht.

Das *RTC-Sync* Modul besteht im Wesentlichen aus einer RX- und einer TX-Zustandsmaschine sowie dem DPRAM. Solange keine Daten von der CB-PHY anliegen, gilt $RxDV = 0$ und die RX-Zustandsmaschine ist im Idle-Zustand. Sobald $RxDV = 1$ gesetzt wird, liegt auch RxCLK stabil an und die RX-Zustandsmaschine sucht im Datenteil des RxBUS nach

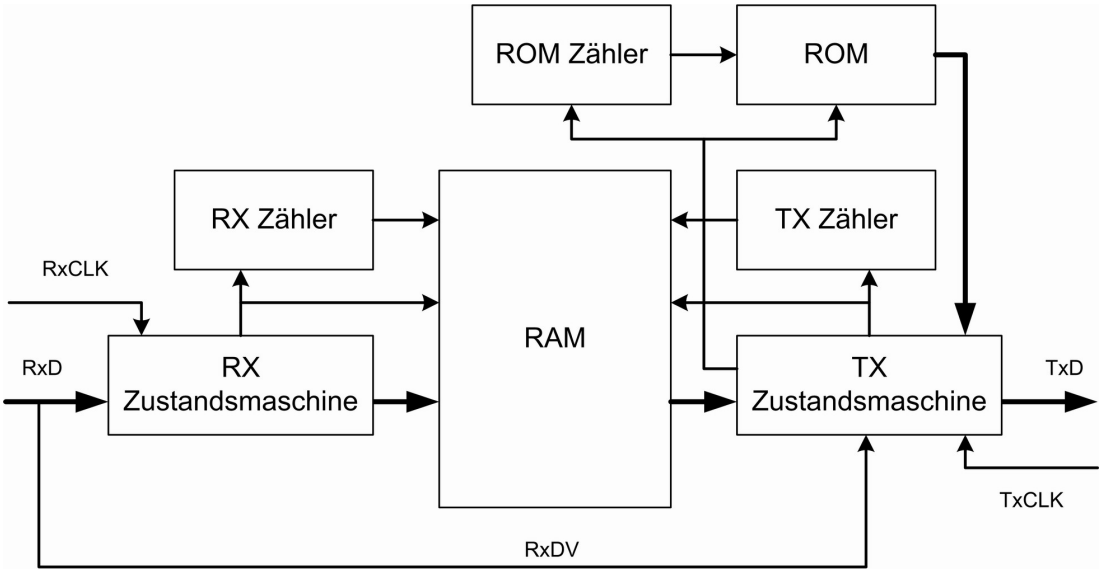


Abbildung 5.22: Blockschaltbild des RTC-Sync Moduls [Hij07]

dem SFD-Nibble des Frames. Alle Nibbles bis einschließlich des SFD-Nibbles werden verworfen, so dass die Präamble und der SFD vom Ethernet-Frame abgeschnitten wird. Nach dem Empfang des SFD werden alle Nibbles in das Dual-Port RAM geschrieben.

Auch die TX-Zustandsmaschine reagiert auf das RxDV-Signal und befindet sich im Idle-Zustand, sofern dieses Signal noch nicht gesetzt ist. Bei $RxDV = 1$ wird nach ein bis zwei Takten für den Ausgleich der Drift und der Phasenlage die Präambel ausgelesen, die sich fest im ROM befindet. Im Anschluss daran werden die Daten des Dual-Port RAMs in der Reihenfolge ausgelesen, in der sie von der RX-Zustandsmaschine geschrieben worden sind. Sind alle Daten gelesen, wechselt die TX-Zustandsmaschine wieder in den Idle-Zustand.

Diese einfachen Schaltungen der RTC-Sync und RTC-Core wurden erfolgreich in VHDL erstellt und simuliert [Hij07]. Dabei besteht noch Optimierungspotential bezüglich der Minimierung der Verzögerungszeit und/oder des Jitters. Alternativ dazu ist auch eine RTC auf Basis der RMII-Schnittstelle, beispielsweise unter Verwendung des Ethernet Transceivers DP83848 (s. Kapitel 5.3.2) denkbar. Der zusätzliche Schaltungsaufwand wird dadurch ebenso minimiert wie die Anzahl der Leitungen. Die Taktfrequenz des FPGA muss jedoch auf $50MHz$ verdoppelt werden.

Durch VHDL-Entwurf und Synthese [Hij07] wurde gezeigt, dass eine Realisierung auf Basis der FPGAs Spartan-3E [Xil07] oder Virtex-5 [Xil07b] der Firma Xilinx möglich ist. Abbildung 5.23 zkizziert die mögliche Realisierung eines modifizierten Switches auf der Basis von zwei FPGAs. Der Virtex-5 LXT bzw. Virtex-5 SXT integriert bereits vier MAC-Bausteine sowie vier FIFO-Warteschlangen sowie einen Intellectual Property Core (IP-Core) zur Realisierung der Switching-Engine. Ein solcher FPGA kann mit dem RTC-Core und dem Scheduler verbunden werden, die beide auf dem Spartan-3 FPGA implementiert sind. Dieser wiederum ist verbunden mit vier handelsüblichen PHY-Bausteinen.

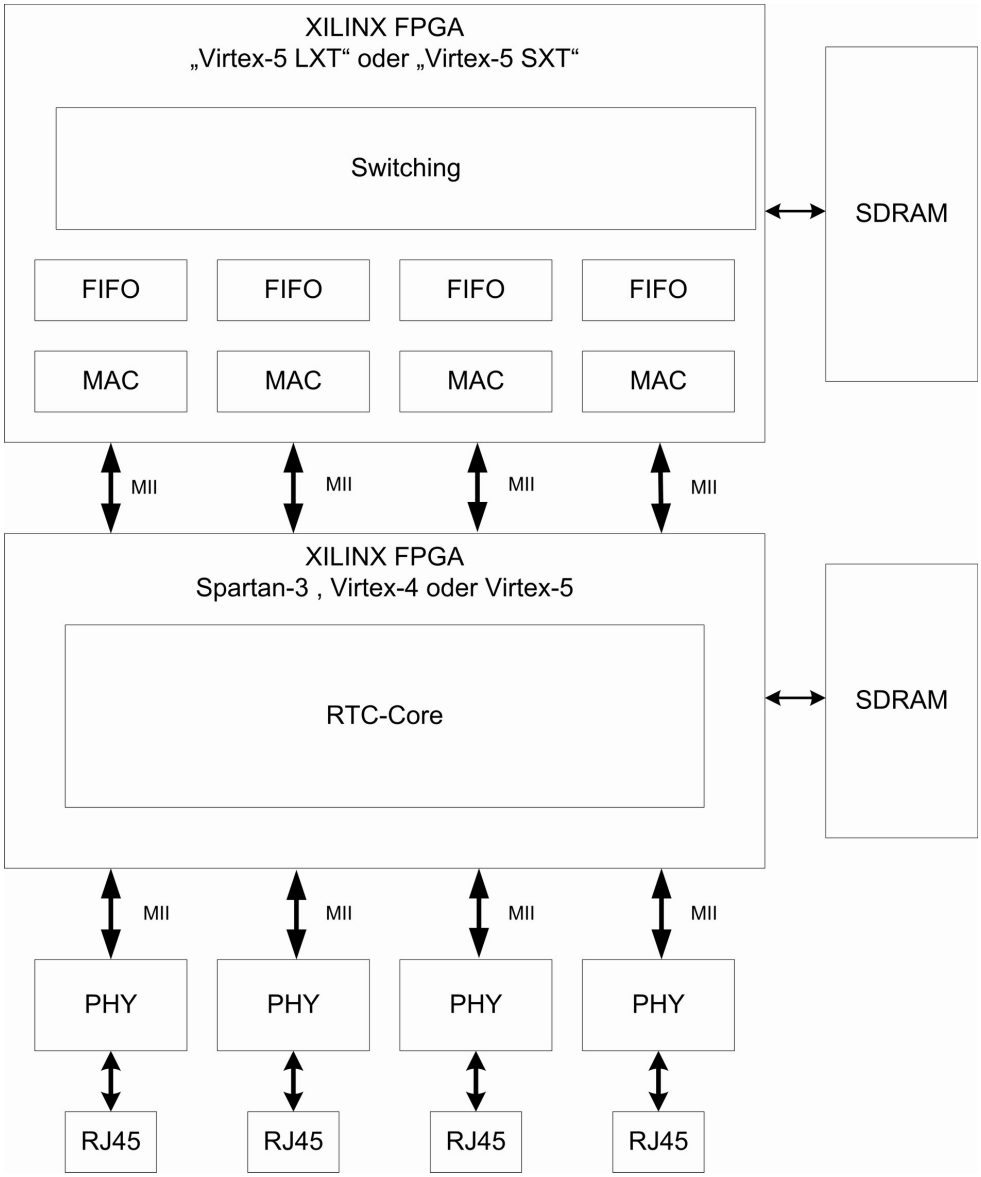


Abbildung 5.23: Blockschaltbild des modifizierten Switches [Hij07]

5.4 Zusammenfassung

Das Ziel dieses Kapitels bestand darin, die technische Umsetzung des in Kapitel 4 beschriebenen formalen Frameworks zu skizzieren.

Dazu wurde im ersten Teil gezeigt, auf welche Weise Netzwerke anhand der formalen Modellierung simuliert werden können. In diesem Zusammenhang wurde die Integration in den bestehenden Netzwerksimulator OMNeT++ ebenso diskutiert wie die Entwicklung eines eigenen Simulators. Beide diskutierten Lösungen basieren auf einem Softwareframework, welches die sequentielle Ausführung der einzelnen Algorithmen ermöglicht, um von einer gegebenen Netzwerkinfrastruktur und gegebenen IRT-Übertragungen zu einer Schedule zu gelangen.

Im Anschluss daran wurden Wege zur Implementierung der Schedule diskutiert. Kommen ausschließlich Hubs zum Einsatz, so ist lediglich ein zentrales Polling sinnvoll, wie es bei EPL durchgeführt wird. In diesem Fall können jedoch keine Geräte ohne Kenntnis der Schedule hinzugefügt werden, ohne die Echtzeitfähigkeit des Netzes durch ihre asynchronen Sendungen zu beeinträchtigen. Es handelt sich um geschlossene Subnetze mit Broadcastübertragungen, die ausschließlich über Gateways zugänglich sind.

Die zweite Implementierung sieht modifizierte Switches mit lokalen Schedules vor, wie sie bei ProfiNet IRT verwendet werden. Dort werden ausschließlich synchronisierte Geräte für den IRT-Betrieb verwendet; Geräte mit weichen Echtzeitanforderungen können über die VLAN-Priorisierung kommunizieren.

Zusätzlich zu diesen bekannten Ansätzen wurde als neuartiger Ansatz in diesem Kapitel der Einsatz von passiven Geräten diskutiert, welche lediglich auf Anforderung senden. Diese Anforderung sollte von den modifizierten Switches an die direkt angeschlossenen Geräte versendet werden. Auf diese Weise können sowohl aktiv sendende synchronisierte IRT-Geräte, als auch preisgünstigere passive Geräte inerhalb des echtzeitfähigen Netzes kommunizieren. Der modifizierte Switch leitet asynchron versendete Frames wie ein store-and-forward Switch Hop-by-Hop weiter. Das echtzeitfähige Netzwerk verhält sich dadurch für angeschlossene Standard-PCs oder Laptops transparent. Diese Geräte müssen keine Kenntnis von der Echtzeitfähigkeit des Netzes besitzen. Gleichzeitig können sie sogar isochron sendende Geräte asynchron abfragen. Diese asynchronen Anfragen werden unabhängig von den IRT-Frames weitergeleitet bis zum jeweiligen synchronisierten IRT-Gerät. Dieses kann die asynchrone Antwort verschicken, sobald es seine lokale Schedule zulässt.

Die in diesem Kapitel beschriebenen Möglichkeiten zur Realisierung der Schedule sind mit dem in Kapitel 4 vorgestellten formalen Modell umsetzbar und simulierbar, wenn das formale Modell in einen Netzwerksimulator integriert wird.

Die Einbindung der Schedules in einen zentralen Controller - bei EPL als Managing Node bezeichnet - ist durch die Verwendung eines Mikrocontrollers leicht realisierbar. Im dritten Teil dieses Kapitels wurde zusätzlich ein Ansatz zur Integration der Schedule in einen Switch vorgestellt. Es wurde herausgearbeitet, dass eine Integration zwischen den PHY- und MAC-Bausteinen am MII sinnvoll ist. Dazu wurde eine Real-Time Crossbar zwischen diese beiden Schichten eingefügt, welche PHY-Bausteine direkt miteinander koppelt und ankommende IRT-Frames ohne deren Analyse weiterleitet. Dabei wurden Probleme der Taktsynchronisation und der Regeneration der Präambel in einer VHDL-Simulation durch entsprechende Schaltungen gelöst [Hij07]. Die Verzögerungszeit eines solchen Switches liegt zwischen der Verzögerungszeit eines handelsüblichen Hubs - der jedoch keine gleichzeitige unabhängige Übertragungen zulässt - und einem proprietären ProfiNet-Switch. Eine zukünftige Betrachtung der vereinfachten RMII-Schnittstelle ist vielversprechend, da eine automatische Taktsynchronisierung vorgenommen wird und die Anzahl der Leitungen reduziert ist.

Kapitel 6

Zusammenfassung und Ausblick

Zu Beginn dieser Arbeit wurde der Trend zur vertikalen Integration erläutert, der die Durchsetzung des kostengünstigen und weit verbreiteten Ethernet-Standards von der Leitebene bis zur Feldebene in der Pyramide der Automatisierungstechnik begründet. Der Ethernet-Standard steht jedoch teilweise in Konflikt zu den Anforderungen einer automatisierten Anlage, so dass eine Integration der Ebenen nicht problemlos erfolgen kann. Das Ziel dieser Arbeit bestand darin, einen Beitrag zu dieser Integration zu leisten.

Dazu wurden in einem ersten Schritt die besonderen Anforderungen der Automatisierungstechnik und insbesondere der Antriebstechnik herausgearbeitet. Es wurde gezeigt, dass die höchsten Anforderungen aus der kaskadierten Positionsregelung von Antrieben stammen und aus einer isochronen Übertragung von echtzeitkritischen Daten bestehen. Bei einer hohen Übertragungshäufigkeit besitzen die Daten der härtesten vierten Echtzeitklasse nach IAONA nur einen geringen Nutzdatenanteil, der jedoch mit minimaler Verzögerung und Jitter deterministisch zu übertragen ist.

Teile des Ethernet-Standards stehen dabei mit den verwendeten Algorithmen und Protokollen in Kontrast zu diesen Anforderungen. Dies betrifft die Menge der Nutzdaten und insbesondere den best-effort Ansatz des Ethernets, der deterministische Aussagen über das Übertragungsverhalten der Ethernet-Frames nicht zulässt. Die üblichen Ethernet-Switches sind nicht in der Lage, Daten mit den Anforderungen der vierten Echtzeitklasse weiterzuleiten.

Um den geforderten Determinismus zu erlangen, werden die Konflikte von IRT-Übertragungen, durch die das nicht-deterministische Verhalten des Netzwerkes ausgelöst wird, mit Hilfe der Graphenfärbung zeitlich entzerrt, so dass Schedules für jeden Switch resultierten. Alternativ dazu kann eine globale Schedule für das Netzwerk erstellt werden. Dabei werden sowohl die Netzwerkinfrastruktur, als auch die Konflikte der IRT-Übertragungen als Graphen modelliert. Als Auslöser für Konflikte wurde das CSMA/CD-Verhalten im Halbduplexmodus sowie das Zwischenspeichern und ggf. sogar das Verwerfen von Frames im Vollduplexmodus des Ethernets identifiziert.

Vor Beginn der formalen Modellierung wurden zunächst gängige Lösungen für echtzeitfähiges Ethernet skizziert und klassifiziert. Die drei Felser-Klassen unterscheiden dabei, welche Protokollschicht im Vergleich zum herkömmlichen Ethernet modifiziert wird. Hier wurden drei Technologien herausgearbeitet, mit denen heutzutage hohe Echtzeitfähigkeit auf Ethernet-Basis erreicht werden kann.

- Die Lösungen der ersten Gruppe basieren auf Hubs. Diese erfüllen zwar die härteste IAONA-Echtzeitklasse, erzeugen jedoch geschlossene Subnetze und erlauben keine nebenläufige Kommunikation. Als Vertreter dieser Gruppe ist Ethernet PowerLink zu nennen.

- Zusätzlich dazu existieren Ansätze wie EtherCAT oder GinLink, welche zwar den höchsten Anforderungen genügen und mit gängigen Feldbussen konkurrieren können, jedoch eine für Ethernet untypische Art des physikalischen Zugriffs auf das Übertragungsmedium praktizieren.

- Lediglich ProfiNet erlaubt eine für Ethernet typische, auf Switches basierende Infrastruktur sowie eine typische Datenübertragung. ProfiNet IRT erfordert jedoch modifizierte Switches.

Prinzipiell kann auch eine (VLAN-)Priorisierung der Frames eingesetzt werden, um die Echtzeitfähigkeit des Netzwerkes zu erhöhen. Es wurde jedoch gezeigt, dass diese Maßnahme allein unzureichend ist, um die härteste Echtzeitklasse der IAONA zu erreichen (s. Kapitel 3.1.2.5). Aus diesem Grund wurde der Fokus dieser Arbeit auf die Bitübertragungsschicht sowie auf die Sicherungsschicht gelegt.

Die Analyse der existierenden Lösungen begründete die Notwendigkeit für ein formales Framework, welches von den etablierten Ansätzen abstrahiert und die Vorteile der bestehenden Ansätze verbindet. Das Ziel dabei war die Entwicklung eines effizienten Scheduling der IRT-Übertragungen.

Dieses allgemeine formale Framework wurde dann in Kapitel 4 entwickelt. Es umfasst baumförmige Netze, zunächst ausschließlich mit Switches als Verteiler, und basiert im ersten Schritt auf einem stark vereinfachten Modell der *idealisierten Halbduplexübertragung*. Als Ergebnis ist eine sequentielle Abarbeitung einer Folge von Algorithmen zu nennen, die letztlich zu der Schedule führt. Diese Folge besteht aus

1. der Berechnung der Kommunikationslinien im Netzwerk bei einer gegebenen Netzwerkinfrastruktur und bei gegebenen Echtzeitübertragungen.

2. der Bildung der Konfliktgraphen für jeden Switch.

3. der Färbung dieser Konfliktgraphen. Es wurde gezeigt, dass diese Färbung im Falle der idealisierten Halbduplexübertragung für jeden Switch unabhängig erfolgen kann. Die Färbung ist also parallelisierbar, wodurch die Komplexität der Berechnung sinkt. Dabei können entweder Färbungsheuristiken wie Greedy oder DSATUR, oder exakte Färbungsalgorithmen zum Einsatz kommen. Es wurde außerdem gezeigt, dass die resultierenden Schedules in polynomieller Zeit synchronisierbar sind. Dies bedeutet, dass die unabhängig berechneten Schedules zu einer Gesamtschedule zusammengefasst werden können und dass die Teil-Schedules, die jeder Switch besitzt, aufeinander abgestimmt werden können.

Als Ergebnis der Untersuchung der *idealisierten Vollduplexübertragung* ist zu nennen, dass lokale bipartite Kantenkonfliktgraphen entstehen, welche in polynomieller Zeit exakt färbbar sind. Die resultierenden lokalen Schedules sind jedoch nicht synchronisierbar. Um dieses Problem zu lösen, kann der globale Knotenkonfliktgraph betrachtet werden. Als

Alternative können zunächst alle lokalen Kantenkonfliktgraphen exakt gefärbt werden. Im Anschluss daran kann ein Switch, dessen Konfliktgraph die höchste Anzahl an Farben ergeben hat, als Wurzel-Switch der baumförmigen Netzwerkinfrastruktur verwendet werden. Verlaufen IRT-Übertragungen über mehrere Switches, so werden die Farben vom Wurzel-Switch aus übernommen, so dass synchrone Schedules entstehen. Nach der Übernahme der Farben werden die übrigen IRT-Übertragungen in den jeweiligen Konfliktgraphen neu gefärbt.

Im nächsten Schritt der formalen Modellierung wurde neben der Unicastkommunikation auch *Multicast- und Broadcastübertragung* berücksichtigt. Es hat sich gezeigt, dass die Einbindung von Broadcastübertragungen trivial ist, da jede Übertragung einen eigenen Zeitslot erfordert und keine konfliktfreie nebenläufige Kommunikation möglich ist. Multicastübertragungen hingegen erzeugen Kommunikationsbäume, die an einem Switch voneinander unabhängig verlaufen können, während sie an einem anderen Switch einen Konflikt auslösen. Dieser Fall kann lediglich durch die Betrachtung des globalen Knotenkonfliktgraphen behandelt werden.

Aufgrund der abstrakteren Sicht auf echtzeitfähige Ethernet-Lösungen wurde begründet, dass auch ein *gemischter Betrieb von Hubs und Switches* sinnvoll sein kann. Es kann also sinnvoll sein, State-of-the-Art Lösungen insbesondere im Umfeld der Antriebstechnik zu kombinieren. Dies gilt insbesondere an den Blättern der baumförmigen Netzwerkinfrastruktur. Dazu wurden Hubs in die formale Modellierung integriert. Dort sind die Vorteile der geringen Verzögerungszeit von Hubs und die Forderung der Automatisierungstechnik nach Querverkehr und der Nutzdatenverarbeitung im Durchlauf der Frames (s. EtherCAT und SERCOS) besonders ausgeprägt. Die Nachteile des Aufbaus von geschlossenen Subnetzen kommen weniger zum Vorschein, da die Antriebe der Anlage meist ohnehin nicht von außen zugänglich sind. Echtzeitfähige modifizierte Switches können hier den nahtlosen Übergang zu den Büro-Netzwerken auf der Leitebene schaffen. Die Schedules können auch hier durch die Betrachtung des globalen Knotenkonfliktgraphen erstellt werden.

Damit in Zusammenhang steht die Diskussion, wie *asynchrone Übertragungen* in die Schedule hinzugefügt werden können. Dies kann einerseits durch die Reservierung eines festen Zeitbereiches innerhalb der Schedule erfolgen. Zusätzlich dazu können asynchrone Frames auch Hop-by-Hop innerhalb der isochronen Phase übertragen werden, sofern sie nicht in Konflikt zu den isochronen Sendungen stehen. Dies bedingt jedoch eine Modifikation der Switches. Asynchrone Kommunikation bei Verwendung von Hubs bedingt, dass jedes asynchron sendende Gerät die durch einen Master vorgegebene Schedule einhält.

Ein weiterer Aspekt der Modellierung besteht darin, IRT-Frames *nicht in jedem Produktionszyklus* zu versenden. Als Lösungsansätze wurde eine Belegung der freien Zeitslots mit asynchroner Kommunikationen ebenso skizziert wie die alternative Ausführung mehrerer Echtzeitübertragungen in einem Zeitslot. Im Gegensatz zu einem Netzwerk aus Hubs, welches keine gleichzeitigen unabhängigen Übertragungen zulässt, wurde gezeigt, dass ein Multiplex-Betrieb von IRT-Frames in einem Netzwerk aus Switches problematisch ist.

Hier besteht ein Lösungsansatz darin, zunächst die Schedules der IRT-Übertragungen zu berechnen, die in jedem Zeitslot ausgeführt werden müssen. Im Anschluss daran können alle IRT-Übertragungen eingebunden werden, die in jedem zweiten Zeitslot ausgeführt werden sollen usw. Die Zeitslots können dann global vereinigt werden, indem die Schedules hintereinandergereiht werden.

Die Modellierung wurde durch die Betrachtung von *verschiedenen Framegrößen* weiter verfeinert. Als Ergebnis ist hier zu nennen, dass nach 2er-Potenzen gestaffelte Frames eine starke Vereinfachung bei der Bildung der Schedules darstellen. Frames beliebiger Länge können aufgrund der Tatsache, dass Konfliktgraphen die verschiedenen Übertragungsdauern nicht berücksichtigen, mit dem bisherigen Ansatz nicht berücksichtigt werden. An dieser Stelle ist jedoch die Verwendung des List-Schedulings erfolgversprechend, welches ursprünglich für die Betrachtung der Verzögerungszeiten der Switches untersucht wurde.

Bei der Betrachtung der *Verzögerungszeiten* zeigte sich zunächst, dass die addierten Verzögerungen die Zeitslots insbesondere bei der Verwendung von Switches in einem Maße vergrößern können, dass die Anforderungen der härtesten vierten IAONA-Echtzeitklasse nicht mehr erfüllbar sind. Der erste Ansatz zur Verringerung der Zykluszeit bestand darin, die Länge jedes Zeitslots auf die längste Kommunikation, welche sich in dem jeweiligen Slot befindet, zu begrenzen. Die dadurch erzielte Verkürzung der Zykluszeit ist jedoch nicht ausreichend. Zusätzlich dazu wurde die Vorverlegung von IRT-Übertragungen diskutiert, da Übertragungen in verschiedenen Zeitslots zwar grundsätlich in Konflikt stehen können, diese Konflikte jedoch unter Umständen aufgrund einer hohen Differenz der Verzögerungszeiten keinerlei Auswirkungen besitzen. Durch diese Vorgehensweise werden die Grenzen der Zeitslots innerhalb eines Zyklus aufgelöst. Ein andersartiger, vielversprechender Ansatz besteht in der Betrachtung des gesamten Netzwerkes und der Einführung eines List-Schedulings. Bei dieser Heuristik ist zunächst der globale Knotenkonfliktgraph hilfreich, der jedoch nicht gefärbt wird. Statt dessen werden die IRT-Übertragungen anhand der Anzahl ihrer Konflikte in die Schedule eingefügt. Die Konflikte werden dabei nicht anhand der Switches, sondern anhand der Kabel betrachtet. Wird eine weitere IRT-Übertragung hinzugefügt, so wird an jedem Kabel zunächst geprüft, ob dort ein Konflikt vorliegt. Falls ja, wird die IRT-Übertragung zeitlich so weit nach hinten verschoben, bis kein Konflikt mehr auftritt. Auf diese Weise können auch variable Framegrößen für echtzeitkritische Daten etabliert werden.

Im Verlaufe der Modellierung hat sich gezeigt, dass die Färbung lokaler Kantenkonfliktgraphen bei stark vereinfachter Modellierung sinnvoll ist. Die Betrachtung von globalen Knotenkonfliktgraphen liegt nahe, falls die lokalen Schedules nicht synchronisierbar sind und dennoch Standard-Algorithmen eingesetzt werden sollen. Alternativ dazu können existierende Färbealgorithmen angepasst werden. So können die bereits ermittelten Färbungen von Übertragungen für Untergraphen übernommen werden. Das vorgestellte List-Scheduling ist dabei ein vielversprechender Ansatz bei der Berücksichtigung von Verzögerungszeiten und variablen Framegrößen. Eine formale Analyse dieser Vorgehensweise mit Aussagen zur Güte der resultierenden Schedules wurde jedoch noch nicht durchgeführt. Zusätzlich dazu ist im Rahmen zukünftiger Forschungsarbeiten auch ein Vergleich mit existierenden Lösungen unter Berücksichtigung realer Anlagenparameter sinnvoll.

Neben der formalen Modellierung lag auch deren Anwendung im Fokus dieser Arbeit. Dazu wurde zunächst die Umsetzung des formalen Frameworks durch einen *Netzwerksimulator* skizziert. Dabei kann ein bereits existierender Simulator wie OMNeT++ erweitert werden. Vorgestellt wurde der Prototyp eines neuen Simulators, welcher die modulare Verbindung von Algorithmen zur Berechnung der Schedules einerseits und andererseits die Ausführung der Schedules in einem simulierten Netzwerk ermöglicht. Neben der Weiterentwicklung des Prototyps besteht ein weiterer Forschungsbedarf darin, typische Netzwerke

realitätsnaher automatisierter Anlagen zu untersuchen und mit verschiedenen Algorithmen zu kalkulieren. Für zentralisierte Anlagen ist zu erwarten, dass sich Lösungen mit Master/Slave-Ansatz als sinnvoller erweisen als dezentrale Schedules. Für stark verteilte Anlagen mit dezentraler Intelligenz werden hingegen dezentrale Schedules eine bessere Echtzeitfähigkeit garantieren können. Dabei wird die strikte Subnetzbildung aufgelöst.

Zusätzlich zur Simulation wurden Alternativen zur *technischen Durchsetzung der Schedules* beschrieben. Während Hubs IRT-Frames nur geringfügig verzögern, lassen sie jedoch keine gleichzeitigen unabhängigen IRT-Übertragungen zu. Dies führt zu einem geschlossenen Subnetz mit zentralem Scheduler. Diese Lösung entspricht den mit Ethernet Powerlink verwandten Ansätzen. Diese Schedules sind jedoch leicht zu berechnen, da lediglich alle Übertragungen hintereinander ausgeführt werden können. Um die Geräte zum Senden aufzufordern und zu synchronisieren, eignet sich das zentrale Polling. Während der Einsatz von dezentralen Schedulern als separat angeschlossene Geräte an jedem Switch problematisch ist (s. Kapitel 5.2.2), bietet die Modifikation von Switches eine weitere Möglichkeit zur Erlangung harter Echtzeitfähigkeit. Modifizierte Switches können über lokale Schedules verfügen und ihre passiven und damit kostengünstige Geräte dezentral pollen. Ein solcher Ansatz existiert bislang in keiner Realisierung, kann jedoch durchaus vielversprechend sein. Das aktive Senden von Übertragungen in Kombination mit modifizierten Switches ist ein Ansatz, der ähnlich zu Siemens ProfiNet ist. In diesem Falle leiten die Switches IRT-Frames entsprechend ihrer Schedule mit geringst möglicher Verzögerungszeit weiter.

Um die technische Realisierbarkeit genauer zu untersuchen, wurde als weiteres Ergebnis dieser Arbeit ein *Entwurf für einen echtzeitfähigen Switch* beschrieben, der eine ähnliche Funktionalität wie der proprietäre ProfiNet-ASIC besitzt. Das Ziel bestand darin, eine möglichst hohe Zahl an COTS-Komponenten zu verwenden. Das Resultat ist der in VHDL vorliegende Entwurf einer Real-Time Crossbar, welche zwischen der physikalischen und der MAC-Schicht zum Einsatz kommt. An dieser Stelle ist das Media Independent Interface etabliert, so dass eine standardisierte Schnittstelle verwendet werden kann in Kombination mit herkömmlichen PHY-Bausteinen und einer Switching Engines. Probleme der unvollständigen Präambel und der Synchronisation zwischen sendenden und empfangenen PHY-Bausteinen wurden gelöst und ein geeigneter FPGA synthetisiert und simuliert.

Die Real-Time Crossbar kann entweder mit einer modifizierten oder mit einer herkömmlichen Switching Engine betrieben werden. Während im ersten Fall unter Verwendung eines eigenen Protokolls eine Fragmentierung und Reassemblierung von asynchronen Frames ermöglicht wird, wird im zweiten Fall die Kompatibilität zum Ethernet-Standard erhöht. Dadurch entsteht jedoch das Problem, jeden MAC-Baustein an der Weiterleitung von Daten aus dem Sendepuffer zu hindern. Dies muss zu jedem Zeitpunkt geschehen, an dem der zugeordnete PHY-Baustein über die Real-Time Crossbar nicht mit dem MAC-Baustein verbunden ist. Ein Lösungsansatz ist hier, jeden Switch in den Halbduplexbetrieb zu versetzen, bei dem die Kontrolle der Sendung über das CRS-Signal erfolgen kann. Dadurch wird zwar nicht die gesamte Bandbreite genutzt, die Kompatibilität bleibt jedoch gewahrt. Ein weiterer Ansatz besteht darin, einen MAC-Baustein zu verwenden, der auch im Vollduplexmodus beim Senden das CRS-Signal interpretiert. Alternativ dazu kann auch ein vorhandener MAC-Baustein modifiziert werden, um auf das CRS-Signal entsprechend zu reagieren. Alle Lösungen variieren zwischen Kompatibilität zu Standard-Ethernet und erhöhter Echtzeitfähigkeit.

Eine unmittelbare Fortsetzung der Forschungsarbeit aus technischer Sicht kann darin bestehen, einen Prototyp eines solchen RTC-Switches zu erstellen und eine Switching Engine mit allen üblichen Switching-Protokollen auf der MAC-Schicht zu verwenden. Ohne eine geladene Schedule sollte sich dieser Switch genauso verhalten wie ein handelsüblicher Switch. Lädt man eine Schedule, so ist die Verzögerung der isochronen Frames konstant gering, kleiner als bei cut-through Switches und gleichzeitig unabhängig von asynchronem Datenverkehr. Zur Synchronisation der lokalen Schedules in kaskadierten Echtzeitswitches sollte eine IEEE 1588-Unterstützung integriert werden. An jeden dieser Echtzeitswitches sollten an jedem Port Standard-Switches zur Integration in die umgebene Netzwerkinfrastruktur angeschlossen werden können. Eine typische Anlagenkonfiguration kann dann als Testnetzwerk nachgebaut und mit gängigen Lösungen verglichen werden.

Zusammenfassend besteht der wissenschaftliche Beitrag dieser interdisziplinären Arbeit in erster Linie aus dem formal fundierten Framework, das in Kapitel 4 vorgestellt wurde. Dieses Framework ermöglicht erstmalig eine abstrakte, formalisierte Sichtweise auf echtzeitfähige Netzwerke der Automatisierungstechnik, die von einer konkreten technologischen Implementierung abstrahiert und einen Katalog von Verfahren von idealisierten Übertragungen und Netzwerken bis hin zu realitätsnahen Anwendungen bietet. Ausgehend von einer gegebenen Netzwerkinfrastruktur und IRT-Übertragungen können Schedules erstellt werden, mit denen ein deterministischer Medienzugang ermöglicht wird. Mit Hilfe dieser Verfahren kann in Abhängigkeit der konkreten Echtzeitanforderungen einer automatisierten Anlage ein individueller Kompromiss zwischen Kompatibilität zu Standard-Ethernet und Echtzeitfähigkeit sowie zwischen asynchronen und isochronen Übertragungen ermittelt werden.

Ein weiterer Beitrag dieser Arbeit besteht darin, dass sowohl die Simulierbarkeit der Verfahren (s. Kapitel 5.1), als auch Möglichkeiten zur Umsetzung der Schedules (s. Kapitel 5.2) und eine technische Realisierung im Rahmen eines modifizierten Ethernet-Switches (s. Kapitel 5.3) skizziert wurde. Obwohl in der weiteren Formalisierung des Frameworks[1], sowie in dem Ausbau der Simulation, der Anwendung des Frameworks auf industrierelevante Netzwerke und ebenso in der Weiterentwicklung des modifizierten Switches weiterer Forschungsbedarf besteht, lässt sich abschließend eine Aussage treffen:

Die Ethernet-Technologie ist im Umfeld der harten Echtzeitfähigkeit von automatisierten Anlagen anwendbar. Dazu wurde in dieser Arbeit ein durchgehendes Konzept mit einer Vielzahl von Stufen zwischen Kompatibilität zum Ethernet-Standard und harter Echtzeitfähigkeit vorgestellt.

[1] beispielsweise in der Untersuchung des List-Schedulings

Abbildungsverzeichnis

Literaturverzeichnis

[Acr05] Acromag Inc.: *Introduction to Modbus TCP/IP*; `http://www.acromag.com/pdf/intro_modbusTCP_765a.pdf` (29.11.07).

[Ada01] A. Adam: *Direkte Prozessdatenerfassung im Netzwerk*; in: Partner Info Qualität; Ausgabe 3/2001; `http://www.adam-software.de/news/press/piq3_01.shtml` (26.06.07).

[Aff06] C. Affolter: *SERCOS III*; FH Bern; Abteilung Elektrotechnik; Studienarbeit zu Industriellen Netzwerken: Embedded Control; 2006; `https://staff.hti.bfh.ch/uploads/media/SERCOS_III.pdf` (11.12.07).

[Agb07] A. B. G. Agbanzo: *Formale Modellierung und Implementierung von Algorithmen zur Graphenfärbung*; Universität Siegen, Fachbereich Mathematik; Bachelorarbeit; 2007.

[AHU88] A. V. Aho, J. E. Hopcroft, J. D. Ullman: *The Design and Analysis of Computer Algorithms*; Addison-Wesley Verlag; München; 1988; ISBN 0-2010-0029-6.

[Aic04] P. Aicher: *Winkelrekonstruktion und sensorlose Regelung von Hybridschrittmotoren*; TU München; Lehrstuhl für Feingerätebau und Mikrotechnik; Dissertation; 2004; `http://deposit.ddb.de/cgi-bin/dokserv?idn=974438928&dok_var=d1&dok_ext=pdf&filename=974438928.pdf` (03.12.07).

[Alo03] N. Alon: *A Simple Algorithm for Edge-Coloring Bipartite Multigraphs*; in: Information Processing Letters 85; Nr. 6; 2003; S. 301-302; ISSN 0020-0190; `http://dx.doi.org/10.1016/S0020-0190(02)00446-5`.

[And06] G. Anders: *Beitrag zur Verhaltensanalyse und Synchronisation von steuerungstechnischen Prozessen durch verteilte echtzeitfähige Kommunikationssysteme*; TU Bergakademie Freiberg; Fakultät für Maschinenbau, Verfahrens- und Energietechnik; Dissertation; 2006; `https://fridolin.tu-freiberg.de/archiv/pdf/MaschinenbauAndersGert72876.pdf` (02.07.07).

[Bec06] Beckhoff Automation AG: *EtherCAT in der elektronischen Reihenklemme*; in: polyscope 21/06; Binkert Medien Verlag; Laufenburg; S. 42-43; `http://www.beckhoff.com/download/press/2006/german/polyscope_21_2006.pdf` (12.12.07).

[BH06] A. Bormann, I. Hilgenkamp: *Industrielle Netze - Ethernet Kommunikation für Automatisierungsanwendungen*; 1. Auflage; Hüthig Verlag; Heidelberg; 2006; ISBN 3-7785-2950-1.

[BM94] A. Baginski, M. Müller: *INTERBUS-S: Grundlagen und Praxis*; Hüthig-Verlag; Heidelberg; 1994; ISBN 3-7785-2293-0.

[BR07] Bernecker + Rainer Industrie Elektronik Ges. m.b.H.: Perfection in Automation; 2007; `http://www.br-automation.com/cps/rde/xchg/br-automation_com/hs.xsl/index_DEU_HTML.htm` (03.12.07).

[Bre79] D. Brélaz: *New Methods to Color the Vertices of a Graph*; in: Communications of the ACM 22; Nr. 4; 1979; S. 251-256; ISSN 0001-0782; `http://dx.doi.org/10.1145/359094.359101` (15.11.07).

[Bri04] S. Brix: *Steuerungstechnik - SPS-Grundkurs*; 2004; `http://www.brix.de/elektrik/sps/sps-grundlagen.html` (12.12.07).

[Bro72] J. R. Brown: *Chromatic Scheduling and the Chromatic Number Problem*; in: Management Science 19; Nr. 4; 1972; S. 456-463.

[CDKFA02] B. Cain, S. Deering, I. Kouvelas, B. Fenner, A. Thyagarajan: *RFC 3376 - Internet Group Management Protocol, Version 3*; 2002; `http://tools.ietf.org/html/rfc3376` (01.07.07).

[CH94] J. Clark, D. A. Holton: *Graphentheorie - Grundlagen und Anwendungen*; Spektrum-Verlag; Heidelberg; 1994; ISBN 3-8602-5331-X.

[Chi99] T. Chiueh: *RETHER: A software-only Real-Time Ethernet for PLC Networks*; in: Proceedings of the Workshop on Embedded Systems on Workshop on Embedded Systems; `http://delivery.acm.org/10.1145/1270000/1267141/p6-chiueh.pdf?key1=1267141&key2=1837517911&coll=&dl=&CFID=15151515&CFTOKEN=6184618` (08.12.07).

[Cho00] D. D. Chowdhury: *High Speed LAN Technology Handbook*; Springer-Verlag; Berlin, Heidelberg; 2000; ISBN 3-5406-6597-8.

[Chr75] N. Christofides: *Graph Theory: An Algorithmic Approach*; Academic Press; 1975; ISBN 0-1217-4350-0.

[Cis07] Cisco Systems: *Understanding Rapid Spanning Tree Protocol (802.1w)*; White Paper; Document ID: 24062; 2007; `http://www.cisco.com/warp/public/473/146.html` (01.07.07).

[Con07] ControlNet International Ltd: *ControlNet*; 2007; *http://www.controlnet.org/* (29.11.07).

[COS01] R. Cole, R., K. Ost, S. Schirra: *Edge-Coloring Bipartite Multigraphs in O(E log D) Time*; in: Combinatorica 21; 2001; S. 5-12.

[Cro04] A. Crosswell: *Some experiments with IGMP snooping behavior of a couple of switches*; Columbia University; New York; 2004; `http://www.columbia.edu/~alan/igmp/` (01.07.07).

[CV94] T. Chiueh, C. Venkatramani: *Supporting Real-Time Traffic on Ethernet*; in: Proceedings of IEEE Real-Time Symposium; 1994; S. 282-286.

[Dec05] J.-D. Decotignie: *Ethernet-Based Real-Time and Industrial Communications*; in: Proceedings of the IEEE; Vol. 93; Issue 6; S. 1102-1117; 2005.

[Dec06] J.-D. Decotignie: *Real-Time Systems II - Real-Time Networking Ethernet*; Ecole Polytechnique Fèdèrale de Lausanne; CSEM Centre Suisse d'Electronique et de Microtechnique SA; Vorlesungsskript; 2006; `http://lamspeople.epfl.ch/decotignie/RTN_Ethernet.pdf` (07.12.07).

[Dee89] S. Deering: *RFC 1112 - Host Extensions for IP Multicasting*; 1989; `http://www.faqs.org/rfcs/rfc1112.html` (23.10.07).

[DH98] R. Dorne, J.-K. Hao: *Tabu Search for Graph Coloring, T-Colorings and Set T-Colorings*; in: Metaheuristics 98 - Theory and Applications; I. H. Osman et al. (Eds.); Kluver Academic Publishers; 1998; `http://citeseer.ist.psu.edu/dorne98tabu.html` (18.11.07).

[Die06] R. Diestel: *Graph Theory*; 3. Auflage; Springer-Verlag; Berlin, Heidelberg; 2006; ISBN 3-5402-6183-4.

[Din01] M. Dinkel: *PROFIdrive*; PROFIBUS International; Karlsruhe; `http://www.profibus.com/celummdb/doc/PROFIBUS/Downloads/Brochures/PROFIdrive-Brochure2001-d.pdf` (17.07.07).

[Dit90] E. Dittmar: *Grundlagen und Optimierung von Regelsystemen*; 4. Auflage; 1990.

[DoD01] Department of Defense: *Global Positioning System, Standard Positioning Service, Performance Standard*; Assistant Secretary of Defense for Command, Control, Communications, and Indelligence; 2001; `http://www.navcen.uscg.gov/gps/geninfo/2001SPSPerformanceStandardFINAL.pdf` (02.07.07).

[Doy04] P. Doyle: *Introduction to Real-Time Ethernet*; in: The Extention; Volume 5, Issue 4; 2004; `http://www.datalinkcom.net/Real%20Time%20Ethernet2.pdf` (08.12.07).

[Dut06] J. Dutine: *Entwurf und Implementierung eines Netzwerk-Management-Systems auf Basis von SNMP*; Universität Siegen; Diplomarbeit der Fachgruppe für Betriebssysteme und verteilte Systeme; 2006; `http://www.bs.informatik.uni-siegen.de/web/wismueller/arbeiten/2006_dutine.pdf` (21.10.07).

[DT97] J. Dittrich, U. von Thienen: *VLANs - Migration zu modernen Netzwerken*; 1. Auflage; ITP Verlag GmbH; Bonn; 1997; ISBN 3-8266-4031-4.

[DW06] F. Dopatka, R. Wismüller: *A Top-Down Approach for Realtime Industrial-Ethernet Networks using Edge-Coloring of Conflict-Multigraphs*; in: IEEE International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM); Taormina, Italien; 23.-26.05.2006; ISBN 1-4244-0193-3; S. 883-890.

[DW07] F. Dopatka, R. Wismüller: *Design of a Realtime Industrial-Ethernet Network Including Hot-Pluggable Asynchronous Devices*; in: IEEE International Symposium on Industrial Electronics (ISIE); Vigo, Spanien; 04.-07.06.2007; ISBN 1-4244-0755-9; S. 1826-1831.

[Eid06] John C. Eidson: *Measurement, Control, and Communication Using IEEE 1588*; Springer-Verlag; Berlin, Heidelberg; 2006; ISBN 1-8462-8250-0.

[EJ97] T. Erlebach, K. Jansen: *Call Scheduling in Trees, Rings and Meshes*; in: Proceedings of The Thirtieth Annual Hawwaii International Conference on System Sciences; ISBN 0-8186-7862-3; `http://csdl2.computer.org/comp/proceedings/hicss/1997/7734/01/7734010221.pdf` (01.01.08).

[Eng00] H. Engels: *CAN-Bus*; Franzis-Verlag; Poing; 2000; ISBN 3-7723-5145-X.

[Epp03] D. Eppstein: *Small Maximal Independet Sets and Faster Exact Graph Coloring*; in: Journal of Graph Algorithms and Applications 7; Nr. 2; 2003; S. 131-140.

[Erl98] T. Erlebach: *Scheduling Connections in Fast Networks*; TU München; Lehrstuhl für Effiziente Algorithmen; Dissertation; `http://www14.in.tum.de/personen/erlebach/dissertation.ps.gz` (11.11.07).

[Eth04] EtherCAT Technology Group: *EtherCAT - Technical Introduction and Overview*; Nürnberg; 2004; `http://www.automation.com/pdf_articles/EtherCAT_Introduction.pdf` (11.11.07).

[Eth07] Ethereal Inc.: *Ethereal - The world's most popular network protocol analyzer*; 2007; `http://www.ethereal.com/` (03.11.07).

[Eth07b] EtherCAT Technology Group: *EtherCAT - Ethernet for Control Automation Technology*; 2007; `http://www.ethercat.de/` (11.12.07).

[Fel00] M. Felser: *Ethernet als Feldbus? Kommunikationsmodelle für industrielle Netzwerke*; in: Infobit 3/2000; S. 21-24; `http://felser.ch/download/FE-TR-0005.PDF` (21.10.07).

[Fel02] M. Felser: *Vom Feldbus-Krieg zur Feldbus-Koexistenz*; Bulletin SEV/VSE; 9/2002; `http://felser.ch/download/FE-TR-0202.pdf` (07.07.07).

[Fel05] M. Felser: *Ethernet Switche für Motion-Control*; in: Automate.now! Jahrbuch der Automatisierungstechnik; 2005; `http://www.felser.ch/download/FE-TR-0505.pdf` (25.10.07).

[Fel05b] M. Felser: *Real-Time Ethernet — Industry Prospective*; in: Proceedings of the IEEE; Vol. 93, Nr. 6; 2005; S. 1118-1129; `http://www.felser.ch/download/FE-TR-0507.pdf` (13.06.06).

[Fel05c] M. Felser: *ProfiNet — Isochronous Real-Time*; Präsentation gehalten in IFAC Summer School - Control, Computing and Communication; Prag; 2005; `http://www.felser.ch/download/7_PN_IRT_E02.pdf` (09.12.07).

[FFMAV07] P. Ferrari, A. Flammini, D. Marioli, A. Taroni, F. Venturini: *Evaluation of timing characteristics of a prototype system based on PROFINET IO RT-Class 3*; in: IEEE Conference on Emerging Technologies and Factory Automation (EFTA); S. 1254-1261; 2007.

[FG05] C. Forman, A. Gron: *Vertical Integration and Information Technology Adoption: A Study of the Insurance Industry*; in: Proceedings of the 38th IEEE International Conference on System Sciences 2005; S. 1-10; 2005.

[FHKMS04] H. Flüs, D. Hübner, H. Kell, B. Moayeri, T. Simon: *Ethernet in Industrie-Umgebungen*; 2004; S. 2-80; `http://www.comconsult-research.de/bilder/EIU-Probe.pdf` (14.01.05).

[Fie07] Fieldbus Foundation: *Fieldbus Foundation*; 2007; `http://www.fieldbus.org` (29.11.07).

[Fra95] H. M. Frazier: *Media Independent Interface - Concepts and Guidelines*; in: Microelectronics Communications Technology Producing Quality Products Mobile and Portable Power Emerging Technologies (WESCON), Conf. record;S. 348-353; 1995.

[GI06] J. Gaisler, M. Isomaki: *GRETH 10/100 Mbit Ethernet MAC - Based on GRLIB-1.0.9*; Gaisler Reseach; 2006; `http://gaisler.com/doc/greth.pdf` (20.01.08).

[GM03] T. Gramann, D. S. Mohl: *Precision Time Protocol IEEE 1588 in der Praxis*; in: Elektronik 24/2003, WEKA Fachzeitschriften-Verlag; Poing; S. 86-94.

[Gör05] F.Göring: *Graphentheorie*; TU Chemnitz; Fakultät für Mathematik; 2005; `http://www.tu-chemnitz.de/mathematik/discrete/lehre/graph/w05/Vizing.pdf` (18.11.07).

[Gra69] R. L. Graham: *Bounds on Multiprocessing Timing Anomalies*; in: SIAM Journal on Applied Mathematics 17; Nr. 2; 1969; S. 416-429.

[Gru01] G. Gruhler (Hrsg.): *Feldbusse und Geräte-Kommunikationssysteme*; Franzis Verlag; Poing; 2001; ISBN 3-7723-5745-8.

[Guc96] M. Guckert: *Anschlußoptimierung in öffentlichen Verkehrsnetzen - Graphentheoretische Grundlagen, objektorientierte Modellierung und Implementierung*; Universität Marburg; Dissertation des Fachbereiches Mathematik; 1996.

[Har01] N. Hartmann: *Automation des Tests eingebetteter Systeme am Beispiel der Kraftfahrzeugelektronik*; Dissertation der Fakultät für Elektrotechnik und Informationstechnik; Universität Karlsruhe; `http://www.ubka.uni-karlsruhe.de/vvv/2001/elektrotechnik/5/5.text` (26.06.07).

[Har02] D. Harrington: *RFC3411 - An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks*; Request for Comments der IETF Network Working Group; 2002; `http://tools.ietf.org/html/rfc3411` (24.10.07).

[Hei98] M. Hein: *Ethernet - Fast Ethernet, Gigabit Ethernet*; 2. akt. und erw. Auflage; ITP Verlag; Bonn; 1998; ISBN 3-8266-4041-1.

[Hei04] T. J. Heistracher: *Interoperable Industry Networks*; Studie 3 des ESYCS/IINet Embedded Systems Cluster Salzburg; Durchgeführt an der Forschungsabteilung des Studiengangs für Telekommunikationstechnik und -Systeme; Salzburg; 2004; `http://esycs.salzburgresearch.at/doc/esycs-studie-iinet-final.pdf` (08.12.07).

[Hel98] G. Held: *Ethernet Networks - From 10Base-T to Gigabit*; 3. Auflage; Wiley-Verlag; New York; 1998; ISBN 0-4712-5310-3.

[Her05] T. Herman: *Time, Synchronization, and Wireless Sensor Networks - Part I*; University of Iowa; 2005; `http://www.cse.ohio-state.edu/~anish/788Notes/TimeSync-One.pdf` (02.07.07).

[HF03] P. Hofmann, M. Fuchs: *Entwurf Mechatronischer Systeme im Kraftfahrzeug*; TU Dresden; DaimlerChrysler Competence Center EE-Architektur; Skript zur Vorlesung; WS 2003/2004; `http://dccc.tu-dresden.de/Medien/Vorlesung0405/Begriffsbestimmung.pdf` (28.10.07).

[Hij07] S. Hijazi: *Erstellung eines hardwarenahen Systementwurfs zur Realisierung eines neuartigen, im Rahmen der Automatisierungstechnik echtzeitfähigen Ethernet-Switches*; Universität Siegen; Diplomarbeit der Fachgruppe für Betriebssysteme und verteilte Systeme; 2007.

[HJSM05] F. Hanssen, P. G. Jansen, H. Scholten, S. Mullende: *RTnet: A distributed Real-Time Protocol for Broadcast-Capable Networks*; in: Autonomic and Autonomous Systems and International Conference on Networking and Services (ICAS-ICNS); 2005; `http://doc.utwente.nl/57030/1/0000013c.pdf` (08.12.07).

[Hof79] H. Hoffmann: *Drehzahlregelung*; 1. Auflage; Vogel-Verlag; Würzburg; 1979; ISBN 3-8023-0122-6.

[Hop07] Hopf Elektronik GmbH: *DCF77 Funkuhr 6855 - Technische Beschreibung*; Version 11.00; 2007; `http://www.hopf-time.com/manuals/deutsch/pdf_6---/d6855_1100.pdf` (02.07.07).

[Huh05] A. Huhmann: *Ethernet, ein universeller Feldbus*; HARTING Deutschland GmbH & Co. KG; 2005; `http://www.harting.com/en/en/de/techinfo/tecnews/data/artikel/01432/index.de.html` (03.07.07).

[IC94] J. P. Ignizio, T. Cavalier: *Linear Programming*;Prentice-Hall; New York; 1994; ISBN 0-1318-3757-5.

[IDG03] IDG Business Media GmbH: *Hub, Router oder Switch: Grundlagen der Netzwerkkomponenten*; München; 2003; `http://www.channelpartner.de/technik/629982/index.html` (27.10.07).

[IEEE98] IEEE: *IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks*; 1998; ISBN 0-7381-1537-1; `http://standards.ieee.org/getieee802/download/802.1Q-1998.pdf` (01.07.07).

[IEEE02] IEEE: *IEEE 1588-2002: Standard for A Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*; 2002; `http://ieee1588.nist.gov/` (02.07.07).

[IEEE05] IEEE: *LAN/MAN CSMA/CD Access Method*; Institute of Electrical and Electronics Engineers; 2005; `http://standards.ieee.org/getieee802/802.3.html` (23.10.07).

[IEEE05b] IEEE: *Use of the IEEE Assigned EtherType Field with IEEE Std 802.3*; in: Local and Metropolitan Area Networks EtherType Field Tutorial Rev 0.7; 2005; `http://standards.ieee.org/regauth/ethertype/type-tut.html` (01.07.07).

[IEEE06] IEEE: *802.1Q - Virtual LANs*; 2006; `http://www.ieee802.org/1/pages/802.1Q.html` (23.10.07).

[Ind04] Indel AG: *GinLink Giga-Bit Feldbus - High-Speed Feldbus im industriellen Einsatz*; 2004; `http://www.indel.ch/ftp/Product_Info/Deutsch/GinLink.pdf` (21.10.07).

[Jan02] D. Janssen: *Ethernet-Kommunikation in Echtzeit - Echtzeit ohne Isolation*; in: IEE 47. Jahrgang 2002; Nr. S1; `http://dbindustrie.work.svhfi.de/AI/resources/e0aed1e95ba.pdf` (21.10.07).

[Jas02] J. Jasperneite: *Leistungsbewertung eines lokalen Netzwerkes mit Class-of-Service Unterstützung für die prozessnahe Echtzeitkommunikation*; Dissertation der Universität Magdeburg; Fakultät für Elektrotechnik und Informationstechnik; Shaker-Verlag; 2002; 3-8322-0832-1.

[JB03] D. Jansen, H. Büttner: *EtherCAT - Der Ethernet-Feldbus, Teil 1: Funktionsweise und Eigenschaften*; in: Elektronik 23/2003; WEKA Fachzeitschriften-Verlag; Poing; S. 64-72.

[JB03b] D. Jansen, H. Büttner: *EtherCAT - Der Ethernet-Feldbus, Teil 2: Adressierung und Aufbau der Protokolle*; in: Elektronik 25/2003; WEKA Fachzeitschriften-Verlag; Poing; S. 62-67.

[JB04] D. Jansen, H. Buttner: *Real-time ethernet the EtherCAT solution*; in: IET Computing and Control Engineering Journal; Vol. 15; Issue 1; S. 16-21; 2004.

[JDUGG74] D. S. Johnson, A. J. Demers, J. D. Ullman, M. R. Garey, R. L. Graham: *Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms*; in: SIAM Journal on Computing 3; Nr. 4; 1974; S. 299-325.

[Jet01] Jetter AG: *Inbetriebnahme-Software DRIVE.EXE für JetMove 600*; Ausgabe 09/2001; Art. 60864813; `http://www.jetter.de/download/Internet/deutsch/3_Antriebe/JetMove_6XX/JetMove%20600%20DRIVE%20Software.pdf` (16.07.07).

[Jet07] Jetter AG: *Jetter AG - Automation*; 2007; `http://www.jetter.de/` (29.11.07).

[Joh74] D. S. Johnson: *Worst Case Behaviour of Graph Coloring Algorithms*; in: Proceedings of the 5th South-Eastern Conference on Combinatorics, Graph Theory and Computing; Utilitas Mathematica Publishing; 1974; S. 513-527.

[Joh03] J. John: *ProfiNet Realtime Communication*; Profibus Nutzerorganisation; 2003; `http://www.ul.ie/~arc/Limerick_Realtime%20Communication_20.pdf` (09.12.07).

[JT00] K.-H. John, M. Tiegelkamp: *SPS-Programmierung mit IEC 61131-3*; 3. Auflage; Springer-Verlag; Berlin, Heidelberg; 2000; ISBN 3-5406-6445-9; `http://www.fen-net.de/karlheinz.john/IEC61131-3JohnTiegelkampDeutschV1.2.pdf` (02.11.07).

[Jun99] D. Jungnickel: *Graphs, Networks and Algorithms*; Springer-Verlag; Berlin, Heidelberg; 1999; ISBN 3-5406-3760-5.

[Kau02] F.-J. Kauffels: *Lokale Netze*; 14. Auflage; mitp-Verlag; Bonn; 2002; ISBN 3-8266-4092-5.

[KHT00] W. Kiesel, T. Heimbold, D. Telschow: *Bustechnologien für die Automation - Vernetzung, Auswahl und Anwendung von Kommunikationssystemen*; 2. überarb. Auflage; Hüthig-Verlag; Heidelberg; ISBN 3-7785-2778-9.

[KHW03] J. Kiszka, N. Hagge, B. Wagner: *RTnet – Eine Open-Source-Lösung zur Echtzeitkommunikation über Ethernet*; in: VDI-Berichte 1785 zur Telematik; 2003; Siegen; Juni 2003; S. 55-64; `http://www.rts.uni-hannover.de/images/9/9b/Kiszka03-Telematik.pdf` (08.12.07).

[Kis06] J. Kiszka: *RTnet - Hard Real-Time Networking for Real-Time Linux*; 2006; `http://www.rtnet.org/` (08.12.07).

[KJ85] M. Kubale, B. Jackowski: *A Generalized Implicit Enumeration Algorithm for Graph Coloring*; in: Communications of the ACM 28; Nr. 4; 1985; S. 412-418; ISSN 0001-0782; `http://dx.doi.org/10.1145/3341.3350` (18.11.07).

[Klu07] R. Kluger: *MES-Systeme: Produktivität steigt spürbar in Aluminium-Gießereien*; in: Elektrotechnik - Das Automatisierungstechnik-Portal; 2007; `http://www.elektrotechnik.vogel.de/fabrikinformationssysteme/articles/64870/` (01.07.07).

[KM99] W. R. Kriesel, O. W. Madelung (Hrsg.): *AS-Interface - Das Aktuator-Sensor-Interface für die Automation*; 2. überarb. und erw. Auflage; Hanser-Verlag; München, Wien; 1999; ISBN 3-4462-1064-4.

[Kön03] B. König-Ries: Universitaet Karlsruhe; Fakultaet fuer Informatik; Vorlesung Transaktionsverwaltung; SS 2003; `http://www.ipd.uni-karlsruhe.de/~koenig/TAV/Material/kapitel4.2a-4.pdf` (19.11.07).

[Kos05] S. Kostecke: *NTP Documentation Index*; 2005; `http://support.ntp.org/bin/view/Main/DocumentationIndex` (02.07.07).

[KR02] J. F. Kurose, K. W. Ross: *Computernetze - Ein Top-Down-Ansatz mit Schwerpunkt Internet*; Addison-Wesley Verlag; München; 2002; ISBN 3-8273-7017-5.

[KS04] J. Kiszka, R. Schwebel: *Alternative: RTnet - Hart echtzeitfähiges Ethernet jenseits von Herstellergrenzen*; in: A&D Newsletter; Oktober 2004; `http://www.rts.uni-hannover.de/rtnet/download/ad104705.pdf` (18.12.07).

[Kup06] M. Kupfer: *Entwicklung eines Optimierungsmodells zur Synchronisation des Luftverkehrs im CEATS Luftraum nach Kosten/Nutzen Gesichtspunkten*; TU Dresden; Institut für Verkehrssystemtechnik; Diplomarbeit; 2006; `http://www.eurocontrol.int/crds/gallery/content/public/thesis/6117-CRDSRNDUNV-KUP.pdf` (20.07.07).

[KWZB05] J. Kiszka, B. Wagner, Y. Zhang, J. Broenink: *RTnet – A Flexible Hard Real-Time Networking Framework*; in: 10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA); 2005; ISBN: 0-7803-9401-1; `http://www.rts.uni-hannover.de/rtnet/download/RTnet-ETFA05.pdf` (08.12.07).

[Lar05] L. H. Larsson: *Fourteen Industrial Ethernet solutions under the spotlight*; in: The industrial Ethernet book; Issue 42; 2005; `http://ethernet.industrial-networking.com/articles/articledisplay.asp?id=854` (29.11.07).

[Lei03] F. von Leitner: *Multicast*; Rohfassung eines c't-Zeitungsartikels; 2003; `http://www.fefe.de/ct/multicast.txt` (25.10.07).

[Lin06] Linum Software GmbH: *Was ist DCF77?*; 2006; `http://www.dcf77.com/index.htm` (02.07.07).

[LL05] A. Lüder, K. Lorentz (Hrsg.): *IAONA Handbook Industrial Ethernet*; Industrial Automation Open Networking Alliance e.V.; Magdeburg; 2005; ISBN 3-0001-6934-2.

[LG99] R. Lauber, P. Göhner: *Prozessautomatisierung 2*; Springer-Verlag; 1. Auflage; Berlin, Heidelberg; ISBN 3-5406-5319-8.

[LHO03] K.-P. Löhr, M. Haustein, K. Otto: *Verteilte Systeme*; FU Berlin; Institut für Informatik; Skript zur Vorlesung; WS03/04; `http://www.inf.fu-berlin.de/lehre/WS03/VS/vorlesung/vs1.pdf` (02.11.07).

[LP96] J. Lee, S. Park: *An error control scheme for Ethernet-based real-time communication*; in: Proceedings of the 3rd International Workshop Real-Time Computing Systems and Applications (RTCSA); S. 214-219; 1996.

[Lsi02] LSI Logic Corp.: *Technical Manual - L80227, 10 Base/T, 100Base TX Ethernet PHY*; 2002; `http://www.lsilogic.com/files/docs/techdocs/networking/80227.pdf` (19.01.08).

[Lut04] P. Lutz: *Next Generation SERCOS interface*; Vortrag auf dem FGCA Workshop „Harte Echtzeitkommunikation“; ZVEI; Frankfurt/Main; 14. Juni 2004.

[Mah03] N. P. Mahalik: *Fieldbus Technology*; Springer-Verlag; Berlin, Heidelberg; 2003; ISBN 3-5404-0183-0.

[Mat04] M. Matheson: *SynqNet - high performance motion control based on ethernet*; in: IET Computing and Control Engineering Journal; Vol. 15; Issue 5; S. 32-38; 2004.

[Mei05] W. Meier: *Betriebssysteme - Prinzipien konzentrierter und verteilter Systemsoftware*; FH Kaiserslautern; Fachbereich Informatik und Mikrosystemtechnik; Skript zur Vorlesung; `http://mozart.informatik.fh-kl.de/download/Lehre/SS2005/Betriebssysteme/Vorlesung/bs.article.pdf` (02.11.07).

[Mer07] M. Merz: *Alternative Algorithmen für das Scheduling in Echtzeitnetzwerken*; Universität Siegen, Fachbereich Elektrotechnik und Informatik; Studienarbeit; 2007.

[Mey87] M. Meyer: *Elektrische Antriebstechnik: Stromrichtergespeiste Gleichstrommaschinen und voll umrichtergespeiste Drehstrommaschinen*; Band 2; Springer-Verlag; Berlin, Heidelberg; 1987; ISBN 3-5401-7022-7.

[Mey04] H. Meyer: *Synergie aus Feldbus und Ethernet*; in: SPS-Magazin; Ausgabe 10/2004; S. 97-99; `www.softing.com/home/de/pdf/ia/professional-article/foundation-fieldbus/2004/0410_sps_industrial_ethernet.pdf` (29.11.07).

[Mil85] D. L. Mills: *Network Time Protocol (NTP)*; Request for Comments 958; M/A-COM Linkabit; 1985; `ftp://ftp.isi.edu/in-notes/rfc958.txt` (02.072007).

[Mil89] D. L. Mills: *Network Time Protocol (Version 2) - Specification and Implementation*; Request for Comments 1119; Universität Delaware, 1989; `ftp://ftp.isi.edu/in-notes/rfc1119.ps` (02.072007).

[Mil89b] D. L. Mills: *Internet Time Synchronization: the Network Time Protocol*; Request for Comments 1129; M/A-COM Linkabit; 1989; `ftp://ftp.isi.edu/in-notes/rfc1129.pdf` (02.072007).

[Mil03] D. L. Mills: *A brief history of NTP time: confessions of an Internet timekeeper*; in: ACM Computer Communications Review 33, 2 (April 2003); S. 9-22; `http://www.eecis.udel.edu/~mills/database/papers/history.pdf` (02.072007).

[Mil06] David L. Mills: *Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI*; Request for Comments 4330; University of Delaware, 2006; `ftp://ftp.isi.edu/in-notes/rfc4330.txt` (02.072007).

[MK04] C. F. Manga, H. B. Keller: *Kopplung verteilter Feldbus-Segmente über Ethernet*; in: Elektronik; 2004; Nr. 23; `http://www2.elektroniknet.de/topics/kommunikation/fachthemen/2005/0001/index.htm` (01.072007).

[ML06] A. Moning, R. Lanz: *Datenkommunikation und Rechnernetze*; 2006; FH Bern; Skript zur Vorlesung; `http://www.sws.bfh.ch/~lanz/SKRIPTE/DataKomm.pdf` (01.07.07).

[MMI72] D. W. Matula, G. Marble, J. D. Isaacson: *Graph Coloring Algorithms*; in: R. C. Read: Graph Theory and Computing; Academic Press; New York; 1972; S. 109-122.

[MPDV07] MPDV Mikrolab GmbH: *HYDRA-Leitstand*; 2007; `http://www.mpdv.de/de/ftp/products/Leitstand.pdf` (01.07.07).

[Mül94] G. Müller: *Grundlagen elektrische Maschinen*; VCH Verlag; Weinheim; 1994; ISBN 3-527-28390-0.

[Nat06] National Semiconductor: *DP83848 Single 10/100 Mb/s Ethernet Transceiver Reduced Media Independent Interface (RMII) Mode*; 2006; `http://www.national.com/an/AN/AN-1405.pdf` (19.01.08).

[NK90] T. Nishizeki, K. Kashiwagi: *On the 1.1 Edge-Coloring of Multigraphs*; in: SIAM Journal on Discrete Mathematics 3; Nr. 3; 1990; S. 391-410; ISSN 0895-4801; `http://dx.doi.org/10.1137/0403035` (19.11.07).

[Nov05] Novelis Deutschland GmbH: *Schaubild eines Walzwerkes*; 2005; `http://www.novelis-goettingen.com/res/pics/zmiutfeffb1104411369_b.jpg` (01.07.07).

[Now06] U. Nowak: *Graphentheoretische Modellierung eines automatisierungstechnischen Echtzeitnetzwerks und Algorithmenentwurf zum Kommunikations-Scheduling*; Universität Siegen; Fachbereich Mathematik; Diplomarbeit; `http://www.uwenowak.de/arbeiten/diplomarbeit.pdf` (11.11.07).

[ODVA07] Open DeviceNet Vendors Association: *EtherNet/IP*; 2007; `http://www.ethernetip.de/` (29.11.07).

[Omn08] OMNeT++ Community: *OMNeT++ - Discrete Event Simulation System*; 2008; `http://www.omnetpp.org/` (23.01.08).

[Ost05] J. Ost: *CIP Safety on Sercos*; in: SPS-Magazin; 2005; `http://www.sps-magazin.de/index.html?/artikel/artikel.asp&DJVWm0zBTgeYcPUgfapVrejRr_I` (14.12.07).

[Par02] S. G. Park: *Fieldbus in IEC61158 Standard*; in: Proceedings on the 15th CISL Winter Workshop; Kushu, Japan; 2002; `http://icat.snu.ac.kr:3333/ww/pdf/ww_2002_8.pdf` (29.11.07).

[PD04] L. L. Peterson, B. S. Davie: *Computernetze - Eine systemorientierte Einführung*; deutsche Ausgabe der 3. amerik. Auflage; dpunkt-Verlag; Heidelberg; 2004; ISBN 3-8986-4242-9.

[Pee83] J. Peemöler: *Correction to Brelaz's Modification of Brown's Coloring Algorithm*; in: Communications of the ACM 26; Nr. 8; 1983; S. 595-597; ISSN 0001-0782; `http://dx.doi.org/10.1145/358161.358171` (15.11.07).

[Pem06] Pemag Engineering: *Dynamische Berechnungen*; 2006; `http://www.pamag.ch/pamag/cms/konstruktion/dynamische_berechnungen.html` (17.07.07).

[PH06] S. H. Prochnow, R. von Hanxleden: *Echtzeit-Betriebssysteme und - Bussysteme*; Christian-Albrechts-Universität zu Kiel; Institut für Informatik und Praktische Mathematik; ; Seminar Echtzeitsysteme und Eingebettete Systeme WS 2006/07; `http://www.informatik.uni-kiel.de/inf/von-Hanxleden/teaching/ws06-07/s-rt/proceedings.pdf` (03.12.07).

[Pho98] Phoenix Contact (Hrsg.): *Grundkurs Sensor/Aktor-Feldbustechnik*; 2. überarb. Auflage; Vogel Buchverlag; Würzburg; 1998; ISBN 3-8023-1764-5.

[Pla05] J. Plate: *Grundlagen Computernetze*; FH München; Institut für Elektrotechnik und Informationstechnik; Skript zur Vorlesung; 2005; `http://www.netzmafia.de/skripten/netze/netz4.html` (01.07.07).

[PM06] R. Pigan, M. Metter: *Automating with ProfiNet - Industrial Communication based on Industrial Ethernet*; Publics Corporate Publishing Verlag; Erlangen; 2006; ISBN 3-89578-256-4.

[PN04] A. Poschraann, P. Neumann: *Architecture and model of Profinet IO*, in: IEEE 7th AFRICON Conference in Africa; Vol. 2; S. 1213-1218; 2004.

[Pne99] P-NET User Organization: *Der P-NET Feldbus für die Prozeßautomation*; 1999; `http://www.infoside.de/infida/pnet/p-net.htm` (29.11.07).

[Pol06] Politecnico di Milano: *RTAI - the RealTime Application Interface for Linux*; 2006; `https://www.rtai.org/` (08.12.07).

[Pop00] M. Popp: *Profibus-DP/DPV1 - Grundlagen, Tipps und Tricks für Anwender*; 2. überarb. Auflage; Hüthig-Verlag; Heidelberg; 2000; ISBN 3-7785-2781-9.

[Pop05] M. Popp: *Das Profinet IO-Buch - Grundlagen und Tipps für Anwender*; Hüthig-Verlag; Heidelberg; 2005; ISBN 3-7785-2966-8.

[Pro02] Profibus Nutzerorganisation e.V. (Hrsg.): *Kurzbeschreibung von Profibus*; Karlsruhe; 2002; `http://www.dke.de/nr/rdonlyres/ff9c5307-0bc1-4d5d-861d-503b58a07685/9191/typ3profibus.pdf` (28.10.07).

[Pro07] Profibus Nutzerorganisation e.V. (Hrsg.): *ERTEC 200 Enhanced Real-Time Ethernet Controller*; Version 1.1.0; 09.07.2007; Siemens-Bestellnr. 6GK1 182-0BB00-0AA; Beitrags-ID 23234691.

[Pro07b] Profibus Nutzerorganisation e.V. (Hrsg.): *ERTEC 400 Enhanced Real-Time Ethernet Controller*; Version 1.2.0; 09.07.2007;Siemens-Bestellnr. 6GK1 184-0BB00-0AA; Beitrags-ID 21631481.

[PW04] M. Popp, K. Weber: *Der Schnelleinstieg in PROFINET*; PROFIBUS Nutzerorganisation e.V.; 2004; PNO-Bestell-Nr.: 4.181.

[Reg05] B. Rega: *Arbeitsweise und Genauigkeitsvergleich DCF77 und GPS Zeitempfänger*; hopf Elektronik GmbH; 2005; `http://www.hopf-time.com/de/dcf-gps.htm` (27.10.07).

[Rei02] B. Reißenweber: *Feldbusse zur industriellen Kommunikation*; 2. Auflage; Oldenbourg-Verlag; München; 2002; ISBN 3-4862-7033-8.

[RMI98] RMII Consortium Members: *RMII Specification*; 1998; `http://www.national.com/appinfo/networks/files/rmii_1_2.pdf` (19.01.08).

[Ren06] J. Renner: *Mobile Agenten für den Fernzugriff auf eingebettete Systeme*; Dissertation der Fakultät für Elektrotechnik und Informationstechnik; TU Chemnitz; 2006; `http://archiv.tu-chemnitz.de/pub/2006/0109/data/Dissertation_Web.pdf` (26.06.07).

[Sch03] G. Schnell (Hrsg.): *Bussysteme in der Automatisierungs- und Prozesstechnik*; 5. Auflage; Vieweg Verlag; Braunschweig, Wiesbaden; 2003; ISBN 3-5284-6569-7.

[Sch03b] G. Schnell (Hrsg.): *Bussysteme in der Automatisierungs- und Prozesstechnik*; 5. Auflage; Vieweg Verlag, Braunschweig, Wiesbaden; 2003; ISBN 3-5284-6569-7.

[Sch03c] H. Scheitlin: *In Echtzeit-Ethernet durch die Fabrikation*; in: IAONA Switzerland SE STZ Automatisierungshandbuch, S. 22-29; 2003; `https://home.zhwin.ch/~sln/Industrial_Ethernet/22-29_SE%20Automate%20now%2003.pdf` (07.07.07).

[Sch03d] H. Scheitin: *Ethernet Powerlink: kurz und knapp*; Zürcher Hochschule Winterthur; Fachbeitrag auf der Internetseite der EPSG; 2003; `http://www.ethernet-powerlink.org/index.php?type=tree&id=2&docaction=docvd&docid=32` (27.04.05).

[Sch04] H. Scheitlin: *Ethernet Powerlink - klipp und klar*; in: IAONA Switzerland SE STZ Automate Now, S. 108–111, 2004.

[Sch04b] J. Schwager: *Ethernet erreicht das Feld - Sechs Echtzeit-Varianten im Vergleich, Teil 1*; in: Elektronik 11/2004; WEKA Fachzeitschriften-Verlag; Poing; S. 48-54; 2004; `http://www-pdv.fh-reutlingen.de/rte/schwager2004-1.pdf` (07.07.07).

[Sch05] G. Schröder: *Industrielle Kommunikation*; Universität Siegen; Institut für Leistungselektronik und Elektrische Antriebe; Skript zur Vorlesung; 2005.

[Sch07] W. Schmitt: *Lokale Netze*; FH Gießen-Friedberg; Institut für Informatik; Skript zur Vorlesung; 2007; `http://homepages.fh-giessen.de/~hg6421/RN&A/RN&A_LokaleNetze.pdf` (01.07.07).

[Sch07b] S. Schöling, K. Bernshausen, J. Brennscheidt, P. Gibson, C. Igel, J. Kammer, M. Mielke, S. Poniewas, K. Sulieman: *NetSim und GSM - Ein Netzwerksimulator in Java und C++*; Universität Siegen; Projektgruppenarbeit der Fachgruppe für Betriebssysteme und verteilte Systeme; 2007.

[SDK83] M. Syslo, N. Deo, J. S. Kowalik: *Discrete Optimization Algorithms with Pascal Programs*; Prentice Hall Professional Technical Reference; New York; 1983; ISBN 0-1321-5509-5.

[Ser04] SERCOS International e.V.: *SERCOS-III - Dritte Generation SERCOS interface*; Version 1.3.4; 2004; `http://www.sercos.de/pdf/SERCOS-III_V134d_public.pdf` (11.12.07).

[Ser06] SERCOS International e.V.: *SERCOS goes Ethernet*; News 1-2006; `http://www.boschrexroth.com/business_units/brc/de/events_de/SERCOS-III_Workshop/sercos-news.pdf` (11.12.07).

[Ser07] SERCOS International e.V.: *SERCOS goes Automation*; 2007; `http://www.sercos.de/` (10.12.07).

[Sik03] A. Sikora: *Echtzeit übers Ethernet - Synchronisation mit dem IEEE-1588-Standard*; in: Elektronik 9/2003; WEKA Fachzeitschriften-Verlag; Poing; S. 46-51.

[SJHH02] H. Scholten, P. G. Jansen, F. Hanssen, T. Hattink: *RTnet, a new approach to in-home real-time multimedia communication*; 2002; `http://www.croky.net/publications/PDFs/scholten-2002-01.pdf` (08.12.07).

[SL07] A. Schemberg, M. Linten: *PC-Netzwerke*; 4. akt. Auflage; Galileo Press; Bonn; ISBN 3-8362-1062-1

[SL07b] S. Soucek, D. Loy: *Vertical Integration in Building Automation Systems*; in: Proceedings of the 5th IEEE International Conference on Industrial Informatics; Vol. 1; S. 81-86; 2007.

[Sof05] Softing AG: *FG-100 FF to HSE Linking Device*; Product Information; 2005; `http://www.softing.com/home/en/pdf/ia/product-info/foundation-fieldbus/D_IA_16E_0509_FG-100_FF_Z.pdf` (29.11.07).

[Som04] E. Sommer: *Profibus*; FH München; Fakultät Elektrotechnik und Informationstechnik; Skript zur Vorlesung; `http://www.e-technik.fh-muenchen.de/fb/lab/lsa/aut/docs/bus/profibus.pdf` (07.07.07).

[SS85] Y. Shimokawa, Y. Shiobara: *Real-time Ethernet for industrial applications*; in: Proceedings of the international Conference on Industrial Electronics, Control and Instrumentation (IECON'85); San Francisco; S. 829-834; 1985.

[Sta99] W. Stallings: *Gigabit Ethernet*; in: Cisco Systems: The Internet Protocol Journal; Volume 2, No. 3; 1999; `http://www.cisco.com/web/about/ac123/ac147/ac174/ac199/about_cisco_ipj_archive_article09186a00800c85a6.html`, (27.10.07).

[Ste04] W. R. Stevens: *TCP/IP: Der Klassiker - Protokollanalysen, Aufgaben und Lösungen*; 1. Auflage; Hüthig-Verlag; Bonn; 2004; ISBN 3-8266-5042-5.

[Ste05] N. Steinig: *Satz von Brooks*; Universität Bielefeld; Fachbereich Mathematik; 2005; `http://www.math.uni-bielefeld.de/~nsteinig/satz%20von%20brooks.pdf` (18.11.07).

[SV07] L. Seno, S. Vitturi: *A Simulation Study of Ethernet Powerlink Networks*; in: IEEE Conference on Emerging Technologies and Factory Automation (EFTA); S. 740-743; 2007.

[SW06] G. Schnell (Hrsg.), B. Wiedemann (Hrsg.): *Bussysteme in der Automatisierungs- und Prozesstechnik - Grundlagen, Systeme und Trends der industriellen Kommunikation*; 6. Auflage; Vieweg Verlag; Braunschweig, Wiesbaden; 2006; ISBN 3-8348-0045-7.

[Syn07] SynqNet User Group: *SynqNet. Fast. Save. Proven.*; 2007; `http://www.synqnet.org/` (12.12.07).

[SZ03] H. Scheitlin, R. Zuber: *Kommunizieren Sie pünktlich?*; 2003; `http://www.edison.ch/uploads/media/Kommunizieren_Sie_puenktlich.pdf` (03.07.07).

[Tan03] A. S. Tanenbaum: *Computernetzwerke*; 4. überarb. Auflage; Pearson Verlag; München; 2003; ISBN 3-8273-7046-9.

[Ter05] Teridian Semiconductor Corp.: *78Q2123-DB MicroPHY MII Evaluation Board*; Rev. 2.0; 2005; `http://www.topas-electronic.net/marketing/teridian2006_01/Teridian-78Q21x3-User-Manual.pdf` (20.01.08).

[Tos06] Toshiba Corp.: *TCnet*; 2006; `http://www3.toshiba.co.jp/sic/english/seigyo/tcnet/technology.htm` (11.06.06).

[Tro05] Trolltech Labs: *Qt Reference Documentation (Open Source Edition)*; 2008; `http://doc.trolltech.com/4.0/index.html` (23.01.08).

[Tur04] V. Turau: *Algorithmische Graphentheorie*; 2. Auflage; Addison-Wesley Verlag; München; 2004; ISBN 3-4862-0038-0.

[UB05] U. Ufuktepe, G. Bacak: *Applications of Graph Coloring*; in: Lecture Notes in Computer Science; Springer-Verlag; Berlin, Heidelberg; Volume 3482/2005; DOI 10.1007/11424857_55; 2005; S. 522-528; `http://www.springerlink.com/content/qaaantfkp5n3926a/` (19.07.07).

[Uni06] Unitechnik AG: *Maschinentechnik*; `http://www.unitechnik.de/downloads/prospekte/maschinentechnik.pdf` (21.10.07).

[Var04] J.-U. Varchmin: *Industrielle Kommunikation mit Feldbussen*; TU Braunschweig; Institut für Elektrische Messtechnik; Skript zur Vorlesung; 2004; `http://www.emg.ing.tu-bs.de/pdf/IKF/Feldbusse_Skript_SS04.pdf` (21.10.07).

[Ver07] Verein Deutscher Werkzeugmaschinenfabriken: *VDW*; 2007; `http://www.vdw.de/` (10.12.07).

[Vol91] L. Volkmann: *Eine Einführung in die Graphentheorie*; Springer-Verlag; Wien, New York; 1991; ISBN 3-2118-2267-4.

[Vog98] J. Vogel: *Elektrische Antriebstechnik*; 6. vollst. überarb. Auflage; Hüthig-Verlag; Heidelberg; 1998; ISBN 3-7785-2649-9.

[VP06] I. Verhappen, A. Pereira: *Foundation Fieldbus*; ISA Verlag; 2006; ISBN 1-5561-7964-2.

[VVTG06] R. Viegas, R.A.M. Valentim, D.G. Texeira 1, L.A. Guedes: *Analysis of Protocols to Ethernet Automation Networks*, in: Proceedings of the SICE-ICASE International Joint Conference 2006; S. 4981-4985.

[Wei06] H. Weibel: *IEEE 1588 Tutorial*; Vortrag im Rahmen der Conference on IEEE 1588, National Institute of Standards and Technology; Gaithersburg; 2006; `http://ines.zhwin.ch/uploads/media/2006_Conference_IEEE_1588_Tutorial.pdf` (02.07.07).

[Wen03] M. Wenk: *Isochrones Realtime Ethernet - PNO zündet dritte Stufe*; in: IEE; 48. Jahrgang; 2003; Nr. 03; S. 72-75; `http://www.all-electronics.de/ai/resources/24dabaa32b9.pdf` (16.07.07).

[Wes06] H. Westerholt: *Entwurf und Entwicklung eines Simulationsmodells für ein neuartiges Echtzeit-Ethernet System im Industrieeinsatz*; Universität Siegen; Diplomarbeit der Fachgruppe für Betriebssysteme und verteilte Systeme; 2006; `http://www.bs.informatik.uni-siegen.de/web/dopatka/arbeiten/2006_westerholt_d2.pdf` (21.10.07).

[Wig83] A. Wigderson: *Improving the Performance Guarantee for Approximate Graph Coloring*; in: Journal of the ACM 30; Nr. 4; 1983; S. 729-735; ISSN 0004-5411; `http://dx.doi.org/10.1145/2157.2158` (18.11.07).

[Wis06] R. Wismüller: *Rechnernetze 2*; Universität Siegen; Fachbereich für Elektrotechnik und Informatik; Skript zur Vorlesung; 4. Kapitel; `http://www.bs.informatik.uni-siegen.de/web/wismueller/vl/ws06/rn2/v04_4.pdf` (07.07.07).

[Wis07] R. Wismüller: *Rechnernetze 1*; Universität Siegen; Fachbereich für Elektrotechnik und Informatik; Skript zur Vorlesung; `http://www.bs.informatik.uni-siegen.de/web/wismueller/vl/ss07/rn1/v05.pdf` (01.07.07).

[Wis07b] R. Wismüller: *Rechnernetze 1*; Universität Siegen; Fachbereich für Elektrotechnik und Informatik; Skript zur Vorlesung; `http://www.bs.informatik.uni-siegen.de/web/wismueller/vl/ss07/rn1/v09.pdf` (01.07.07).

[WP67] D. J. A. Welsch, M. B. Powell: *An upper Bound for the Chromatic Number of a Graph and its Application to Timetabling*; in: The Computer Journal; Oxford Journals; 1967; S. 85-86; `http://comjnl.oxfordjournals.org/cgi/reprint/10/1/85.pdf` (19.11.07).

[Xil07] Xilinx, Inc.: *Spartan-3E Data Sheets*; 2007; `http://www.xilinx.com/support/documentation/spartan-3e.htm#19564`(22.01.08).

[Xil07b] Xilinx, Inc.: *Virtex-5 Data Sheets*; 2007; `http://www.xilinx.com/support/documentation/virtex-5.htm#19294`(22.01.08).

[Xu03] G. Xu: *GPS. Theory, Algorithms and Applications*; Springer Verlag; Berlin; 2003; ISBN 3-5406-7812-3.

[Zen99] A. Zenk: *Lokale Netze - Die Technik fürs 21. Jahrhundert*; Addison-Wesley Verlag; München; 1999; ISBN 3-8273-1592-1.

[Zha02] Q. Zhang: *Konzepte für die Kommunikation zwischen Automatisierungsgeräten*; Dissertation der Fakultät für Elektrotechnik und Informatik; TU Berlin; 2002; `http://edocs.tu-berlin.de/diss/2002/zhang_qimin.pdf` (21.10.07).

[Zin07] U. Zinsser: *Ethernet-Transceiver mit IEEE-1588-PTP-Hardware-Support*; in: elektronik.net; 15.102007; WEKA Fachzeitschriften-Verlag; Poing; `http://www.elektroniknet.de/home/bauelemente/produkte/uebersicht/aktive-bauelemente/kommunikations-ics/p/d/ethernet-transceiver-mit-ieee-1588-ptp-hardware-su-1/` (27.10.07).

[Zve07] Zentralverband Elektrotechnik- und Elektronikindustrie e.V.: *Zvei*; 2007; `http://www.zvei.org/` (10.12.07).

Abkürzungsverzeichnis

$[\alpha]_R$	R-Äquivalenzklasse von α	56
α	Schedule	136
$\alpha(r)$	Zeitpunkt des Beginns einer Übertragung	136
$\chi'(G)$	kantenchromatischer Index	58
$\chi(G)$	knotenchromatischer Index	58
$\Delta(G)$	Maximalgrad eines Graphen	55
$\delta(G)$	Minimalgrad eines Graphen	55
$\delta(r)$	Übertragungsdauer	126
Γ	Menge von Übertragungen	126
Γ_v	Menge von Übertragungen an einem Netzwerkknoten	126
$\hat{\alpha}(r)$	Aktivitätsintervall der Ausführung einer Übertragung	136
$\mathbb{D}$	Menge von Geräten	126
$\mathbb{E}$	Menge von Netzwerkkabeln	126
$\mathbb{E}(v)$	Menge von Netzwerkkabeln an einem Switch	126
$\mathbb{S}$	Menge von Switches	126
$\mathbb{V}$	Menge von Knoten	126
$\mu(G)$	Multiplizität eines Graphen	55
$\Re_\alpha$	Menge der Äquivalenzklassen	145
$\sim$	Äquivalenzrelation	56
$\underline{n}$	Menge zu einer natürlichen Zahl	54
$\vec{\psi}$	maximale gerichtete Last im Netzwerk	146
C_{Kanten}	Kantenkonfliktgraph	131

C_{Knoten}	Knotenkonfliktgraph	129
$cl(G)$	größte Clique	56
$d(v)$	Grad eines Knotens	55
E	Menge von Kanten	54
e^{P+}	Endknoten einer Kante in einem Weg	57
e^{P-}	Anfangsknoten einer Kante in einem Weg	57
$E_{KL(r)}$	Menge an Netzwerkkanten einer Übertragung	127
$E_P = E$	Kanten eines Weges	128
E_r	Menge an Netzwerkkanten einer Übertragung	127
$G' = G[V']$	Teilgraph, Untergraph	54
$G := (V, E, g)$	(ungerichteter) (Hyper-)Graph	54
$h(G)$	maximale Länge eines Weges in einem Graphen	127
$K \leftrightsquigarrow_e L$	Kommunikationslinien K und L in Konflikt an Kabel e	127
$K \leftrightsquigarrow_V L$	Kommunikationslinien in Konflikt an der Menge von Switches	127
$K \leftrightsquigarrow_v L$	Kommunikationslinien K und L in Konflikt an Switch v	127
$KL(\Gamma - ID)$	Kommunikationslinie	126
$L(G)$	Kantengraph	56
$l(G)$	maximale Länge eines Weges	58
$M^{\underline{n}}$	n-elementige Teilmenge	54
$P(M)$	Potenzmenge von M	54
$P = (\bar{x}, \bar{e})$	Pfad in einem Graphen	56
P^{-1}	inverser Weg	57
$Q \triangleleft P$	Q ist Teilweg von P	57
T_{SC}	SendClock von ProfiNet	89
T_U	Übertragungszeit eines Frames	169
T_V	Verzögerungszeit eines Verteilers	168
T_Z	Zykluszeit	12

V	Menge von Knoten	54
v^{P+}	Nachfolgerkante eines Knotens in einem Weg	57
v^{P-}	Vorgängerkante eines Knotens in einem Weg	57
$V_{KL(r)}$	Menge an Netzwerkknoten einer Übertragung	127
$V_P = V$	Knoten eines Weges	128
V_r	Menge an Netzwerkknoten einer Übertragung	127
vw	Kante mit Randknoten v und w	54
ABK	Anzeige- und Bedienkomponente	22
ABK	Reduced Media Independent Interface	92
ADU	Analog-Digital Umsetzer	9
API	Application Programming Interface	109
AS-i	Aktor/Sensor-Interface	1
ASIC	Application Specific Integrated Circuit	92
AT	Acknowledge Telegramm	101
ATM	Asynchronous Transfer Mode	123
B2E	Business-to-Employee	32
BPDU	Bridge Protocol Data Unit	39
C2C	Controller-to-Controller Communication	101
CAN	Controller Area Network	27
CBA	Component Based Automation	87
CIP	Common Industrial Protocol	71
CMP	Completed Message	80
CN	Controlled Node	76
CoE	CANopen over EtherCAT	107
COTS	Commercial off the Shelf	71
CPF	Communications Profile Family	79
CPU	Central Processing Unit	11

CRC	Cyclic Redundancy Check	36
CSMA/CD	Carrier Sense, Multiple Access with Collision Detection	3
DCOM	Distributed Component Object Model	87
DIX	Digital Equipment Corp., Intel und Xerox	34
DOMA	Deterministic Ordered Multiple Access	80
DP	Dezentrale Peripherie	27
DPRAM	Dual-Port Random Access Memory	93
DSATUR	Saturation Largest First	65
ECSME	EPA Communication Scheduling Management Entity	82
EDF	Earliest Deadline First Scheduling	121
EoC	End-of-Cycle	76
EoE	Ethernet over EtherCAT	107
EPA	Ethernet for Plant Automation	82
EPL	Ethernet PowerLink	75
ERP	Enterprise-Resource-Planning	2
ERTEC	Enhanced Real Time Ethernet Controller	92
ETH	Ethernet	27
EtherCAT	Ethernet for Control Automation Technology	103
EtherNet/IP	Ethernet Industrial Protocol	71
FCFS	First Come First Served	121
FCS	Frame Check Sequence	36
FF	First-Fit	123
FIFO	First in, First out	206
FMMU	Fieldbus Memory Management Unit	105
FoE	FileSystem over EtherCAT	107
FPGA	Field Programmable Gate Array	103
FQ	Fair Queuing	121

GPL	General Public License	83
GPS	Global Positioning System	48
GSM	Graphical Schedule Manager	177
HD	Halbduplexmodus	37
HDR	Header	101
HSE	High Speed Ethernet	74
HTTP	HyperText Transfer Protocol	19
I/O	Input/Output	78
IAONA	Industrial Automation Open Network Alliance	23
IDA	Interface for Distributed Automation	22
IEC	International Electrotechnical Commission	12
IEEE	Institute of Electrical and Electronics Engineers	33
IETF	Internet Engineering Task Force	41
IFG	Interframe Gap	43
IGMP	Internet Group Message Protokoll	44
ILP	Integer Linear Program	62
IP	Internet Protocol	27
IP-Core	Intellectual Property Core	208
IRT	Isochronous Realtime Traffic	26
ISO	International Organization for Standardization	7
ISP	Interoperable Systems Project	73
KL	Kommunikationslinie	126
LAN	Local Area Network	32
LF	Largest First Anordnung	64
LLC	Logical Link Control	197
MAC	Media-Access-Control	33
MC	Mikrocontroller-Steuerug	99

MDI	Media Dependent Interface	196
MDT	Master Data Telegramm	101
MES	Manufacturing Execution System	22
MIB	Management Information Base	41
MII	Media Independent Interface	93
MII	Media Independent Interface	196
MN	Managing Node	76
MST	Master Sync Telegramm	101
MTU	Maximum Transfer Unit	77
NED	Network Definition Datei	181
nRT	non-Realtime Traffic	27
NTP	Network Time Protocol	48
ODVA	Open DeviceNet Vendor Association	71
OSI	Open Systems Interconnection	7
OUI	Organizationally Unique Identifier	33
PAA	Prozessabbild der Ausgänge	12
PAE	Prozessabbild der Eingänge	12
PC	Personal Computer	1
PCI	Peripheral Component Interconnect	92
PCS	Physical Coding Sublayer	196
PDO	Process Data Object	78
PHY	Physical Attachment Layer	196
PHY	Physical-Layer Baustein	92
PI-Regler	Regler mit Proportional- und Integralteil	15
PLS	Physical Line Signaling	196
PMA	Physical Media Attachment	196
PMD	Physical Media Dependent	196

PNK	prozessnahe Komponente	22
PNO	Profibus Nutzerorganisation e. V.	30
PQ	Priority Queuing	121
PTP	Precision Time Protocol	49
RETHER	Real-Time Ethernet	86
RFC	Requests for Comments	34
RFID	Radio Frequency Identification Transponder	8
RISC	Reduced Instruction Set Computing	111
ROM	Read Only Memory	208
RSTP	Rapid Spanning Tree Protocol	39
RT	Realtime Traffic	27
RTAI	Real Time Application Interface	83
RTC	Real-Time Crossbar	202
SCNM	Slot Communication Network Management	76
SDO	Service Data Object	78
SDRAM	Synchronous Dynamic Random Access Memory	111
SERCOS	Serial Real Time Communication System Interface	98
SFD	Start Frame Delimiter	35
SL	Smallest Last Anordnung	65
SNMP	Simple Network Management Protokoll	41
SNTP	Simple Network Time Protocol	48
SoA	Start-of-Asynchronous	77
SoC	Start-of-Cycle	76
SoE	Servodrive over EtherCAT	107
SPS	speicherprogrammierbare Steuerung	2
SRT	Soft Realtime	87
STP	Spanning Tree Protocol	39

TCI	Tag Control Information	36
TCnet	Time-critical Control Network	79
TCP	Transmission Control Protocol	27
TDMA	Time Division, Multiple Access	3
THT	Token Holding Time	86
TPID	Tag Protocol Identifier	36
TTL	Transistor=Transistor=Logik	198
UART	Universal Asynchronous Receiver/Transmitter	30
UDP	User Datagram Protocol	27
VD	Vollduplexmodus	38
VDW	Verein Deutscher Werkzeugmaschinenfabriken e. V.	99
VHDL	Very High Speed Integrated Circuit Hardware Description Language	103
VLAN	Virtual Local Area Network	36
VPS	verbindungsprogrammierte Steuerung	11
WC	Working Counter	106
XML	Extensible Markup Language	111
ZVEI	Zentralverband Elektrotechnik- und Elektronikindustrie e.V.	98

Index