

PC-WELT

SONDERHEFT

Jan./Feb./März 2005
www.tecChannel.de

NEU!

Empfohlen als
Vorlesungsskript
an Hochschulen

tecCHANNEL

tecCHANNEL COMPACT

KOMPENDIUM FÜR IT-PROFIS

€ 9,90 Österreich € 10,90 Benelux € 11,40 Schweiz SFR 19,80

Netzwerk- Know-How

100% Wissen kompakt und verständlich

- Ethernet, WLAN, ATM und Token Ring
- Alle Details zu IP, TCP und UDP
- So funktionieren HTTP, SMTP und VoIP
- Sicherheit durch Firewall und VPN
- Verschlüsselung mit DES und RSA
- Netzwerkprogrammierung mit Java und C

Großer Praxisteil

PROGRAMMIERUNG

- › Client&Server-
Programmierung
- › TCP- und UDP-
Sockets mit C und Java

NETZWERK-SNIFFING

- › Grundlagen, kostenlose
Tools, Analyse im LAN
- › Sniffing im WLAN,
Angreifer entlarven

ÜBUNGEN & LÖSUNGEN

- › Umfangreiche Trainings-
aufgaben zu den Kapiteln
- › Detaillierte Lösungen
und Listings

01

4 195914 909900

Editorial

Rechnernetzwerke



Im Oktober 1969 begann das ARPANET als Verbindung zwischen vier Knoten. Aus dem ARPANET entwickelte sich das Internet mit heute etwa 200 Millionen angeschlossenen Rechnern. Das Internet – hier stellvertretend für alle Rechnernetze genannt – dient dabei zunehmend als Infrastruktur für die unterschiedlichsten Anwendungen. Längst übertragen derartige Netze neben klassischen Computerdaten auch Sprache, Musik und Video. Gleichzeitig enthalten immer mehr mobile und stationäre Geräte die Möglichkeit, direkt oder indirekt über das Netz miteinander zu kommunizieren.

Nach Einführung der grundlegenden Konzepte behandelt dieses tecCHANNEL-Compact Netzwerke zunehmender Komplexität. Ausgangspunkt ist die Übertragung von Daten zwischen lediglich zwei Rechnern. Darauf aufbauend beschreibt das Compact schrittweise die Erweiterung auf lokale Netze und schließlich das globale Internet. Die Darstellung folgt dabei im Wesentlichen den derzeit dominierenden Standards Ethernet und TCP/IP. Mehrere Kapitel sind der Programmierung von Netzwerkanwendungen gewidmet. Fragen der Datensicherheit und Verschlüsselung behandelt ein eigenes Kapitel. Alle Listings dieses Compacts sowie ausgewählte fertige Programme können Sie bequem von tecChannel.de über den webcode a1541 herunterladen.

Der Schwerpunkt dieser tecCHANNEL-Compact-Ausgabe liegt auf der Darstellung von Grundprinzipien von Netzwerken. Es sollen weniger die technischen Details als vielmehr die grundlegenden Design-Entscheidungen und die daraus resultierenden Möglichkeiten und Grenzen vermittelt werden. Zahlreiche Programmbeispiele und Aufgaben dienen der Vertiefung des Stoffes. Zur eigenen Kontrolle sind ausgewählte Lösungen angegeben.

Ich empfehle dieses tecCHANNEL-Compact für Studierende der Informatik, Wirtschaftsinformatik und verwandter Fächer zum Selbststudium oder begleitend zu Vorlesungen sowie allen Interessierten, die einen Einblick in die immer selbstverständlicher werdende Vernetzung erhalten wollen.

Prof. Dr. Stephan Euler

An unserer Leserumfrage mit Gewinnchance zu dieser Compact-Ausgabe können Sie unter www.tecChannel.de/compact0105.html teilnehmen. Die Listings und fertig compilierte Beispielprogramme finden Sie auf tecChannel.de unter dem webcode a1541.

Impressum

Chefredakteur: Michael Eckert (verantwortlich, Anschrift der Redaktion)

Chef vom Dienst / Textchef: Kerstin Lohr

Grafik: stroemung, Michael Rupp, Oliver Eismann; Yvonne Reitinger, Bernd Maier-Leppla

Titelgestaltung: Twentyfirst Communications

Redaktion tecCHANNEL:

Leopoldstraße 252b, 80807 München,

Tel.: 0 89/3 60 86-897, Fax: -878

Homepage: www.tecChannel.de,

E-Mail: redtecchannel@idginteractive.de

Autoren dieser Ausgabe: Prof. Dr. Stephan Euler, Mike Hartmann

Copyright: Das Urheberrecht für angenommene und veröffentlichte Manuskripte liegt bei der IDG Interactive GmbH. Eine Verwertung der urheberrechtlich geschützten Beiträge und Abbildungen, vor allem durch Vervielfältigung und/oder Verbreitung, ist ohne vorherige schriftliche Zustimmung des Verlags unzulässig und strafbar, soweit sich aus dem Urheberrechtsgesetz nichts anderes ergibt. Eine Einspeicherung und/oder Verarbeitung der auch in elektronischer Form vertriebenen Beiträge in Datensysteme ist ohne Zustimmung des Verlags nicht zulässig.

Bildquelle Titel: Sun Microsystems

Anzeigen:

Anzeigenleitung: Dirk Limburg,

Tel.: 0 89/3 60 86-871

Anzeigendisposition: Rudolf Schuster (leitend),

Tel.: 0 89/3 60 86-135, Fax: -328

Anzeigentechnik: Martin Mantel, Andreas Mallin

Digitale Anzeigenannahme: Thomas Wilms

(leitend), Tel.: 0 89/3 60 86-604, Fax: -328

Vertrieb / Produktion:

Vertrieb: Josef Kreitmair (leitend), Katrin Elsler

Vertriebsmarketing: Peter Priewasser (leitend), Stefanie Kusseler

Vertrieb Handelsauflage: MZV Moderner Zeitschriften Vertrieb, Breslauer Straße 5, 85386

Eching, Tel.: 0 89/3 19 06-0, Fax: -113, E-Mail:

mzv@mzv.de, Website: www.mzv.de

Produktionsleitung: Heinz Zimmermann

Druck: Schoder Druck, Gutenbergstraße 12, 86368 Gersthofen

Haftung: Eine Haftung für die Richtigkeit der Beiträge können Redaktion und Verlag trotz sorgfältiger Prüfung nicht übernehmen. Veröffentlichungen in tecCHANNEL-Compact erfolgen ohne Berücksichtigung eines eventuellen Patentschutzes. Warennamen werden ohne Gewährleistung einer freien Verwendung benutzt. Veröffentlichung gemäß § 8, Absatz 3 des Gesetzes über die Presse vom 8.10.1949: Alleiner Gesellschafter der IDG Interactive GmbH ist die IDG Communications Verlag AG, München, eine 100-prozentige Tochter der IDG Inc., Boston, Mass., USA

Verlag: IDG Interactive GmbH, Leopoldstraße 252b, 80807 München, Tel.: 0 89/3 60 86-0, Fax: -501, Website: www.idg-verlag.de

Handelsregisternummer: HR 127140

Umsatzidentifikationsnummer: DE 203 066

Geschäftsführer: York von Heimburg

Group Publisher: Stephan Scherzer

Verlagsleitung: Frank Klinkenberg

Vorstand: York von Heimburg, Keith Arnot, Pat Kenealy

Mitglieder der Konzerngeschäftsführung:

Karin Giffhorn, Stephan Scherzer, Josef Lohner

Aufsichtsratsvorsitzender:

Patrick J. McGovern

Dieses tecCHANNEL-Compact wurde mit der Adobe Creative Suite CS produziert. tecCHANNEL-Compact erscheint im Verlag der PC-WELT. Zu unserer Verlagsgruppe gehören folgende Zeitschriften:

COMPUTERWOCHE
Macwelt

ComputerPartner
GameStar

PC-WELT
CIO

DigitalWorld
gamepro

Leser- und Abo-Service:

Dialog-Service-Center GmbH, Konrad-Zuse-Straße 16, 74172 Neckarsulm, Telefon: 0 18 05/9 99-802, Fax: 0 71 32/9 59-166, E-Mail: tecchannel@d-s-center.de

Inhalt

	Editorial	3
	Impressum	4
1.	Einführung	9
2.	Netzwerkgrundlagen	11
2.1	Vermittlungsverfahren	11
2.2	Leistungsfähigkeit	12
2.2.1	Bedeutung von Bandbreite und Latenz	14
2.2.2	Verzögerung-Bandbreite-Produkt	14
2.2.3	Anwendungen	15
2.3	Netzwerktypen und -topologien	16
2.3.1	Größe	16
2.3.2	Topologien	17
2.4	Referenzmodelle	20
2.4.1	Protokollstapel	20
2.4.2	OSI-Referenzmodell	21
2.4.3	Bedeutung des OSI-Referenzmodells	23
3.	Rechner-zu-Rechner-Verbindung	25
3.1	Übertragung von Bits	25
3.2	Rahmenerstellung	27
3.2.1	Markierungszeichen	27
3.2.2	Zeichenzähler	28
3.3	Fehlererkennung	29
3.3.1	Parität	30
3.3.2	Zyklische Redundanzprüfung	32
3.4	Sichere Übertragung von Rahmen	34
3.4.1	Stop-and-Wait-Algorithmus	35
3.4.2	Sliding-Window-Algorithmus	36
4.	Ethernet & Co.	38
4.1	Ethernet	38
4.1.1	Adressen	38
4.1.2	Rahmenformat	39
4.1.3	Medienzugriff	39
4.1.4	Physikalische Eigenschaften	41
4.1.5	Bewertung	42

4.2	Token Ring	42
4.2.1	Medienzugriff	43
4.2.2	Netzüberwachung	44
4.3	Drahtlose LAN	44
4.4	Andere Netztechnologien	46
5.	Vermittlung	49
5.1	Datagramme	51
5.1.1	Lernende Bridges	52
5.2	Virtuelle Verbindungen	52
5.3	Design von Switches	54
5.3.1	Weiterleitung	55
5.3.2	Knockout-Switch	56
5.3.3	Batcher- und Banyan-Netzwerke	56
6.	Internet-Protokoll IP	58
6.1	Adressen	58
6.2	Paketformat	61
6.3	Weiterleitung	62
6.4	Zuordnung IP-Adresse zu Ethernet-Adresse	63
6.5	Internet Control Message Protocol	64
6.6	Routing – eine kurze Einleitung	65
6.7	Distanzvektor-Routing	67
6.8	Interdomain-Routing	69
6.9	Domain Name System DNS	70
6.10	Internet-Standards	71
7.	UDP und TCP	73
7.1	UDP	73
7.2	TCP	75
7.2.1	Segmentierung	75
7.2.2	Verbindungsaufbau	77
7.2.3	Sliding Window	77
7.2.4	Überlastkontrolle	79
8.	ATM-Netzwerktechnologie	81
8.1	ATM-Grundlagen	82
8.2	ATM-Zellengrößen	83
8.3	ATM-Zellen-Header	84
8.4	ATM-Anpassungsschicht	85
8.5	Übergänge zwischen LAN und ATM	86

9.	Sockets	88
9.1	Verbindungsorientierte Kommunikation	88
9.1.1	Funktion „socket“	89
9.1.2	Funktion „bind“	90
9.1.3	Funktion „listen“	91
9.1.4	Funktion „accept“	91
9.1.5	Funktion „send“ und „recv“	92
9.1.6	Einfacher Server	92
9.1.7	Funktion „connect“	94
9.2	Verbindungslose Kommunikation	95
9.3	Realisierung in Perl	99
10.	Remote Procedure Call RPC	100
10.1	Grundlagen	101
10.2	RPC-Beispiel	101
10.3	RPC-Server	103
10.4	RPC-Client	104
10.5	Compilieren und Linken	106
10.6	Abfrage-Prozedur	107
11.	Netzwerkanwendungen	110
11.1	World Wide Web WWW	110
11.1.1	Das Protokoll HTTP	111
11.1.2	Universal Resource Identifier	113
11.1.3	HTTP-Cache	114
11.2	E-Mail	115
11.2.1	Das Protokoll SMTP	116
11.2.2	Nachrichtenformat und MIME	116
11.2.3	E-Mail-Adressierung	118
11.3	Usenet	119
11.4	Netzwerk-Management	120
11.5	Multimedia-Kommunikation	121
11.5.1	Real-Time Transport Protocol	121
11.5.2	RTP-Verbindungsaufbau	124
11.5.3	VoIP – Sprachübertragung über IP	124
11.5.4	Sprachkodierung	124
11.5.5	Telefonanwendungen	127
12.	Java-Einsatz im Netzwerk	129
12.1	Java Client-Sockets	129

12.2	Java Server-Sockets	130
12.3	Java UDP-Sockets	131
12.4	Die Klasse URL	135
12.5	Mailto-Links	137
13.	Datensicherheit und Verschlüsselung	139
13.1	Firewall	139
13.2	Virtuelle private Netze	141
13.3	Verschlüsselung	142
13.3.1	Monoalphabetische Verschlüsselung	143
13.3.2	Digitale Verschlüsselung	145
13.3.3	Gemeinsame Schlüsselvereinbarung	146
13.3.4	Öffentliche Schlüssel	149
13.4	Anwendungen	151
13.4.1	PGP	151
13.4.2	Sichere Transportschicht	151
14.	RSA im Detail	153
15.	Netzwerk-Sniffer	159
15.1	Sniffing-Grundlagen	159
15.2	Sniffing-Tools	160
15.3	Analyzer 3.0alpha	161
15.4	CommView 4.1	163
15.5	Erste Schritte mit Sniffing	165
15.6	Sniffing im Netzwerk	167
15.7	Sniffing im geschwichtten LAN	168
15.8	Sniffing im WLAN	169
15.9	Lokale Angriffe von Sniffen	170
15.10	Entdecken von Sniffen	170
15.11	Fazit	171
16.	Übungen und ausgewählte Lösungen	172
	Akronyme	185
	Literaturverzeichnis	189
	Glossar	191
	Index	196
	tecCHANNEL-Leserumfrage – Mitmachen und gewinnen!	200

1. Einführung

Netzwerke dienen der Kommunikation. Sie überbrücken dabei eine mehr oder weniger große Distanz zwischen den Teilnehmern. Betrachten wir zunächst den einfachen Fall einer Kommunikation zwischen lediglich zwei Teilnehmern. Einer der Teilnehmer fungiert als Sender (Quelle), der andere als Empfänger (Ziel) der Nachricht. Diese Rollen – Sender und Empfänger – stehen nicht für immer fest, sondern wechseln im Allgemeinen. Der Sender formuliert die Nachricht und schickt sie anschließend über das Netzwerk an den Empfänger. Seit jeher wurden für unterschiedliche Anwendungen auch unterschiedliche Netzwerke eingesetzt. Hierzu einige Beispiele:

- Der römische Kaiser schickt einen Befehl an einen Statthalter in einer entfernten Provinz. Als Netzwerk dienen berittene Boten.
- Sie schicken eine Ansichtskarte aus dem Urlaub nach Hause.
- Ein Professor ruft im Sekretariat seiner Hochschule an.
- Die Bankmitarbeiterin sendet eine elektronische Kursabfrage an die Börse.
- Ein Student verschickt eine Nachricht als SMS an einen Kommilitonen.
- Ein Benutzer schickt eine Datei zum Ausdrucken an einen Druck-Server.

Anhand dieser einfachen Beispiele lassen sich bereits einige der Hauptthematiken rund um Netzwerke erkennen. Das erste fundamentale Problem lautet: Wie kommt die Nachricht vom Sender zum Empfänger? Daraus ergibt sich die Frage nach der Adressierung und Vermittlung. Zunächst benötigt man eine eindeutige Zieladresse, um den Empfänger identifizieren zu können. Dann kann, ausgehend von der Startadresse, ein Weg zum Ziel festgelegt werden.

Weitere Grundfragen betreffen die Leistungsfähigkeit des Kanals: Wie schnell können wie viele Daten übertragen werden? Dabei sind sowohl physikalische Grenzen (wie viele Pakete passen in ein Postauto?) als auch die Verteilung der Last unter den verschiedenen Teilnehmern im Netzwerk zu betrachten. Im Zusammenhang mit Entwurf und Betrieb von Netzwerken spielen noch weitere Gesichtspunkte eine Rolle, wie unter anderem:

- Physikalische Realisierung
- Netzwerkarchitektur
- Gebühren
- Sicherheit gegenüber Dritten
- Zuverlässigkeit im Falle von Übertragungsfehlern
- Ausfallsicherheit

Im Rahmen dieses tecCHANNEL-Compacts werden wir uns mit der Verbindung zwischen Rechnern beschäftigen. Traditionell unterscheidet sich diese in Anforderung und Technik von Kommunikationsnetzen, wie sie für Telefonverbindungen genutzt werden. Bedingt durch die fortschreitende Digitalisierung verschwim-

men aber zunehmend die Grenzen zwischen den verschiedenen Netzwerken, insbesondere zwischen Rechnernetzen und anderen Kommunikationsnetzen. So finden beispielsweise Systeme, die es ermöglichen, über Computerverbindungen zu telefonieren (Voice over IP), zunehmend Verbreitung. Ein großer Vorteil besteht hierbei im Wegfall einer zweiten, getrennten Verkabelung. Bei mobilen Endgeräten werden bereits Kombinationen von Telefon und Rechner angeboten. Auch im Unterhaltungsbereich ist der Übergang zu digitalen Inhalten, etwa bei Audio oder Fotografie, bereits weit fortgeschritten. Ganz allgemein geht der Trend hin zu intelligenten Endgeräten. Damit spielt die Vernetzung all dieser Geräte eine immer wichtigere Rolle. Man denke etwa an die Vision eines intelligenten Hauses, bei dem Sensoren und Geräte zur Klimasteuerung, Beleuchtung, Sicherheit, Kommunikation und Unterhaltung untereinander verknüpft sind.

Im Zuge dieser so genannten Konvergenz verschiedener Netze ist abzusehen, dass die Techniken der Rechnerkommunikation zukünftig in immer mehr Bereichen Anwendung finden werden. Allerdings führt die Kombination verschiedener Dienste mit sehr unterschiedlichen Anforderungen bezüglich Ausfallsicherheit und Antwortzeit zu neuen, anspruchsvollen Herausforderungen für den Entwurf von Netzwerken.

Im nächsten Kapitel werden einige grundlegende Begriffe und Konzepte eingeführt. Davon ausgehend diskutieren wir anschließend zunächst die direkte Kommunikation zwischen zwei Rechnern. In den darauf folgenden Kapiteln wird dann schrittweise der Umfang der betrachteten Netze erweitert, bis hin zum globalen Internet. Ein abschließendes Kapitel thematisiert Datensicherheit und Verschlüsselung. Die Vorlesung folgt dabei im Wesentlichen den derzeit dominierenden Standards Ethernet und TCP/IP. Auf die Vielzahl alternativer oder konkurrierender Technologien können wir nur in Einzelfällen näher eingehen. Die dargestellten Grundlagen sind jedoch weitgehend unabhängig von der konkreten Technologie und daher leicht übertragbar.

Drei Kapitel behandeln Themen der Programmierung von Netzwerkanwendungen. Dabei wird zunächst die Programmierung von Sockets und Remote Procedure Calls in der Sprache C vorgestellt. Ein weiteres Kapitel bietet eine kurze Einführung in die Netzwerkprogrammierung mittels Java.

In vielen Fällen lassen sich die Ansätze in Rechnernetzen auf vertraute Konzepte aus den klassischen Bereichen von Post und Telefon zurückführen. Insbesondere die beiden Grundprobleme Adressierung und Zustellung bleiben trotz aller Unterschiede in der technischen Realisierung dieselben. Im Folgenden wird häufig explizit auf diese Analogie hingewiesen. Darüber hinaus ist es immer wieder sinnvoll, sich die Algorithmen durch Übertragung – beispielsweise auf Briefe oder Postpakete – zu veranschaulichen.

Prof. Dr. Stephan Euler

2. Netzwerkgrundlagen

Zum wirklichen Verständnis von Netzwerktechnik ist ein solides Basiswissen die Grundvoraussetzung. Dieses Kapitel verschafft Ihnen das notwendige Know-how von den Basics bis hin zu Netzwerktopologie und OSI-Schichtenmodell.

2.1 Vermittlungsverfahren

Lange Zeit waren die beiden wohl wichtigsten Kommunikationsnetze Post und Telefon. Diese beiden Netze sind gleichzeitig Repräsentanten für zwei Netzwerktypen. Bei dem Postdienst werden einzelne Sendungen (Briefe, Pakete oder Ähnliches) vermittelt. Jedes Paket trägt die Zieladresse. Innerhalb des Netzes wird anhand dieser Zieladresse der Laufweg festgelegt: Briefkasten – örtliches Postamt – Postverteilzentrum – ... – Empfänger. Aber selbst wenn man mehrere Briefe gleichzeitig an den gleichen Adressaten aufgibt, ist nicht garantiert, dass die Briefe den gleichen Weg nehmen und gleichzeitig ankommen. Typischerweise wird an jedem Punkt nur über den nächsten Schritt entschieden. Diese Art der Vermittlung nennt man „Paketvermittlung“. Charakteristisch sind:

- diskrete Einheiten (Pakete)
- jedes Paket trägt die Adresse
- jedes Paket wird einzeln weitergeleitet
- im Allgemeinen keine Garantien bezüglich Laufweg und Laufzeit
- Robustheit gegenüber Unterbrechung einzelner Verbindungen

Demgegenüber wird beim klassischen Telefonnetz eine feste Verbindung zwischen den beiden Teilnehmern aufgebaut. Der Anrufer meldet seinen Verbindungswunsch an. Daraufhin wird im System eine Leitung zu dem Ziel aufgebaut. Wenn die Gesprächsverbindung zu Stande kommt, wird für die Dauer des Gesprächs eine feste Übertragungsrate exklusiv reserviert. Man spricht daher von „Leitungsvermittlung“. Wesentliche Eigenschaften sind:

- Aufbau einer Verbindung (Leitung) zwischen Quelle (Anrufer) und Ziel (Angerufenem)
- Erreichbarkeit des Zieles wird vor Beginn der Kommunikation sichergestellt
- Kommunikation erfolgt unter exklusiver Nutzung der Verbindung
- Zum Abschluss wird die Verbindung abgemeldet (Verbindungsabbau)
- Vorteil: konstante und garantierte Qualität
- Nachteil: schlechte Ausnutzung bei variabler Übertragungsrate
- Nachteil: empfindlich gegenüber Störung im Verbindungspfad

2.2 Leistungsfähigkeit

Für die Realisierung verschiedener Dienste sind unterschiedlich leistungsfähige Netzwerke erforderlich. Primäre Kriterien sind dabei Durchsatz und Verzögerung. Durchsatz oder Bandbreite gibt an, wie viele Daten in einer Zeiteinheit übertragen werden können. Das übliche Maß ist Bit pro Sekunde beziehungsweise die daraus abgeleiteten Einheiten wie etwa Mbit/s oder Gbit/s. Die Angabe 10 Mbit/s beispielsweise bedeutet, dass das Netzwerk 10 Millionen Bits in einer Sekunde übertragen kann. Umgekehrt lässt sich aus dieser Angabe berechnen, wie viel Zeit die Übertragung eines einzelnen Bits benötigt. Für das Beispiel des 10-Mbit/s-Netzwerks ergibt sich der Wert von $0,1 \mu\text{s}$ (Mikrosekunden). Dies ist allerdings nur die Zeit, bis der Sender das nächste Bit abschicken kann. Man darf diesen Wert nicht mit der Laufzeit bis zum Empfänger verwechseln. Wenn man sich die einzelnen Datenbits als eine Folge von Impulsen auf der Zeitachse vorstellt, kann man sagen, dass ein Bit (Impuls plus Pause) $0,1 \mu\text{s}$ breit ist.

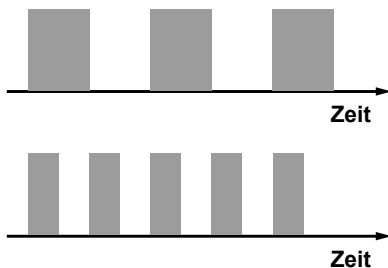


Bild 2.1: Einzelne Bits als zeitliche Impulse.

© tecCHANNEL

Die Bandbreite wird durch die technische Realisierung der Leitung bestimmt. Eine Bandbreite von 10 Mbit/s bedeutet, dass irgendwo im System ein Takt von 10 MHz vorgegeben wird. Die Angabe von Frequenzen folgt der üblichen Konvention von Zehnerpotenzen, 10 MHz bedeutet 10^6 Hz. Daraus abgeleitet versteht man unter 10 Mbit/s auch 10^6 Bit/S. Demgegenüber erfolgt die Größenangabe von Computerspeicher in Potenzen von 2. Ein Kbit umfasst $2^{10}=1024$ Bit. Die unterschiedliche Interpretation muss im Prinzip immer berücksichtigt werden. So dauert etwa die Übertragung einer Datenmenge von 1 kbit über eine 1-kbit/s-Verbindung nicht exakt eine Sekunde. Vielmehr berechnet sich der exakte Wert zu:

$$\frac{1024 \text{ bit}}{1000 \text{ bit/s}} = 1,024 \text{ s}$$

Glücklicherweise ist der Unterschied gering (1,024 im Vergleich zu 1,000). Da in den allermeisten Fällen nur die ungefähre Größenordnung benötigt wird, kann man diese Feinheit zunächst ignorieren und den kleinen Fehler in Kauf nehmen.

Die zweite wichtige Charakteristik einer Verbindung ist die Verzögerung oder Latenz (von Lateinisch: lateo = verborgen, versteckt, unbekannt sein). Latenz bezeichnet die Zeit, die eine Nachricht für den Weg vom Sender bis zum Empfänger benötigt. Die Latenz beinhaltet drei Komponenten:

1. Ausbreitungsverzögerung
2. Übertragungsverzögerung
3. Wartezeit

Die Ausbreitungsverzögerung t_A resultiert aus der endlichen Geschwindigkeit, mit der sich Signale ausbreiten. Kein Signal kann schneller als mit Lichtgeschwindigkeit übertragen werden. Die Lichtgeschwindigkeit hängt vom Übertragungsmedium ab. Als Richtwert kann der Wert $c = 3.0 \times 10^8$ m/s für die Ausbreitung im Vakuum dienen. Damit kann man näherungsweise für die Ausbreitungsverzögerung bei einer Leitungslänge l den Wert $t_A = l/c$ ansetzen. Bei der Strecke Friedberg – Berlin (zirka 500 km) erhält man

$$t_A = (500 \times 10^3) / (3 \times 10^8) = 166.7 / 10^5 = 1,667 \times 10^{-3} = 1,667 \text{ ms} \quad (2.1)$$

Die Übertragungsverzögerung t_U ist durch die bereits oben diskutierte Dauer eines Bits bestimmt. Wir haben gesehen, dass bei einer Bandbreite von B Bit/s ein einzelnes Bit $1/B$ s an Übertragungszeit benötigt. Für ein Objekt der Größe K ergibt sich dann

$$t_U = K/B \quad (2.2)$$

Die Definition bezieht sich auf größere Objekte, da in der Regel nicht ein einzelnes Bit, sondern immer größere Datenblöcke als eine Einheit gesendet werden.

Die dritte Komponente – Wartezeit t_w – spielt dann eine Rolle, wenn die Übertragungsstrecke nicht nur aus einer einzelnen Leitung, sondern aus mehreren Teilstücken besteht. Dann entsteht in den dazwischen liegenden Vermittlungsknoten eine Wartezeit.

Die gesamte Wartezeit hängt von der Verarbeitungsgeschwindigkeit der einzelnen Knoten, der Größe der Datenpakete sowie der Anzahl der Knoten ab. In praktischen Fällen ist die Wartezeit oft der größte Beitrag zur Latenz. Bei Paketvermittlung wartet ein Vermittlungsknoten häufig, bis er ein Paket vollständig erhalten hat, bevor er es weiter schickt. Bei hohem Verkehrsaufkommen muss das Paket zusätzlich auf eine freie Lücke warten. Typisch für beispielsweise einen transkontinentalen Kanal ist eine Latenz in der Größenordnung von 50 ms.

Die Latenz ist die Verzögerung vom Sender zum Empfänger. Häufig ist auch die Angabe zu finden, wie lange ein Paket vom Sender zum Empfänger und dann wieder zurück benötigt. Man spricht dann allerdings korrekterweise von Roundtrip-Zeit (Englisch: Round-Trip Time, RTT). Zur besseren Unterscheidung wird in Fällen, in denen eine Verwechslung auftreten könnte, für die einfache Verzögerung der Ausdruck „Einweglatenz“ verwendet.

2.2.1 Bedeutung von Bandbreite und Latenz

Prinzipiell kann man sagen, je größer die Bandbreite und je kleiner die Verzögerung, desto „besser“ ist die Verbindung. Allerdings hängen die konkreten Anforderungen von der jeweiligen Anwendung ab. Bei kleinen Objekten – zum Beispiel der Übermittlung eines Tastendrucks – dominiert die Verzögerungszeit. Bei einem Kanal mit 1 Mbit/s beträgt die Übertragungsverzögerung für 1 Byte

$$t_U = 8 / (1 \times 10^6) = 8 \mu\text{s} = 0,008 \text{ ms} \quad (2.3)$$

Dieser Wert ist im Vergleich zur gesamten Latenz nahezu vernachlässigbar. Bei größeren Objekten erfolgt die Übertragung in vielen kleineren Paketen. Die gesamte Übertragungsdauer – die vom Benutzer wahrgenommene Latenz – berechnet sich dann aus der Latenz eines einzelnen Pakets und der benötigten Zeit, um entsprechend viele Pakete zu übertragen. Für die Übertragung einer Video-Datei mit 20 MByte berechnete man bei dem 1-Mbit/s-Kanal für die reine Übertragungsverzögerung

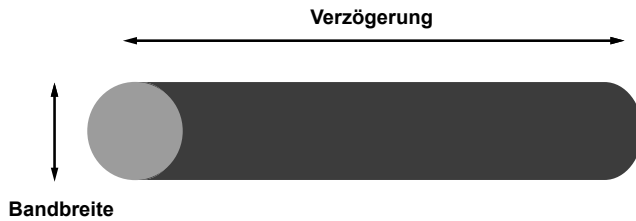
$$t_U = 8 \cdot 20 \times 2^{20} / (1 \times 10^6) = 168 \text{ s} \quad (2.4)$$

Hier spielt die Laufzeit eines einzelnen Pakets keine Rolle. Die Übertragung lässt sich nur durch Erhöhen der Bandbreite signifikant verbessern. In Tabelle 2.1 sind für verschiedene Bandbreiten und RTT-Werte die resultierenden Werte der wahrgenommenen Latenz zusammengestellt. Die Werte zeigen, dass die höhere Bandbreite erst bei größeren Datenmengen zu spürbaren Verbesserungen führt.

Tabelle 2.1: Latenz				
	1ms RTT		100ms RTT	
Größe	1 Mbit/s	10 Mbit/s	1 Mbit/s	10 Mbit/s
1 Byte	1.008	1.0008	100.008	100.0008
1 kByte	9.192	1.8192	108.192	100.8192
1 MByte	8389.60	839.86	8488.61	938.86

2.2.2 Verzögerung-Bandbreite-Produkt

Während die ersten Daten auf dem Weg vom Sender zum Empfänger sind, kann der Empfänger bereits weitere Daten schicken. Ansonsten, wenn der Sender beispielsweise auf eine Empfangsbestätigung wartet, ist der Kanal lediglich schlecht ausgenutzt. Die Anzahl der Daten, die gleichzeitig auf dem Weg sind, wenn der Sender permanent Pakete schickt, berechnet sich als Produkt aus Latenz und Bandbreite. In Bild 2.2 ist das so genannte Verzögerung-Bandbreite-Produkt grafisch als Röhre beziehungsweise Pipeline dargestellt.



© tecCHANNEL

Bild 2.2: Darstellung von Verzögerung-Bandbreite-Produkt als Pipeline.

Ein Kanal mit einer Bandbreite von 10 Mbit/s und einer Latenz von 50 ms nimmt ein Volumen von

$$50 \times 10^{-3} \text{ s} \cdot 10 \times 10^6 \text{ Bit/s} = 500 \times 10^3 \text{ Bit} \approx 62 \text{ kByte} \quad (2.5)$$

auf. Wenn der Empfänger das erste Bit empfängt, sind bereits etwa 62 kByte an Daten unterwegs zu ihm. Signalisiert der Empfänger dem Sender, dass er keine weiteren Daten mehr aufnehmen kann, benötigt der Sender wiederum die Latenzzeit von 50 ms, um reagieren zu können.

Der Empfänger muss daher ausreichend Speicher vorhalten, um die Leitung noch vollständig leeren zu können. Andernfalls – wenn die Daten verloren gehen – müssen sie später erneut gesendet werden.

2.2.3 Anwendungen

In der Praxis sind die Anforderungen von Anwendungen an die Leistungsfähigkeit des Übertragungskanal sehr unterschiedlich. Häufig schwankt die Anforderung auch noch stark während der Verbindungsdauer. So benötigt man beispielsweise beim Surfen im Internet während der meisten Zeit beim Lesen und Anschauen nur eine geringe Bandbreite.

Demgegenüber kann etwa beim Wechsel zu einer neuen Online-Seite kurzfristig sehr viel mehr Bandbreite notwendig sein, um beispielsweise aufwendige Bilder oder Videos schnell zu laden.

Am anspruchsvollsten im Hinblick auf die Netzwerkanforderungen sind jedoch Anwendungen, die einen kontinuierlichen Datenstrom mit engen Randbedingungen für die Verzögerung erfordern.

Ein klassisches Beispiel ist das Telefonnetz. Die erforderliche Datenrate ist zwar gering (64 kbit/s bei ISDN, 13 kbit/s bei GSM), aber die Daten müssen ohne große Verzögerung und für die Dauer der Verbindung ohne erkennbare Pausen übertragen werden. Aus diesem Grund werden Telefonverbindungen bis heute in großen Teilen mit leitungsvermittelnden Netzwerken realisiert. In Tabelle 2.2 sind für einige Anwendungen die erforderlichen Datenraten angegeben.

Tabelle 2.2: Datenraten

Anwendung	Datenrate
Sprachübertragung (Telefonqualität)	64 kbit/s
Audiosignale (hohe Qualität)	1-2 Mbit/s
Videosignale (komprimiert)	2-10 Mbit/s
Video	1-2 Gbit/s

Angaben nach [4]

Neben der absoluten Verzögerung kann auch die relative Schwankung eine Rolle spielen. Bei einer Videoanwendung ist beispielsweise die absolute Verzögerung nicht entscheidend. Ob das Video nach 50 ms oder 200 ms beginnt, ist lediglich ein geringer Unterschied. Aber nach dem Starten des Videos müssen die einzelnen Bilder immer rechtzeitig im erforderlichen Takt (zum Beispiel 30 Bilder pro Sekunde) geliefert werden. Bei einem paketvermittelnden Netzwerk kann es zu Schwankungen der Laufzeit – so genanntem Jitter – kommen. Im Extremfall überholt ein Paket das vor ihm gestartete Paket. Beim Empfänger kommt in solchen Fällen das Abspielen des Videos ins Stocken. Wenn die absolute Verzögerung keine große Rolle spielt, kann der Empfänger das Problem umgehen, indem er zunächst einige Bilder speichert und das Video erst mit einer entsprechenden Verzögerung abspielt. Dann können bei eventuellen Verzögerungen vermehrt Bilder aus dem Speicher wiedergegeben werden, bis der Sender neue Bilder schickt.

2.3 Netzwerktypen und -topologien

2.3.1 Größe

Zwei wichtige Kategorien sind lokale Netze (LAN, local area network) und Fernnetze (WAN, wide area network). Ein LAN ist ein geschlossenes Netz innerhalb eines Gebäudes oder Firmengeländes. Die Reichweite ist damit auf wenige Kilometer beschränkt. Die Anzahl der angeschlossenen Teilnehmer ist bekannt. Weiterhin kann der Betreiber die eingesetzte Technik festlegen. Diese Faktoren vereinfachen das Netzwerkmanagement erheblich.

Ein WAN umfasst demgegenüber einen sehr großen Bereich – beispielsweise ein ganzes Land oder sogar einen Kontinent. Die Signallaufzeiten können hier daher unter Umständen sehr groß werden.

In der Regel ist das Netz offen. Ständig werden neue Teilnehmer angeschlossen oder auch vorhandene Teilnehmer wieder abgemeldet. Für die Benutzung des Netzes müssen die Teilnehmer Gebühren bezahlen.

Ein öffentliches Netz ist gerade dadurch gekennzeichnet, dass jeder es auch benutzen darf, der die technischen Voraussetzungen für einen Netzanschluss erfüllt

und die Gebühren bezahlt. Der Betrieb eines solchen Netzes erfordert daher eine klare Regelung der technischen und rechtlichen Fragen. Durch Standards wird sicher gestellt, dass die in aller Regel sehr heterogenen Endgeräte von verschiedenen Herstellern mit dem Netz zusammen funktionieren.

Neben diesen beiden Grundtypen unterscheidet man noch zwischen verschiedenen Mischformen. Häufig benutzt man die Kategorie Stadtnetz (MAN, metropolitan area network). Das weltweite Netz kann man als Netzwerkverbund mehrerer WANs oder als ein GAN (global area network) sehen. In [4] finden Sie weiterhin die Definitionen

- PAN (personal area network) oder auch Piconetz für die Vernetzung eines Arbeitsplatzes
- SAN (storage / system area network) für die Vernetzung eines Raumes – typischerweise ein Rechenzentrum

Bezüglich der Offenheit lässt sich weiterhin unterscheiden zwischen

- Internet: das öffentliche und für alle offene Netz
- SAN (storage / system area network) für die Vernetzung eines Raumes – typischerweise ein Rechenzentrum
- Intranet: ein Netzwerk, basierend auf der gleichen Technik wie das Internet, aber nur für einen geschlossenen Benutzerkreis (zum Beispiel Mitarbeiter einer Firma, Angehörige einer Hochschule) zugänglich
- Extranet: ein Intranet, das ausgewählten, externen Benutzern (beispielsweise Kunden) einen Zugang ermöglicht

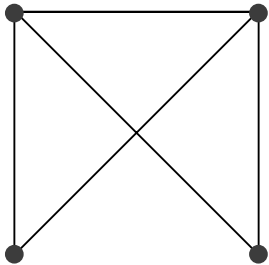
2.3.2 Topologien

Bei dem Entwurf eines Netzes gibt es zahlreiche Möglichkeiten, die einzelnen Rechner zu verbinden. Dies betrifft sowohl die physikalische Verkabelung als auch die logische Struktur. So kann es beispielsweise sinnvoll sein, bestimmte Teile des Netzes gegen andere abzuschirmen, um etwa den Zugriff auf sensible Daten zu kontrollieren. Für die Verkabelung spielen Fragen wie die erforderlichen Bandbreiten und einzuhaltenden Verzögerungen, aber auch wirtschaftliche Überlegungen eine Rolle. Innerhalb eines Netzwerks gibt es einige Standardstrukturen oder Topologien für Netzwerke. Im Folgenden stellen wir diese Standardtopologien kurz vor. In der Praxis wird man selten die reinen Formen, sondern durch Kombination verschiedener Typen Mischformen realisieren.

Punkt-zu-Punkt-Topologie

Beim Maschennetz sind je zwei Endpunkte direkt verbunden. Dadurch ist eine sehr schnelle Kommunikation zwischen Hosts möglich. Darüber hinaus können einzelne Verbindungen auch an betreffende Geräte angepasst werden. Insofern wäre eine komplette Vermaschung aller Rechner innerhalb eines Netzes also die leistungsfähigste Lösung. Allerdings ist dies bei wachsender Anzahl von Endge-

räten nicht mehr praktikabel. Bei N Endgeräten braucht jedes Endgerät $N-1$ Anschlüsse, und insgesamt hätte man $N*(N-1)$ Leitungen. Daher wird meist nur eine teilweise Vermaschung durchgeführt. Nicht direkt verbundene Endgeräte kommunizieren dann über Zwischenstationen.

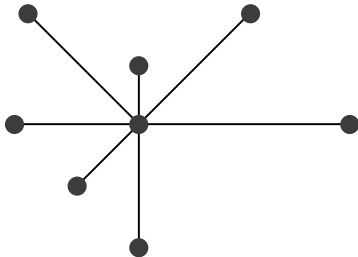


© tecCHANNEL

Bild 2.3: Ein teilweise vermaschtes Netz ist bei einer wachsenden Anzahl von Endgeräten nicht mehr praktikabel.

Sterntopologie

Eine Topologie mit sehr einfacher Struktur erhält man durch sternförmige Anordnung der Endgeräte um einen zentralen Knoten. Jedes Endgerät kommuniziert dann direkt nur noch mit dem Zentralknoten. Der Zentralknoten muss entsprechend leistungsfähig sein. Sinnvoll kann es sein, an den Zentralknoten Komponenten wie Fileserver oder Backup-Geräte anzuschließen.



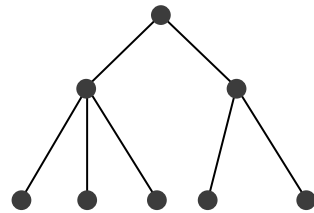
© tecCHANNEL

Bild 2.4: Beim Sternnetz kommunizieren die Endgeräte über einen Knoten.

Baumtopologie

Verwandt mit der Sterntopologie ist die Baumtopologie. Im Unterschied zur Sternvariante sind dabei aber nicht alle Endgeräte an einen einzigen Zentralknoten angeschlossen, sondern es gibt hier auch Zwischenknoten, die jeweils mehrere Endgeräte bedienen können.

Bild 2.5: Bei der Baumtopologie gibt es Zwischenknoten, an die jeweils mehrere Endgeräte angeschlossen werden können.

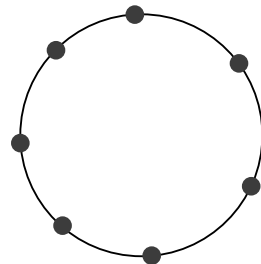


© tecCHANNEL

Ringtopologie

Eine Topologie mit gleichmäßiger Arbeitsteilung – zumindest bezüglich der Kommunikation – ist die Ringtopologie. Dabei ist jeder Knoten nur mit zwei anderen Knoten verbunden. Jedes Datenpaket wird dabei in jedem Knoten gelesen und – falls es nicht für diesen Knoten bestimmt ist – weiter gereicht. Da auf jeder Leitung nur jeweils ein Knoten sendet und empfängt, hat man gute Kontrolle über die Leitung und kann einen hohen Durchsatz erzielen. Nachteilig ist die hohe Abhängigkeit des Gesamtsystems von jeder einzelnen Leitung.

Bild 2.6: Im Ringnetz ist jeder Knoten nur mit zwei anderen Knoten verbunden.



© tecCHANNEL

Bustopologie

In den bisher besprochenen Topologien verbindet eine Leitung stets zwei Knoten. Eine grundsätzliche Alternative ist die Nutzung einer gemeinsamen Leitung – dann auch als Bus bezeichnet. Jeder Knoten wird direkt mit diesem Bus verbunden. Um einen anderen Knoten zu erreichen, schickt ein Knoten eine Nachricht auf den gemeinsamen Bus. Alle Knoten müssen ständig die Daten auf dem Bus beobachten und die an sie adressierten Nachrichten aufnehmen. Eine Stärke dieser Anordnung ist die effiziente Realisierung von Broadcast-Nachrichten, das heißt Nachrichten, die an mehrere oder sogar alle Knoten im Netz gerichtet sind. Ein derartiges System stellt eine Reihe von besonderen Anforderungen. So muss man mit entsprechenden Protokollen sicherstellen, dass nicht zwei Knoten gleichzeitig senden oder ein Knoten den Bus dauerhaft belegt.

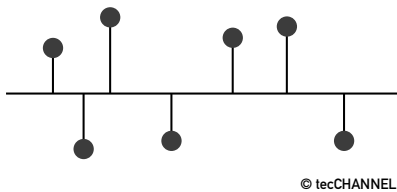


Bild 2.7: Bei der Bustopologie müssen alle Knoten ständig die Leitung überwachen, um an sie adressierte Daten aufzunehmen.

2.4 Referenzmodelle

2.4.1 Protokollstapel

In vielen Fällen kommunizieren Sender und Empfänger nicht direkt, sondern über Zwischenstufen. Betrachten wir folgendes Beispiel: Herr A. möchte zwölf Flaschen Rotwein von Chateau Rothschild in Bordeaux kaufen. Seine Nachricht kann man wie folgt darstellen:

[Herr A.] [12 Flaschen Rotwein] [Chateau Rothschild]

Herr A. spricht weder Französisch, noch kennt er die Anschrift von Chateau Rothschild. Deshalb wendet er sich an seinen Weinhändler als Vermittler. Der Weinhändler kennt einen Großhändler in Bordeaux, spricht selbst allerdings auch kein Französisch. Er übersetzt die Bestellung in Englisch und schickt folgende Nachricht an den Großhändler:

[Herr A.] [12 bottles red wine] [Chateau Rothschild] [Großhändler in Bordeaux]

Der französische Großhändler entfernt seine eigene Adresse, übersetzt in Französisch und leitet die Bestellung weiter.

[Herr A.] [12 bouteilles vin rouge] [Chateau Rothschild]

Der gesamte Ablauf der Bestellung ist in Bild 2.8 dargestellt. Dabei ist als zusätzliches Element die Übertragung der Nachricht als Brief per Post eingetragen. Das Bild zeigt, wie die eigentliche Kommunikation über mehrere Zwischenstufen abläuft. Auf diesem Weg wird die Nachricht zwei Mal übersetzt. Wichtig ist, dass die einzelnen Schritte unabhängig voneinander erfolgen. Für die Bestellung ist es unwesentlich, ob die beiden Händler sich in Englisch oder einer anderen Sprache verständigen, solange sie sich auf eine gemeinsame einigen können. Entsprechend brauchen die beiden Endbenutzer die Übertragungssprache nicht zu verstehen.

Für jede Schnittstelle muss nur zwischen den beiden beteiligten Partnern ein Protokoll verabredet werden. Die gesamte Übertragung beinhaltet eine Kette von Protokollen – den so genannten Protokollstapel (Englisch: protocol stack).

Auf der Sendeseite wird die Protokollkette von oben nach unten abgearbeitet, bis die Nachricht physikalisch verschickt wird. Beim Empfänger durchläuft die Nachricht die Kette in umgekehrter Reihenfolge. Typisch ist dabei, dass auf der Sendeseite die unteren Ebenen zusätzliche Informationen anfügen (im Beispiel die Adresse des französischen Großhändlers), die dann beim Empfänger auf der gleichen Ebene wieder entfernt werden.

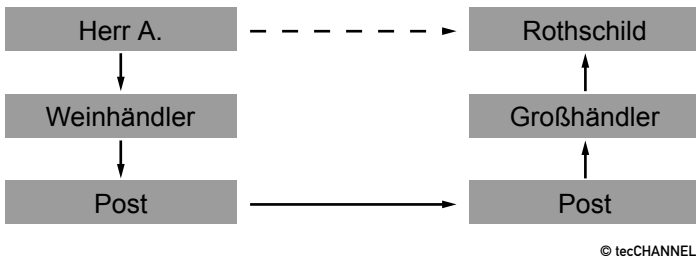


Bild 2.8: Ablauf Rotwein-Bestellung

Diese Situation – die beiden Kommunikationspartner verständigen sich indirekt über eine Reihe von Zwischenschichten – stellt die große Herausforderung für heterogene Netzwerke dar. Ein typisches Beispiel ist die Kommunikation zwischen einem Webbrowser, der auf einem PC unter Windows 2000 läuft, und einem Webserver auf einem Großrechner mit Linux. Trotz aller Unterschiede in Hard- und Software können die beiden Anwendungen sich verständigen.

2.4.2 OSI-Referenzmodell

Zum Verständnis der Abläufe in Netzwerken ist eine standardisierte Darstellung der verschiedenen Protokollebenen hilfreich. Eine klare Definition funktionaler Ebenen erlaubt eine starke Kapselung. Die Schnittstelle zwischen zwei Ebenen kann dann leicht ausgetauscht werden, ohne dass es zu Rückwirkungen auf die anderen Protokolle kommt.

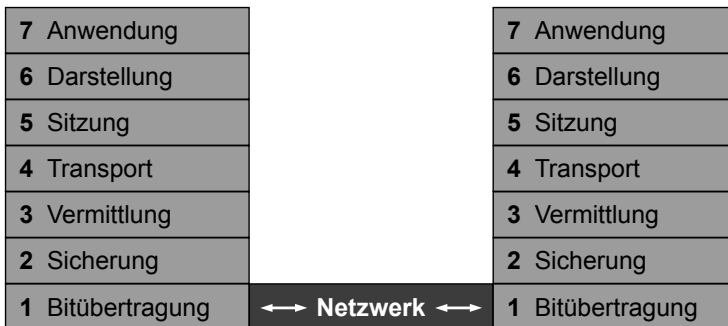


Bild 2.9: Schichtenreferenzmodell ISO/OSI 7

Eine formale Definition der verschiedenen Schichten entwickelte die ISO (International Standards Organization) ab etwa 1977. Resultat ist das Schichtenmodell ISO/OSI 7 (OSI: Open Systems Interconnection).

Das Bild 2.9 zeigt die sieben Schichten des Referenzmodells. In diesem Modell sind weiterhin die Schnittstellen zwischen den Schichten festgelegt. Man unterscheidet zwischen Protokollen und Diensten. Protokolle definieren die Schnittstelle zwischen zwei Systemen auf einer Ebene. Demgegenüber sind in der OSI-Terminologie aber Dienste Funktionen, die eine Schicht der nächsthöheren Schicht zur Verfügung stellt.

Bitübertragung

Die Bitübertragungsschicht (physical layer) stellt einen Bitstrom zwischen Sender und Empfänger bereit. Auf dieser Ebene spielt die Bedeutung der Bits keine Rolle. Vielmehr geht es darum, wie die einzelnen Bits übertragen werden. Darunter fallen Fragen wie Übertragungsmedien, Stecker, Darstellung einzelner Bits, Aufbau einer Verbindung et cetera.

Sicherung

Bei der Bitübertragung können Fehler auftreten. Aufgabe der Sicherungsschicht (data link layer) ist es, solche Fehler zu erkennen und durch entsprechende Mechanismen zu korrigieren. Dazu werden Blöcke von Datenbits zusammen gefasst. Die so gebildeten Datenrahmen (data frames) werden mit zusätzlichen Bits zur Kennung von Anfang und Ende versehen. Weitere Kontrollbits erlauben dem Empfänger, die korrekte Übertragung des Rahmens zu überprüfen. Erkennt der Empfänger Fehler, kann er sie bis zu einer gewissen Grenze korrigieren. Ist der Rahmen so gestört, dass eine Korrektur nicht mehr möglich ist, fordert der Empfänger den Sender auf, den Rahmen erneut zu senden.

Vermittlung

Der Weg, den die Daten nehmen sollen, wird in der Vermittlungsschicht (network layer) festgelegt. Diese Festlegung kann beispielsweise zu Beginn einer Verbindung erfolgen. Alternativ kann auch für jedes Paket neu die gerade optimale Route bestimmt werden. Man spricht auf der Vermittlungsschicht nicht mehr von Rahmen, sondern von Paketen. Die Vermittlungsschicht enthält in der Regel auch Abrechnungsfunktionen, um die entstandenen Gebühren zu erfassen. Wenn auf dem Weg vom Sender zum Empfänger mehrere Knoten liegen, werden in jedem der Knoten die drei untersten Schichten benötigt.

Transport

Die Transportschicht (transport layer) ist die erste Ende-zu-Ende-Schicht. Das heißt, das Programm in der Transportschicht der Sendeseite kommuniziert mit einem passenden Gegenstück auf der Empfängerseite. Die Aufgabe der Transportschicht ist es, größere Datenmengen von den oberen Schichten zu übernehmen, sie, falls notwendig, in kleinere Einheiten zu zerlegen und dann an die Vermittlungsschicht weiterzugeben. Die Transportschicht ist auch noch zuständig für den Aufbau der Verbindung.

Sitzung

Die Sitzungsschicht (session layer) bietet eine übergeordnete Sicht der Verbindung. Eine Sitzung kann verschiedene Verbindungen beinhalten, zum Beispiel für Hin- und Rückkanal, oder in zeitlicher Folge, wenn die Verbindung nicht permanent gehalten wird.

Darstellung

In der Darstellungsschicht (presentation layer) werden die von unten gelieferten Daten interpretiert und in die für das System richtige Darstellung gebracht. Beispielsweise werden für die Übertragung Standardrepräsentationen für Daten wie Zeichen, Festkommazahlen et cetera definiert. In der Darstellungsschicht erfolgt dann die Umsetzung aus der allgemeinen Repräsentation in das maschinenspezifische Format. Zu den weiteren Aufgaben in dieser Schicht gehören Verschlüsselung und Datenkompression.

Anwendung

Die Anwendungs- oder Verarbeitungsschicht (application layer) stellt dem Benutzer bestimmte Dienste zur Verfügung. Dazu gehören der Dateitransfer, das Anmelden auf anderen Rechnern oder die Kontrolle von Prozessen.

2.4.3 Bedeutung des OSI-Referenzmodells

Das ISO/OSI-7-Schichtenmodell bietet eine gute Grundlage zum Verständnis sowie zur vergleichenden Beurteilung von eingesetzten Technologien. Die gewählte Einteilung in sieben Schichten sollte allerdings nicht zu starr gesehen werden. In vielen Fällen erweist sich die Einteilung als teilweise zu fein oder zu grob. Die höheren Schichten sind oft in weniger Schichten realisiert. Beispielsweise hat die im Internet verwendete TCP/IP-Architektur insgesamt nur vier Schichten. Tanenbaum benutzt in seinem Standardwerk [5] ein vereinfachtes Modell mit fünf Schichten. Umgekehrt ist es sinnvoll, bei genauerer Untersuchung der unteren Schichten weitere Zwischenschichten einzuführen.

In der Praxis hat das Modell nie die von den Initiatoren erhoffte universelle Bedeutung erlangt. Durch den aufwendigen und zeitraubenden Standardisierungsprozess setzten sich oft am Markt mehr pragmatisch entstandene, aber frühzeitig verfügbare Systeme durch. Universitäten als Vorreiter neuer Techniken haben eine andere Kultur als die ITU/ISO-Gremien. Hier gilt mehr das Interesse an funktionierenden Programmen als an detailgenauen Spezifikationen („Grober Konsensus und laufender Code“).

Dieses tecCHANNEL-Compact orientiert sich daher nicht streng an dem ISO/OSI-7-Schichtenmodell. Vielmehr liegt der Schwerpunkt auf einzelnen Themen, die sich teilweise über mehrere Schichten ziehen. Im Zweifelsfall wird die inhaltliche Nähe stärker berücksichtigt als die Zuordnung zu den Schichten. Die folgenden vier Kapitel behandeln Rechnernetze zunehmender Komplexität. Im Einzelnen lassen sich die Stufen

1. direkte Verbindung zweier Rechner
2. Kommunikation in einem lokalen Netz
3. Verbindung mehrerer lokaler Netze

unterscheiden. Anhand der Verbindung zweier Rechner werden Grundlagen wie die Bildung von Rahmen oder das Einfügen von Sicherungsinformationen dargestellt. Die Anforderungen, die sich aus dem Zusammenschluss mehrerer Rechner ergeben, diskutieren wir zunächst für lokale Netze. Schließlich wird die Verbindung vieler einzelner lokaler Netze unterschiedlichster Art zu einem großen Netzwerk – dem Internet – beschrieben.

Prof. Dr. Stephan Euler

tecCHANNEL-Links zum Thema	Webcode	Compact
Netzwerk-Grundlagen	a1492	S.11
Metro Ethernet: Eine Leitung für Daten, Telefon und Video	a1286	–
Von Kibibits und Gibibytes	a848	–

Mit den Webcodes gelangen Sie auf www.tecChannel.de direkt zum gewünschten Artikel. Geben Sie dazu den Code in das Feld „Webcode“ in der Titelleiste von www.tecChannel.de ein. Mit „p“ gekennzeichnete Webcodes beziehen sich auf Online-Artikel aus tecCHANNEL-Compact. Es ist möglich, dass Artikel zunächst exklusiv in tecCHANNEL-Compact veröffentlicht werden und daher erst später über den Webcode erreichbar sind.

3. Rechner-zu-Rechner-Verbindung

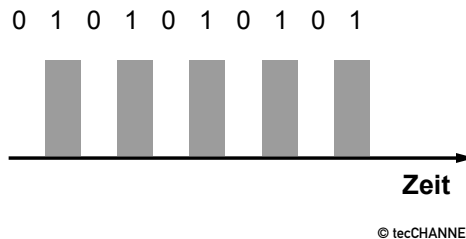
Die einfachste Form eines Netzwerks ist die direkte Verbindung zwischen zwei Computern. Sie eignet sich gut, um die grundlegenden Standards und Anforderungen bei der Datenübertragung zu erläutern.

3.1 Übertragung von Bits

Das einfachste Modell für eine Übertragung ist eine Leitung, auf der ein Signal zwei Zustände annehmen kann. Dieses Signal bewegt sich mit der Ausbreitungsgeschwindigkeit vom Sender zum Empfänger. Die Dauer eines einzelnen Zustandes bestimmt die erreichbare Übertragungsrate. Bei einer Dauer Δt kann man pro Sekunde $1/\Delta t$ Zustände übermitteln.

Bei einer elektrischen Leitung werden die beiden Zustände beispielsweise durch zwei verschiedene Spannungen realisiert. Die beiden Zustände seien als low und high bezeichnet. Über diese Leitung sollen Daten übertragen werden. Dazu muss eine Zuordnung der Bits zu den Zuständen getroffen werden – die Kodierung. Die nahe liegende Vereinbarung ist, für den Wert 1 das Signal high und für 0 low zu übertragen. Bild 3.1 zeigt für eine Folge von 0-1-Werten die Zustände in zeitlicher Folge. Aus den im Folgenden genannten Gründen nennt man diese Art der Kodierung Non-Return to Zero (NRZ).

Bild 3.1: Folge von Bits mit den Werten 0 und 1.



Einen etwas allgemeineren Fall zeigt Bild 3.2. Folgen mehrere gleiche Werte aufeinander, bleibt das Signal auf einem festen Pegel, und es findet kein Wechsel mehr statt. Wenn beispielsweise der Sender eine lange Folge von Nullen schickt, sieht der Empfänger für die entsprechende Zeitdauer einen konstanten Pegel. Dann kann es schwierig werden, exakt zu zählen, wie viele Nullen geschickt wurden. Falls Sender und Empfänger keine gemeinsame Zeitbasis haben, können kleine Abweichungen in der Zeitmessung zwischen beiden dazu führen, dass der Empfänger zu wenige oder zu viele Nullen zählt. Im Normalfall – das heißt, wenn keine zu langen Folgen mit konstanten Werten auftreten – kann der Empfänger kleine Abweichungen kompensieren, indem er die Signalwechsel nutzt, um sich immer wieder auf den Sender zu synchronisieren (Taktregenerierung).

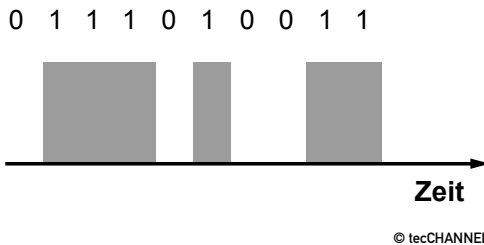


Bild 3.2: Allgemeine Folge von Bits.

Andererseits können Taktfehler weit reichende Folgen haben, wenn man zum Beispiel die Übertragung eines Stroms von Bytes betrachtet. Sobald ein Bit verloren geht, verschiebt sich für alle nachfolgenden Werte die Zuordnung der Bits zu den Bytes, so dass sich komplett andere Byte-Werte ergeben.

Es ist daher sinnvoll, dafür zu sorgen, dass auch bei konstanten Werten Signalwechsel stattfinden. Ein Ansatz dazu ist die Kodierung Non-Return to Zero Inverted (NRZI). Dabei gilt die Regel, dass bei einer 1 der Pegel sich ändert, das heißt von low auf high oder von high auf low springt. Dadurch wird aus der Folge 0 1 1 1 1 die Signalfolge low high low high low. NRZI löst das Problem von vielen aufeinander folgenden Einsen, hilft aber nicht bei langen Folgen von Nullen.

Ein weiter gehender Ansatz ist die Manchester-Kodierung. Man kann sich dabei die Übertragung mit doppelter Rate vorstellen, das heißt für jeden Wert stehen zwei Zeiteinheiten zur Verfügung. Dann überträgt man die Folge low-high für 0 und high-low für 1. So ist gewährleistet, dass für jeden übertragenen Wert mindestens ein Signalwechsel stattfindet. Nachteil ist, dass jetzt für jeden Wert zwei Zeiteinheiten gebraucht werden oder umgekehrt in der gleichen Zeit nur halb so viele Werte übertragen werden können. Von den vier möglichen Signalfolgen werden nur noch zwei verwendet.

Es gibt eine ganze Reihe weiterer Schemata für die Kodierung, die auch auf diesen Ideen beruhen. Der verbreitete 4B5B-Code überträgt vier Bit mit fünf Signalwerten. Die Zuordnung ist dabei so gewählt, dass nie mehr als drei Nullen hintereinander kommen. Zusammen mit einer NRZI-Kodierung ermöglicht der Code deshalb bei nur geringem Mehraufwand (20 Prozent) eine deutliche Verbesserung der Taktregenerierung.

In diesem Zusammenhang wird oft das Maß Baud (Bd) benutzt. Die Einheit ist benannt nach dem französischen Ingenieur Jean Maurice Emile Baudot, der den Baudot-Code – den Vorläufer des ASCII-Codes – entwickelte. Die Anzahl der pro Sekunde erfolgten Übertragungen bezeichnet man als Baudrate.

Wenn mit jeder Übertragung genau ein Bit dargestellt wird, entspricht die Baudrate dem Maß Bit/s. Andererseits sind bei der Manchester-Kodierung zwei Übertragungen pro Bit notwendig. Damit ist die Baudrate dann doppelt so hoch wie die Anzahl der Bits pro Sekunde. Können bei der Übertragung aber mehr als nur zwei Zustände unterschieden werden, ist umgekehrt die Baudrate kleiner als die Zahl der Bits pro Sekunde.

3.2 Rahmenerstellung

Die Übertragung von Bits ist immer bis zu einem gewissen Grad fehleranfällig. Neben den besprochenen Synchronisationsfehlern können einzelne Bits falsch dekodiert werden. Solche Fehler sind in einem kontinuierlichen Datenstrom nicht zu erkennen. Es ist daher wichtig, den Datenstrom zu strukturieren und durch Zusatzinformationen unempfindlich gegen Störungen zu machen. Weiterhin ist bei Rechnernetzen die kontinuierliche Übertragung eher die Ausnahme. Bei typischen Anwendungen (zum Beispiel Surfen im Internet) wechseln sich Phasen von Datenübertragungen – aus Sicht des Kommunikationskanals – mit Ruhephasen ab. Daher ist der Beginn einer neuen Übertragung kenntlich zu machen.

Aus diesen Gründen werden mehrere Bits oder Bytes zusammengefasst, um entsprechende Informationen ergänzt und als Einheit übertragen. Man spricht dann von Rahmen oder Frames. Dabei ist zwischen Byte-orientierten und Bit-orientierten Protokollen zu unterscheiden, je nach dem, ob Bytes oder Bits als kleinste Bausteine für einen Frame zum Einsatz kommen.

Bei klassischen Kommunikationsanwendungen wie Telefonie ist die Übertragungsrate fest, so dass sich der Datenstrom gut in eine Reihe gleich großer Rahmen aufteilen lässt. Wenn eventuell am Ende der Übertragung ein Rahmen nicht mehr komplett gefüllt wird, kann man problemlos mit Nullen auffüllen. Die dadurch entstehende kleine Pause stört nicht. Demgegenüber ist die Rechnerkommunikation durch starke Schwankungen in der Übertragungsrate gekennzeichnet. Außerdem ist das Auffüllen des letzten Rahmens beispielsweise beim Kopieren einer Datei nicht akzeptabel. Daher werden in der Regel Rahmenformate für eine variable Anzahl von Daten eingesetzt. Hierzu muss der Sender dem Empfänger mitteilen, wie viele Daten im aktuellen Rahmen enthalten sind. Zwei Verfahren werden dazu benutzt: Verwendung einer Endmarkierung oder explizite Angabe der Anzahl der Daten. Zusammengefasst lassen sich daher folgende Methoden der Rahmenbildung unterscheiden:

- Feste Anzahl von Nutzdaten
- Variable Anzahl von Nutzdaten
- Markierung des Endes der Nutzdaten
- Angabe der Anzahl der Nutzdaten im aktuellen Frame

Im Folgenden diskutieren wir am Beispiel von Byte-orientierten Protokollen die beiden Verfahren zur Erstellung von Rahmen mit einer variablen Anzahl von Daten-Bytes.

3.2.1 Markierungszeichen

Bei der Übertragung von Texten verwendet man die im ASCII-Standard definierten Steuerzeichen zum Aufbau von Rahmen. Ein einfacher Rahmen könnte dann folgende Form haben (Beispiel nach [5]):

[DLE] [STX] [Daten] [DLE] [ETX]

Die zeitliche Abfolge ist bei dieser Darstellung von links nach rechts, das heißt, das am weitesten links stehende Element wird zuerst gesendet. Das Zeichen DLE (Data Link Escape, ASCII Code 127) markiert den Beginn einer Steuersequenz. Anfang und Ende werden durch STX (Start of TeXt, ASCII Code 2) beziehungsweise ETX (End of TeXt, ASCII Code 3) gekennzeichnet. Eingebettet zwischen diesen beiden Steuersequenzen liegen die Nutzdaten.

Dieses Verfahren funktioniert bei der Übertragung von Textdaten problemlos. Bei beliebigen Binärdaten – Gleitkommazahlen, Bilder oder Ähnliches – besteht allerdings die Möglichkeit, dass die Endsequenz DLE ETX auch in den Nutzdaten auftritt:

[DLE] [STX] [...] [DLE] [ETX] [...] [DLE] [ETX]

Der Empfänger würde dann vorzeitig das Ende des Rahmens erkennen. Ein ähnliches Problem ist bei der Programmierung in C und verwandten Sprachen die Darstellung des Zeichens „, als Teil einer Zeichenkette. Eine Lösung besteht in der Einfügung eines zusätzlichen Zeichens. Man kann etwa in den Daten vor jedem DLE ein zweites DLE einfügen:

[DLE] [STX] [...] [DLE] [DLE] [ETX] [...] [DLE] [ETX]

Durch dieses so genannte Zeichenstopfen (Englisch: character stuffing) wird sichergestellt, dass innerhalb des Datenfeldes nie ein einzelnes DLE erscheint. Vielmehr folgen stets zwei oder – allgemein betrachtet – eine gerade Anzahl von DLE-Zeichen aufeinander. Der Empfänger muss dann nur aus jedem Paar von DLE-Zeichen eines entfernen, um die gesendete Byte-Folge zu rekonstruieren.

3.2.2 Zeichenzähler

Die zweite Methode zur Rahmenerzeugung beruht auf der Angabe der Anzahl der nachfolgenden Daten. Im einfachsten Fall wird ein Feld mit einem Zähler vor den Daten eingefügt:

[DLE] [STX] [Anzahl] [Daten]

Im dem Anzahlfeld steht explizit, wie viele Bytes Daten in diesem Rahmen noch kommen werden. Die Größe des Nutzfeldes ist dann durch die Größe des Anzahlfeldes beschränkt. Steht nur ein Byte für die Größenangabe zur Verfügung, so kann das Nutzfeld demnach maximal 256 Bytes lang sein. In dieser einfachsten Form ist die Übertragung recht fehleranfällig.

Tritt ein Fehler im Anzahlfeld auf, berechnet der Empfänger die Rahmengröße falsch. Es kann unter Umständen lange dauern, bis der Empfänger sich wieder auf einen korrekten Rahmenanfang synchronisiert. Man verwendet daher in der Regel den Zeichenzähler nur in Kombination mit anderen Methoden.

3.3 Fehlererkennung

Im Allgemeinen ist die Übertragung über einen Kanal nicht fehlerfrei. Ein Rahmen kann daher fehlerhafte Daten enthalten. Wie oben beschrieben, können Fehler im Extremfall sogar dazu führen, dass das Rahmenende nicht richtig erkannt und damit auch der Empfang der nachfolgenden Rahmen gestört wird. Deshalb ist es wichtig, dass der Empfänger durch Analyse des Rahmens feststellen kann, ob Übertragungsfehler auftraten.

Bei optimaler Ausnutzung der Übertragungskapazität entspricht jedem Bitmuster ein gültiges Zeichen. Durch Übertragungsfehler werden dann aus gültigen Zeichen auch wieder „gültige“ Zeichen. Der Empfänger kann dann nicht erkennen, ob Übertragungsfehler vorliegen. Um eine Fehlererkennung zu ermöglichen, muss die Übertragung so erweitert werden, dass Fehler zu ungültigen Zeichen (oder Rahmen) führen. Dazu fügt der Sender zusätzliche Informationen (Redundanz) in den Rahmen ein.

Eine Anwendung dieses Prinzips sind die Prüfzeichen bei wichtigen oder leicht zu verwechselnden Informationen wie Konto- und Kreditkartennummern oder die Internationale Standard Buchnummer (ISBN). Diese Nummern enthalten ein zusätzliches Zeichen, das nach einem vorgegebenen Algorithmus aus den anderen Stellen berechnet wird.

Auch in der sprachlichen Kommunikation setzen wir in schwierigen Fällen Redundanz ein. Ein Beispiel ist das Buchstabieren bei der Übermittlung von Namen. Noch mehr Redundanz kann man durch die Verwendung von Buchstabieralphabeten wie A wie Anton, B wie Berta ... erreichen. Die Eigennamen sind als längere Einheiten besser zu erkennen als die einzelnen Buchstaben.

Oft lassen sich nicht erkannte Fehler in der unteren Schicht auf einer höheren Schicht noch feststellen. Bei der Übermittlung einer Textdatei führen Übertragungsfehler mit einiger Wahrscheinlichkeit dazu, dass erkennbar falsche Wörter daraus resultieren. Diese Fehlererkennung ist aber nur auf Grund der Redundanz der Sprache möglich. Nicht jede Buchstabenfolge ergibt ein Wort in der jeweiligen Sprache, so dass Sprache auch ein Maß an Redundanz enthält.

Die Zusatzinformation als Sicherung gegen Fehler verringert die Nettoübertragungsrate. Daraus ergibt sich die Herausforderung, mit möglichst wenig Zusatzinformation eine zuverlässige Fehlererkennung zu ermöglichen.

Man unterscheidet dabei:

- Fehlererkennung: Der Empfänger kann erkennen, dass ein Rahmen fehlerhaft übertragen wurde.
- Fehlerkorrektur: Der Empfänger kann bis zu einem gewissen Grad erkannte Übertragungsfehler auch korrigieren.

Kommt nur eine Fehlererkennung zum Einsatz, muss der Empfänger im Fehlerfall den Sender auffordern, den Rahmen erneut zu schicken. Will man eine Fehlerkorrektur realisieren, muss mehr Sicherheitsinformation in den Rahmen einge-

baut werden. Trotzdem ist auch damit nur ein Teil der Fehler zu reparieren. Enthält ein Rahmen zu viele Fehler, dann lässt sich die ursprüngliche Information nicht wieder herstellen.

Welches Verfahren effizienter in Hinblick auf die Sicherheit und die insgesamt erreichbare Nettodatenrate ist, hängt von dem Übertragungskanal ab. Dabei spielen sowohl die Auftrittswahrscheinlichkeit für einen Bitfehler als auch die statistische Verteilung der Fehler eine Rolle. Bei vielen Kanälen treten Fehler nicht gleichmäßig verteilt auf, sondern in Gruppen (Bursts). Wenn der Kanal in einem schlechten Zustand ist – zum Beispiel bei einer Funkverbindung in einem „Schatten“ hinter einem Hochhaus – kommt es zu sehr viel mehr Fehlern als im Normalzustand. Dann scheitert die Fehlerkorrektur, und die entsprechenden Rahmen müssen verworfen werden.

Insgesamt ist die Fehlererkennung eine komplexe Aufgabe. Die eingesetzten Methoden erfordern ein tiefes Verständnis von Statistik und Informationstheorie. Letztere erlaubt auch Aussagen zu den theoretischen Grenzen für die Nachrichtenübertragung über einen gegebenen Kanal.

Im Folgenden werden ohne Anspruch auf mathematische Tiefe anhand von zwei wichtigen Beispielen einige grundsätzliche Prinzipien der Fehlererkennung dargestellt. Bei diesen beiden Verfahren bleiben die Daten unverändert und werden um zusätzliche Sicherungsdaten ergänzt.

3.3.1 Parität

Eine weit verbreitete Methode zur Fehlererkennung ist das Einfügen von Paritätsbits. Dazu zählt man die Bits mit Wert 1 in dem Datenwort und fügt dann ein weiteres Bit hinzu, so dass bei der so genannten geraden Parität (even parity) insgesamt eine gerade Anzahl von Einsen entsteht. Bei ungerader Parität (odd parity) ist die Anzahl der gesetzten Bits ungerade. Das folgende Beispiel zeigt, was man für das ASCII-Zeichen C bei gerader Parität erhält.

Tabelle 3.1: Gerade Parität für das ASCII-Zeichen C

Zeichen	Bitmuster	Paritätsbit
C (0x43)	100 0011	1

Das zusätzliche Paritätsbit eignet sich sehr gut für ASCII-Zeichen. Wenn man sich auf den 7-Bit-Zeichensatz beschränkt, dann kann das achte Bit als Paritätsbit verwendet werden. Die Parität hat weiterhin den Vorteil, dass sich die Prüfung auch in Hardware ohne großen Aufwand mit hintereinander geschalteten XOR-Elementen realisieren lässt.

Von den mit acht Bit möglichen 256 Zeichen bleiben nach der Paritätsprüfung nur noch 128 gültige Zeichen übrig. Dabei gilt, dass sich zwei gültige Zeichen in min-

destens zwei Bitpositionen unterscheiden. Ändert man nur ein Bit, so resultiert daraus ein ungültiges Zeichen. Man sagt, der Hamming²-Abstand beträgt 2. Allgemein bedeutet ein Hamming-Abstand n , dass sich zwei Zeichen in mindestens n Positionen voneinander unterscheiden.

Mit einem Paritätsbit wird ein Fehler (oder allgemein eine ungerade Anzahl von Fehlern) innerhalb eines Bytes erkannt. Zwei Fehler kompensieren sich aber gegenseitig und führen wieder zu einem korrekten Paritätsbit. Betrachtet man einen Block – das heißt eine Anzahl aufeinander folgender Zeichen – gemeinsam, lässt sich die Sicherheit deutlich erhöhen. Dazu wird zu der beschriebenen Parität pro Zeichen eine zweite Parität pro Position in Zeitrichtung eingeführt. Zur Unterscheidung zur einfachen (eindimensionalen) Parität spricht man auch von zweidimensionaler Parität. Die Tabelle zeigt die zweidimensionale Parität am Beispiel der Zeichenfolge HALLO. Die Parität in Zeilen- und Spaltenrichtung heißt Quer- und Längsparität.

Tabelle 3.2: Zweidimensionale Parität

Zeichen	Bitmuster	Paritätsbit
H (0x48)	100 1000	0
A (0x41)	100 0001	0
L (0x4C)	100 1100	1
L (0x4C)	100 1100	1
O (0x4F)	100 1111	1
Längsparität	100 0110	1

Ein einzelner Fehler macht sich sowohl in einer Zeile als auch Spalte bemerkbar und ist damit korrigierbar. Zwei Fehler – selbst in unterschiedlichen Zeilen und Spalten – sind aber nicht mehr auszubessern. Allerdings lassen sich die Fehler relativ gut lokalisieren, und es gibt nur zwei Möglichkeiten, wo sie sich befinden können. Liegen die beiden Fehler zusätzlich in einer Zeile, stimmen alle Quersparitäten. Die Fehler werden zwar durch die Längsparität detektiert, können aber nicht mehr korrigiert werden. Allgemein lassen sich alle 3-Bit-Fehler und auch die meisten 4-Bit-Fehler erkennen. Nur wenn die vier Fehler genau an den passenden Kreuzungspunkten liegen, stimmen wieder alle Quer- und Längsparitäten.

Die Paritätsprüfung zählt die vorkommenden Einsen und prüft, ob sich in Summe eine gerade Anzahl ergibt. Die Übertragung umfasst einen Block Daten und eine Anzahl von Paritätsinformationen:

[Datenbits] [Paritätsbits]

Der Empfänger kann dann durch die Prüfoperation auf dem Gesamtblock die Integrität testen. Dieses allgemeine Schema „Daten – Prüfinformation – Prüfoperation“ lässt sich leicht verallgemeinern, indem man andere Prüfoperationen nutzt.

In Internet-Protokollen wird ein Verfahren mit einer Prüfsumme angewandt. Die Prüfoperation ist dabei eine Addition im Einerkomplement. Als Prüfinformation wird das Ergebnis dieser Addition an die Daten angehängt. Der Empfänger führt dann ebenfalls die Addition aus und vergleicht das Ergebnis mit der Prüfsumme. Dieses Verfahren ist einfach zu realisieren, bietet aber nicht sehr viel Schutz. Wesentlich mächtiger ist das Verfahren der zyklischen Redundanzprüfung.

3.3.2 Zyklische Redundanzprüfung

Bei dem Verfahren der zyklischen Redundanzprüfung (Cyclic Redundancy Check CRC) ist die Prüfoperation im Prinzip eine Division. Betrachten wir zunächst ein Beispiel mit Dezimalzahlen. Übertragen werden soll eine große Dezimalzahl wie etwa 35628828297292. Weiterhin wählt man einen Divisor D aus. Nimmt man dafür beispielsweise den Wert 17 und führt die Division aus, so erhält man bei ganzzahliger Division (Modulo-Operator % in C) einen Rest von 8. Der Sender kann damit zu der großen Zahl als Prüfwert die 8 senden. Der Empfänger rechnet dann zur Kontrolle $(35628828297292-8)\%17$. Bleibt bei der Division ein Rest übrig, so liegt ein Übertragungsfehler vor.

Schon an diesem einfachen Beispiel werden einige Grundeigenschaften deutlich. Zunächst einmal sind nicht alle Divisoren gleich gut geeignet für diesen Zweck. Beispielsweise wäre 10 ein schlechter Divisor, da sich dann nur Fehler an der letzten Stelle auswirken würden.

Weiterhin wird ein größerer Divisor einen besseren Schutz bieten. Allerdings muss dann entsprechend mehr Platz für den Rest vorgehalten werden. Der Rest kann maximal den Wert D-1 annehmen. Der Divisor selbst hat sinnvollerweise einen festen Wert und braucht dann nicht mehr übertragen zu werden.

Die Übertragung dieses Ansatzes auf Binärdaten beruht auf der Interpretation von Bitfolgen als Polynomen. Für eine Bitfolge abcdef schreibt man

$$M(x) = a \cdot x^5 + b \cdot x^4 + c \cdot x^3 + d \cdot x^2 + e \cdot x^1 + f \cdot x^0 \quad (3.1)$$

Dabei genügt es, die Terme mit Einsen anzugeben. Für die Bitfolge 100101 erhält man dann zum Beispiel

$$M(x) = x^5 + x^2 + x^0 \quad (3.2)$$

Die Prüfbits werden durch eine Division von solchen Polynomen bestimmt. CRC benutzt eine polynomiale Modulo-2-Arithmetik. In dieser Arithmetik reduziert sich die Subtraktion auf eine XOR-Operation. Die Division lässt sich – wie bei der üblichen Division – durch fortgesetzte Subtraktion ausführen.

Es sei als Beispiel x^3+x+1 das Divisorpolynom und damit 1011 das zugehörige Bitmuster. Die Nachricht 110101101 ist zu übertragen. Zunächst wird sie entsprechend der höchsten Potenz im Divisor um drei Bits erweitert: 110101101000. Genau in diese zusätzlichen Bits kommen später die Prüfbits.

Der erste Schritt in der Division ist dann:

$$\begin{array}{r}
 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0 \\
 1\ 0\ 1\ 1 \\
 \hline
 1\ 1\ 0
 \end{array}$$

Das Ergebnis der Division wird nicht benötigt und daher nicht notiert. Dann wird die nächste Stelle von oben übernommen und der Divisor erneut subtrahiert:

$$\begin{array}{r}
 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0 \\
 1\ 0\ 1\ 1\ \downarrow \\
 \hline
 1\ 1\ 0\ 0 \\
 1\ 0\ 1\ 1 \\
 \hline
 1\ 1\ 1
 \end{array}$$

Insgesamt ergibt sich

$$\begin{array}{r}
 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0 \\
 1\ 0\ 1\ 1\ \downarrow \\
 \hline
 1\ 1\ 0\ 0 \\
 1\ 0\ 1\ 1\ \downarrow \\
 \hline
 1\ 1\ 1\ 1 \\
 1\ 0\ 1\ 1\ \downarrow \\
 \hline
 1\ 0\ 0\ 1 \\
 1\ 0\ 1\ 1\ \downarrow\ \downarrow \\
 \hline
 0\ 1\ 0\ 0\ 1 \\
 1\ 0\ 1\ 1\ \downarrow\ \downarrow \\
 \hline
 1\ 0\ 0\ 0 \\
 1\ 0\ 1\ 1\ \downarrow \\
 \hline
 1\ 1\ 0
 \end{array}$$

Bei der Division bleibt ein Rest von 110. Zieht man diesen von der erweiterten Zahl ab, so ist die neue Zahl ohne Rest durch den Divisor teilbar. Da Subtraktion einer XOR-Operation entspricht, erhält man 110101101110. Diese um die drei Prüfbits erweiterte Bitfolge wird gesendet. Der Empfänger führt ebenfalls die Division durch. Falls dabei ein Rest übrig bleibt, liegt ein Übertragungsfehler vor.

Wie bei dem Beispiel mit den Dezimalzahlen gibt es auch für die Polynomdivision mehr oder weniger gut geeignete Divisorpolynome. In der Fachliteratur – beziehungsweise in den entsprechenden Spezifikationen – findet man jedoch schnell geeignete Polynome.

Man bezeichnet die Polynome in diesem Zusammenhang oft auch als Generatorpolynome. Einige davon sind unten in Tabelle 3.3 aufgelistet. Die Zahl nach dem Namen gibt an, wie viele Prüfbits jeweils genutzt werden. Details der Implementierung zusammen mit einer Realisierung in der Programmiersprache C findet man beispielsweise in [6].

Tabelle 3.3: Beispiele für gebräuchliche CRC-Polynome	
CRC	Divisor
CRC-8	$x^8 + x^2 + x^1 + 1$
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + 1$
CRC-16	$x^{16} + x^{15} + x^2 + 1$

Die Prüfverfahren schützen in allen Fällen, in denen die Anzahl der Bitfehler kleiner ist als die Anzahl der Prüfbits. Darüber hinaus werden auch die „meisten“ anderen Fehler erkannt.

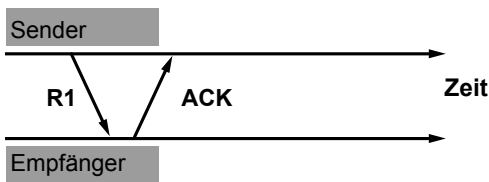
3.4 Sichere Übertragung von Rahmen

Wenn der Sender einen Rahmen abgeschickt hat, dauert es eine Weile, bis ihn der Empfänger erhalten und anschließend die Korrektheit festgestellt beziehungsweise wieder hergestellt hat.

Der Empfänger kann dann den Empfang quittieren, indem er dem Sender eine Bestätigung schickt. Insgesamt ergibt sich folgender Ablauf:

1. Sender schickt Rahmen
2. Empfänger wartet, bis der Rahmen vollständig angekommen ist
3. Empfänger prüft Korrektheit
4. Empfänger schickt Bestätigung
5. Sender erhält Bestätigung

Dieser Zusammenhang ist in Bild 3.4 mit einem Zeitstrahl grafisch dargestellt. R1 bezeichnet den ersten Rahmen und ACK die Bestätigung (Acknowledgement). In dieser Darstellung ist beim Empfänger eine kleine Zeitspanne nach Erhalt des Rahmens zur Prüfung eingetragen. Diese Zeit wird zwar im Prinzip immer benötigt, im Folgenden wird sie aber zur Vereinfachung der Darstellung nicht mehr explizit eingetragen.

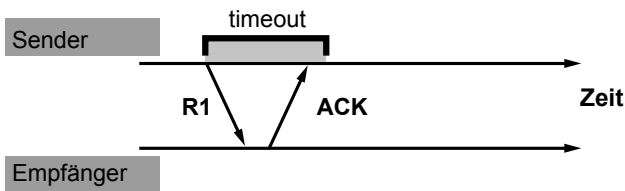


© tecCHANNEL

Bild 3.4: Zeitlicher Ablauf bei Bestätigung eines Rahmens R1.

3.4.1 Stop-and-Wait-Algorithmus

Eine direkte Implementierung des beschriebenen Verfahrens ist der Stop-and-Wait-Algorithmus. Der Sender schickt einen Rahmen und wartet auf die Bestätigung. Erhält er innerhalb einer gewissen Wartezeit keine Bestätigung, so geht er davon aus, dass der Rahmen verloren gegangen ist und schickt ihn erneut. Die Wartezeit bezeichnet man als Timeout. Damit ergibt sich Bild 3.5.



© tecCHANNEL

Bild 3.5: Zeitlicher Ablauf beim Stop-and-Wait-Algorithmus.

Der Timeout soll dem Empfänger genügend Zeit zur Bestätigung lassen. Andererseits soll er aber so kurz wie möglich sein, um die Wartezeit bei einem verloren gegangenen Rahmen zu minimieren. Ein Problem tritt auf, wenn die Bestätigung zu spät oder gar nicht ankommt. Beim Sender ist mittlerweile der Timeout abgelaufen, und in der Annahme, dass der Rahmen nicht korrekt empfangen wurde, schickt der Sender ihn erneut. Der Empfänger hat andererseits den Rahmen bestätigt und erwartet nun bereits den nächsten. Ohne weitere Maßnahmen würde er den zum zweiten Mal geschickten Rahmen bereits als nächsten interpretieren.

Es ist daher eine Kennung für die einzelnen Rahmen nötig. Beim Stop-and-Wait-Algorithmus verwendet man in der Regel die einfachste Nummerierung mit 0 und 1. Falls ein Rahmen fälschlicherweise erneut geschickt wird, erkennt dies der Sender an seiner Nummer. Er kann daher den Rahmen ignorieren, muss aber natürlich eine Bestätigung schicken. Den vollständigen Ablauf zeigt Bild 3.6. In diesem Beispiel geht die Bestätigung für den Rahmen 0 verloren. Der Sender schickt ihn daher nochmals. Diesmal wird die Bestätigung ordnungsgemäß zugestellt, und der Sender geht zum nächsten Rahmen, der die Kennung 1 erhält.

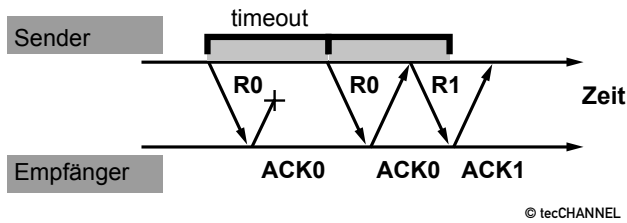


Bild 3.6: Stop-and-Wait-Algorithmus mit Rahmennummer, Verlust einer Bestätigung.

3.4.2 Sliding-Window-Algorithmus

Im Stop-and-Wait-Algorithmus gibt es lange Pausen, in denen der Sender auf eine Bestätigung wartet. Daher wird der Kanal nur schlecht ausgenutzt. Eine bessere Auslastung lässt sich aber erreichen, indem der Empfänger bereits während der Wartezeit weitere Rahmen schickt. Zu jedem Zeitpunkt sind dann mehrere Rahmen auf der Leitung unterwegs, und der Sender wartet auf entsprechend viele ausstehende Bestätigungen.

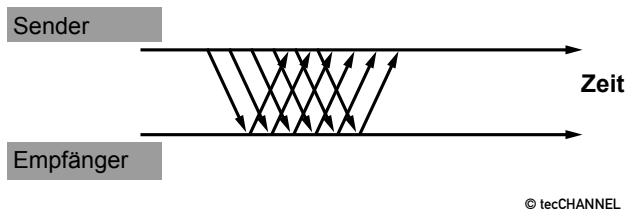


Bild 3.7: Sliding-Window-Algorithmus.

Der Sender nummeriert die Rahmen, so dass sie individuell bestätigt werden können. Die maximale Anzahl von gleichzeitig auf dem Weg befindlichen Frames – das Sendefenster (Send Window Size, SWS) – kann bei bekannter Leitungseigenschaft an die RTT der Leitung angepasst werden. Im Idealfall wird das Fenster dann so gewählt, dass der Timeout für den ersten Rahmen unmittelbar nach Abschicken des letzten Frames im Sendefenster abläuft. Der Sender muss alle Frames im Sendefenster vorrätig halten, um sie bei Bedarf nochmals senden zu können. Kommt die Bestätigung für den ersten Rahmen an, dann kann der Empfänger diesen löschen und das Sendefenster weiterschieben. Anderenfalls sendet er diesen Rahmen erneut.

Im Allgemeinen kommt es vor, dass zwar ein Rahmen verloren geht, aber die nachfolgenden Frames trotzdem ankommen. Eine gute Kanalauslastung erreicht man, wenn jeder angekommene Rahmen bestätigt wird (selektive Bestätigung, Englisch: selective acknowledgements). Allerdings ist dann der Aufwand in der Verwaltung des Sendefensters und des Empfangsfensters kompliziert. Geht bei-

spielsweise ein bestimmter Rahmen mehrfach verloren, muss der Empfänger sehr viele neue Frames speichern, um diesen alten Rahmen an der richtigen Stelle einfügen zu können. Wenn erforderlich, kann der Empfänger den Sender bremsen, indem er Frames, die zu weit vor dem letzten fehlenden Rahmen liegen, nicht bestätigt oder zumindest die Bestätigung verzögert.

Die Arbeit des Senders wird wesentlich vereinfacht, wenn der Empfänger die aus der Reihe angekommenen Rahmen nicht bestätigt. Angenommen Rahmen 4 geht verloren oder wird stark verzögert. Der Empfänger erhält stattdessen bereits die folgenden Rahmen 5 und 6. Er bestätigt diese Rahmen aber nicht, sondern wartet auf den Rahmen 4 – entweder das verspätete Original oder die wieder gesendete zweite Version. Sobald Rahmen 4 ankommt, bestätigt der Empfänger den Frame 6. Bei entsprechendem Protokoll erkennt der Sender daraus, dass alle Rahmen bis zur Nummer 6 gut angekommen sind (kumulative Bestätigung). Er kann dann das Sendefenster auf die Position 7 weiterschieben.

Wenn tatsächlich Frames verloren gehen, wird bei diesem Verfahren der Kanal nicht optimal ausgenutzt. Der Sender schickt während des Wartens nach dem wiederholten Rahmen unnötigerweise auch die nachfolgenden Frames zum zweiten Mal. Andererseits ist in dieser Form die Verwaltung der Rahmen in Sende- und Empfangsfenster recht einfach. Die Nummer eines Rahmens (Sequenznummer) ist eine zusätzlich zu übertragende Information. Aus Effizienzgründen sollten möglichst wenige Bits für diese Nummer verbraucht werden. Die Sequenznummer ist daher auf einen relativ kleinen Bereich beschränkt. Sobald das Maximum erreicht ist, springt der Zähler wieder auf 0 zurück. Es muss allerdings gewährleistet sein, dass die Rahmen noch eindeutig identifiziert werden können.

Prof. Dr. Stephan Euler

tecCHANNEL-Links zum Thema	Webcode	Compact
Netzwerk-Grundlagen	a1492	S.11
Rechner-zu-Rechner-Verbindung	a1493	S.25
So funktionieren TCP/IP und IPv6	p209	–

Mit den Webcodes gelangen Sie auf www.tecChannel.de direkt zum gewünschten Artikel. Geben Sie dazu den Code in das Feld „Webcode“ in der Titelleiste von www.tecChannel.de ein. Mit „p“ gekennzeichnete Webcodes beziehen sich auf Online-Artikel aus tecCHANNEL-Compact. Es ist möglich, dass Artikel zunächst exklusiv in tecCHANNEL-Compact veröffentlicht werden und daher erst später über den Webcode erreichbar sind.

4. Ethernet & Co.

Dieses Kapitel beschreibt die Funktionsweise von lokalen Netzen. Unter den LAN-Technologien ist Ethernet derzeit am weitesten verbreitet, während die Bedeutung von Funknetzen wächst. Daneben gibt es für Spezialanwendungen eine große Anzahl weiterer Technologien.

4.1 Ethernet

Unter der Bezeichnung Ethernet fasst man eine Familie von Netztechnologien zusammen. Diesen gemeinsam sind die Adressierung, das Rahmenformat und die Zugriffskontrolle auf das Übertragungsmedium. Ethernet wurde in den siebziger Jahren im PARC (Palo Alto Research Center), dem Forschungslabor der Firma Xerox, entwickelt. Später wurde daraus in Zusammenarbeit mit den Firmen DEC und Intel ein offener Standard. Dieser wiederum bildete die Grundlage für den offiziellen IEEE-802.3-Standard.

IEEE (sprich Eye-triple-E) ist das Akronym für Institute of Electrical and Electronics Engineers. Von dieser Organisation (www.ieee.org) werden neben vielen anderen Aktivitäten Standards entwickelt.

Die dreistellige Zahl im Namen eines IEEE-Standards bezeichnet das allgemeine Thema. Die Nummer 802 umfasst diverse Standards für Local and Metropolitan Area Networks (LAN/MAN). Durch die Zahl hinter dem Punkt (Part) wird der Standard genauer bezeichnet. Die Gruppe 802.3 beispielsweise trägt den Titel CSMA/CD Access Method und 802.11 spezifiziert Wireless LANs. Einzelne Standards innerhalb einer Gruppe werden durch angehängte Buchstaben oder Ziffern sowie eine Jahreszahl unterschieden (zum Beispiel 802.11g-2003, 802.15.3-2003). Der Standard IEEE-802.3 umfasst also eine ganze Familie von Techniken. Die jeweiligen Namen werden nach dem Schema

<Datenrate><Übertragungsverfahren><Segmentlänge oder Kabeltyp>

gebildet. So bezeichnet 10Base5 eine Variante mit 10 Mbit/s Basisband-Übertragung und einer maximalen Segmentlänge von 500 Metern.

4.1.1 Adressen

Die Adressen für Ethernet bestehen aus sechs Bytes. Üblich ist die Angabe der einzelnen Bytes mit je zwei Hex-Zeichen, getrennt durch Doppelpunkte. Führende Nullen werden oft weggelassen. Das Programm `ipconfig` gibt die Adresse mit allen Nullen und Bindestrichen aus.

Beispiele:

8:34:3e:0:55:a2

00-D1-59-6B-88-63

Jeder Rechner – oder genauer jede Netzwerkkarte – in der Welt, hat eine eindeutige Ethernet-Adresse. Diese enthält einen Anfangsteil, der den Hersteller kennzeichnet. Das erste Bit hat eine besondere Bedeutung. Ist es gesetzt, handelt es sich nicht um eine individuelle Adresse, sondern um die einer ganzen Gruppe von Rechnern (Multicast). Der Extremfall ist eine Adresse, die nur Einsen enthält. Mit dieser Adresse (Broadcast) werden alle Rechner gleichzeitig angesprochen.

4.1.2 Rahmenformat

Die Rahmen im Standard IEEE-802.3 haben folgendes Format:

8 Byte	6	6	2	46-1500	4
Präambel	Zieladresse	Quelladresse	Länge	Daten	CRC

Die Präambel enthält im Wesentlichen eine Folge von abwechselnden Nullen und Einsen. Sie erlaubt den Empfängern die Synchronisation auf den Rahmen. Anschließend folgen die Adressen von Ziel(en) und Sender. Bei IEEE-802.3 gibt das nächste Feld die Größe der darauf folgenden Daten an.

Dabei gelten eine Mindestgröße von 46 Byte und eine maximale Größe von 1500 Byte. Im ursprünglichen Ethernet-Standard enthält dieses Feld eine Kennung für den Rahmentyp. Wenn man sich für die Typkennzeichnung auf Werte größer als 1500 beschränkt, können beide Rahmentypen ohne Verwechslungsgefahr parallel verwendet werden. Nach den Daten folgen noch die Prüfbits einer 32-Bit-CRC.

Diese Darstellung ist etwas vereinfacht, denn die Realität ist ein wenig komplizierter, und es gibt mehrere Ausprägungen von 802.3-Rahmen. Auf die Details gehen wir nicht näher ein. Es sei lediglich erwähnt, dass die ersten drei Bytes der Nutzlast als Information für die so genannte Logical Link Control (LLC) gemäß IEEE-802.2 dienen.

4.1.3 Medienzugriff

Ethernet wurde für die gemeinsame Nutzung eines Mediums durch viele Stationen (Rechner) entwickelt (Mehrfachzugriffsnetz). Der erste Vorläufer war ein Funknetz zwischen den Hawaii-Inseln mit dem Namen Aloha. In diesem Fall war die Atmosphäre das gemeinsam genutzte Medium. Bei Ethernet war es ursprünglich ein Kabel, das die Rechner in einer Bustopologie verband.

Der gemeinsame Zugriff (Medienzugriffssteuerung, engl. Media Access Control MAC) wird über die Methode CSMA/CD (Carrier Sense Multiple Access with Collision Detection) geregelt. Das Grundprinzip ist einfach: Jede Station kann feststellen, ob die Leitung momentan frei ist (Carrier Sense). Wenn eine Station einen Rahmen senden möchte und die Leitung frei ist, beginnt sie sofort mit der Übertragung. Die Übertragungsdauer ist durch die maximale Datenmenge von

1500 Bits begrenzt. Wie bei einer Bustopologie üblich, hören alle Stationen permanent die Leitung ab. Wenn sie einen Rahmen entdecken, der für sie bestimmt ist, übernehmen sie ihn.

Jede Station entscheidet selbstständig, wann sie senden will. Dadurch kann es passieren, dass zwei Stationen in der Annahme, dass die Leitung frei ist, gleichzeitig anfangen zu senden. Die beiden Rahmen kollidieren auf der Leitung und werden damit unbrauchbar. Die Stationen erkennen die Kollision (Collision Detection), brechen die Übertragung ab und schicken ein 32 Bit langes Stausignal.

Nach einiger Zeit versuchen die Stationen erneut, ihre Rahmen zu schicken. Um zu verhindern, dass wieder beide Stationen gleichzeitig anfangen zu senden, wird die Wartezeit jetzt zufällig variiert. Im ersten Wiederholungsversuch beträgt sie 51,2 μ s, und jede Station entscheidet zufällig, ob sie 0 μ s oder 51,2 μ s wartet. Falls es erneut zu einer Kollision kommt, verdoppelt sich die maximale Wartezeit, und jede Station wartet zufällig für 0, 51,2, 102,4 oder 153,6 μ s.

Dieses Verfahren der Verdopplung der Wartezeit (exponential backoff) wird bis zu einer gegebenen Maximalzahl von Versuchen angewandt. Im Versuch i wählt jede Station zufällig eine Wartezeit aus der Menge 0, 51,2, 2, ..., $(2^i - 1) \times 51,2$ aus. In der Praxis wird allerdings i nicht größer als 10, und nach 16 erfolglosen Versuchen bricht der Netzwerkadapter mit einer Fehlermeldung ab.

Mit dieser Strategie werden einerseits Sendekonflikte zwischen zwei oder einigen wenigen Stationen schnell geklärt. Andererseits löst sich auch in dem Fall, dass viele Stationen gleichzeitig senden wollen, nach einigen Verdoppelungen der Gesamtwartezeit der Stau rasch auf.

Aus der CSMA/CD-Strategie ergeben sich Grenzwerte für die erlaubte maximale Laufzeit und die Rahmengröße. Betrachten wir dazu als Beispiel den extremen Fall zweier Stationen A und B an den entgegengesetzten Enden der Leitung. Die Station A sendet einen Rahmen. Der Rahmen erreicht nach der Latenz T die Station B. Für B war bis zu diesem Moment die Leitung frei. Im ungünstigsten Fall hat Station B daher unmittelbar, bevor dieser Rahmen bei ihr eintrifft, selbst mit einer Übermittlung begonnen. Die Station B entdeckt sofort den Konflikt und bricht daraufhin die Übertragung ab.

Allerdings dauert es wiederum die Zeit T , bis der von B begonnene Rahmen bei A eintrifft. Damit A seinen Rahmen noch zurücknehmen kann, darf A zu diesem Zeitpunkt noch nicht mit der Übertragung fertig sein. In Summe muss daher die Übertragung eines Rahmens stets mindestens die längste RTT $2T$ innerhalb des Netzes dauern. Die minimale Übertragungsverzögerung muss die Beziehung

$$t_U = K/B \geq 2T \quad (4.1)$$

erfüllen. Bei der Übertragungsrate 10 Mbit/s und einer minimalen Rahmengröße von 64 Bytes beziehungsweise 512 Bits ergibt sich als maximale RTT 51,2 μ s. Erhöht man die Übertragungsrate beispielsweise auf 100 Mbit/s, so reduziert sich die maximale RTT bei gleichen Rahmen auf 5,12 μ s.

4.1.4 Physikalische Eigenschaften

In der ursprünglichen Form wird Ethernet über ein Koaxialkabel (10Base5) in Bustopologie betrieben. Das Kabel hat eine maximale Länge von 500 Metern. Direkt an den Kabeln werden Transceiver angebracht (Mindestabstand 2,5 Meter), von denen aus Kabeln zu den Rechnern führen. Als preisgünstige Alternative wird dünneres Koaxialkabel (10Base2) verwendet, bei dem eine Abzweigung über ein T-Stück realisiert wird. Zwei Kabelsegmente lassen sich über einen Repeater verbinden.

Die moderne Form 10Base-T basiert auf verdrehten Kabelpaaren (twisted pair). Damit sind nur noch Punkt-zu-Punkt-Verbindungen möglich. Mehrere Stationen werden dann auf einen mehrwegigen Repeater (Sternkoppler oder Hub (Englisch: Nabe, Mittelpunkt)) geführt. Die einzelnen Kabeltypen sind in Tabelle 4.1 zusammengestellt. Repeater und Hubs arbeiten auf Bitebene und leiten die Signale ohne Analyse der Adressen weiter. Dann bleibt – unabhängig von der technischen Realisierung – das Netz eine logische Einheit. Jede Nachricht wird über das gesamte Netz verteilt, so dass Kollisionen an beliebiger Stelle auftreten können. Das Netz bildet eine Kollisionsdomäne (collision domain).

Tabelle 4.1: Kabeltypen für 10 Mbit/s Ethernet

Bezeichnung	Typ	Max. Länge	Bemerkung
10Base5	Dickes Koaxialkabel (ThickWire)	500 m	Kabel wird für Anschlüsse (Transceiver) angebohrt
10Base2	Dünnes Koaxialkabel (ThinWire)	200 m	Anschluss über T-Stück
10Base-T	Verdrelltes Paar (Twisted pair)	100 m	Punkt-zu-Punkt-Verbindungen
10Base-F	Glasfaser	2000 m	

Einen höheren Datendurchsatz erreicht man durch Trennung eines Netzes in mehrere Teilnetze, die durch vermittelnde Geräte wie Switches oder Router verbunden sind. Diese Geräte leiten Pakete nur an die Teilnetze weiter, in denen sich die Zielknoten befinden. Dadurch werden die anderen Teilnetze von unnötigem Verkehr freigehalten. Der Durchsatz lässt sich dann durch geeignete Topologien weiter optimieren. So ist es sinnvoll, Stationen die viel miteinander kommunizieren, an ein gemeinsames Teilnetz anzuschließen. Weiterhin können Stationen mit sehr hohem Datenaufkommen einen eigenen Port am Switch erhalten, so dass ihnen die Bandbreite exklusiv zur Verfügung steht. Eine Leistungssteigerung erreicht man auch durch Full-Duplex-Betrieb mit getrennten Kanälen für die beiden Richtungen.

Aufbauend auf dem Standard für 10 Mbit/s wurden mit 100 Mbit/s Fast Ethernet und 1000 Mbit/s Gigabit Ethernet zwei Nachfolgetechnologien mit höheren Band-

breiten entwickelt. Die beiden Verfahren benötigen Punkt-zu-Punkt-Verbindungen, wobei sowohl Twisted Pair als auch Lichtleiterkabel eingesetzt werden können. Die weit gehende Kompatibilität erleichtert die Migration bestehender Ethernet-Netze zu den höheren Datenraten.

4.1.5 Bewertung

Ethernet ist eine überaus erfolgreiche Technologie. Die Bandbreite wird zwar nicht optimal ausgenutzt, aber solange die Belastung nicht zu hoch ist (kleiner 30 Prozent), ist der Verlust durch die Kollisionen nicht störend. Vorteilhaft ist die Einfachheit von Installation und Betrieb. Es ist kein Problem, Stationen einzufügen oder aus dem Netz zu nehmen. Auch der Ausfall einzelner Stationen hat in der Regel keine negativen Auswirkungen auf den Netzbetrieb. Durch das robuste Protokoll gibt es auch kaum Situationen, in denen eine einzelne Station durch Fehlfunktionen das ganze Netz in Mitleidenschaft zieht.

Für normale Computer-Anwendungen wie etwa Fileservice ist Ethernet gut geeignet. Schwierig sind zeitkritische Anwendungen, die enge Anforderungen an die Reaktionszeit stellen. Darunter fallen auch Applikationen wie etwa Telefonverbindungen, bei denen Schwankungen in der Laufzeit sich störend bemerkbar machen. Durch den zufälligen Charakter des Medienzugriffs gibt es keine Garantie, dass ein Rahmen innerhalb einer vorgegebenen Zeit sein Ziel erreicht. In der Praxis kann man diese Probleme weit gehend vermeiden, indem man das Netzwerk nicht zu stark belastet.

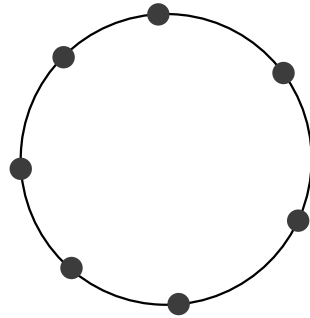
Ethernet ist die bei weitem am häufigsten eingesetzte LAN-Technologie. Die Tendenz geht zu immer höheren Datenraten, wobei aber stets noch die Kompatibilität mit den älteren Techniken gewahrt bleibt.

4.2 Token Ring

Im Zugriffsverfahren CSMA/CD können die Rechner zu jeder Zeit versuchen, eine Nachricht zu senden, sofern die Leitung frei ist. Die dadurch immer wieder auftretenden Kollisionen reduzieren jedoch den Durchsatz. Eine grundsätzliche Alternative ist ein geregelter Zugriff auf das Medium, bei dem dann zu jedem Zeitpunkt nur jeweils eine Station das Senderecht hat. Eine flexible Regelung basiert auf der Verwendung eines Tokens (Englisch: Zeichen, Erkennungsmarke). Nur die Station darf senden, die jeweils das Token besitzt. Die Details des Medienzugriffs werden dann durch den Umlauf des Tokens sowie durch die Regeln zu seiner Verwendung bestimmt.

Eine vergleichende Analyse der Leistungsfähigkeit von CSMA/CD und Token Ring findet sich in [7]. Die erfolgreichste Technik auf dieser Basis sind Token-Ring-Netze. Weit verbreitet ist hier der IBM-Token-Ring, der größtenteils identisch ist mit dem IEEE-Standard 802.5. Ein neuerer Standard ist dagegen FDDI (Fiber Distributed Data Interface).

Bild 4.1: Token-Ring ist ein Netzwerk mit Ring-Topologie.



© tecCHANNEL

Bei dem Token Ring sind alle Stationen – zumindest logisch – in Form eines Ringes angeordnet (Bild 4.1). Die Daten werden stets nur in eine Richtung weitergegeben. Jede Station hat genau eine Vorgängerin und eine Nachfolgerin. Die Stationen werden über spezielle Anschaltvorrichtungen an den Ring angeschlossen. Fällt eine der Stationen aus oder wird sie vom Netz genommen, dann schließt ein Relais den Ring wieder.

4.2.1 Medienzugriff

Das Token ist eine spezielle Nachricht, die zunächst frei im Ring kreist. Möchte eine Station senden, so nimmt sie das Token aus dem Ring und schickt stattdessen ihre Nachricht in einem oder mehreren Rahmen. Ähnlich wie bei Ethernet enthält jeder Rahmen die Zieladresse. Die Rahmen laufen dann im Netz von Station zu Station. Die Zielstation kopiert die Nachricht, nimmt die Rahmen aber nicht aus dem Netz, sondern lässt sie weiter kreisen. Erst die sendende Station selbst nimmt die Nachricht wieder heraus. Anschließend gibt sie das Token weiter, so dass die nachfolgende Station die Gelegenheit zum Senden hat.

Für eine sichere Übertragung ist ein Protokoll mit zwei Bits im Rahmen vorgesehen. Die beiden Bits mit der Bezeichnung A und C werden vom Sender auf 0 gesetzt. Erkennt eine Station einen an sie gerichteten Rahmen, setzt sie das A-Bit. Bei erfolgreicher Kopie setzt sie zusätzlich auch das C-Bit. Der Sender kann auf diese Art erkennen, ob

1. der Empfänger aktiv ist;
2. der Empfänger den Rahmen übernehmen konnte.

Wenn eine Station das Token übernommen hat, kann sie ihre Daten verschicken. Optimal im Sinne der Netznutzung wäre es, diese Station ihre gesamten Daten direkt hintereinander schicken zu lassen. Insbesondere bei ansonsten geringen Aktivitäten könnte man die Wartezeit einsparen, bis das Token jeweils umgelaufen ist. Dann hätten allerdings die anderen Stationen keine Chance, in der Zwischenzeit eigene Nachrichten zu versenden. Ein umfangreicher Transfer würde auch viele kleine Nachrichten blockieren. Daher wird die Zeit, für die eine Station

das Token behalten darf (THT, Token Hold Time), beschränkt. Vor jedem weiteren Rahmen prüft die Station, ob sie diesen Rahmen noch innerhalb der THT schicken kann. Andernfalls hält sie die Nachricht zurück und gibt das Token weiter. Im Standard 802.5 beträgt die THT 10 ms.

Damit kann man die obere Grenze für die Wartezeit angeben, bis eine Station wieder an die Reihe kommt. Im ungünstigsten Fall (worst case) übernimmt jede Station das Token und nutzt ihre THT voll aus. Für den Umlauf des Tokens (TRT, Token Rotation Time) gilt damit

$$TRT \leq \text{AnzahlStationen} \times THT + \text{RingLatenz} \quad (4.2)$$

Das Protokoll gibt allen Stationen eine gleiche Chance zum Senden. Die maximale Dauer für eine Übertragung (ohne Wiederholung von Rahmen auf Grund von Fehlern) lässt sich aus der obigen Abschätzung 4.2 für TRT bestimmen.

4.2.2 Netzüberwachung

Der Betrieb eines Token-Ring-Netzes erfordert eine Überwachung oder Wartung während des Betriebs. Einige mögliche Fehlerfälle sind:

- Das Token geht auf Grund von Bitfehlern verloren.
- Die Station im Besitz des Tokens fällt aus.
- Rahmen werden verfälscht und kreisen ewig im Ring.
- Ein Rahmen „verwaist“, weil die Sendestation ausfällt, bevor sie ihn wieder aus dem Ring nimmt.

Daher übernimmt eine der Stationen im Netz die Funktion eines Monitors. Jede Station kann Monitorstation werden. Es gibt ein wohl definiertes Protokoll, nach dem die Stationen bei der Inbetriebnahme des Rings oder nach Ausfall der Monitorstation aushandeln, welche diese Rolle übernimmt. Entdeckt die Monitorstation, dass nach der maximalen Umlaufzeit gemäß Gleichung 4.2 das Token immer noch nicht zurückgekommen ist, erzeugt sie einen neuen. Weiterhin kann sie anhand eines speziellen Monitorbits Rahmen erkennen, die mehrfach durch den Ring gereist sind. Diese Rahmen nimmt sie aus dem Ring.

4.3 Drahtlose LAN

Drahtlose Netze (Wireless LAN, WLAN) gewinnen immer mehr an Bedeutung. Nur durch drahtlose Netze ist freie Beweglichkeit möglich. Dadurch können auch mobile Teilnehmer – sowohl Rechner (Laptops) als auch Geräte wie Scanner, Barcode-Leser et cetera – in ein Netz eingebunden werden. Der besondere Vorteil liegt im schnellen Netzaufbau. Dieser Vorzug kommt in Fällen zum Tragen, in denen die Teilnehmer häufig wechseln oder einen schnellen, unkomplizierten Zugriff auf ein Netz brauchen. Typisch ist der Einsatz als Netzzugang in öffentlichen Bereichen (Hotspots) wie Flughäfen, Bahnhöfen oder Hotels.

Die wichtigsten Standards für drahtlose Netze sind IEEE 802.11 und die daraus abgeleiteten Varianten. Eine Übersicht über diese Familie gibt Tabelle 4.2. Derzeit aktuell sind 802.11b/g, auch bekannt als Wi-Fi (für Wireless Fidelity). Der Standard 802.11g ist kompatibel mit 802.11b.

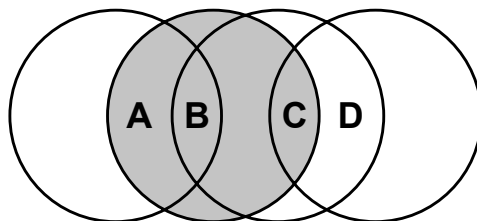
Als potenzieller Nachfolger erscheint der Anfang 2003 verabschiedete Standard IEEE 802.16 aussichtsreich. Bei Entfernungen bis 50 Kilometer und Übertragungsleistungen bis zu 134 Mbit/s soll er eine großflächige Bereitstellung von drahtlosen Netzen ermöglichen. In diesem Zusammenhang spricht man bereits von Wireless Metropolitan Area Networks (WMAN).

Tabelle 4.2: Standards für WLAN

Standard	Beschreibung	Technische Daten
802.11	erster Standard von 1997	2 bis 3 Mbit/s, 2,4 GHz-Band
802.11b	Nachfolger von 802.11	bis 11 Mbit/s, 2,4 GHz-Band
802.11g	Nachfolger von 802.11b	bis 54 Mbit/s, 2,4 GHz-Band
802.11a	Nachfolger von 802.11b	bis 54 Mbit/s, 5 GHz-Band

IEEE 802.11 wurde für zwei Methoden der Funkübertragung und eine Art der Infrarot-Übertragung ausgelegt. Die Kollisionsvermeidung erfolgt ähnlich wie bei Ethernet. Allerdings ergeben sich aus der endlichen Reichweite der Signale spezielle Probleme. Anders als bei Ethernet erreicht ein Rahmen nicht alle Stationen. Bild 4.2 zeigt zur Veranschaulichung ein Netz mit vier Stationen A, B, C und D. Die jeweilige Reichweite ist durch einen Kreis dargestellt.

Bild 4.2: Drahtloses Netz mit vier Stationen.



© tecCHANNEL

In einer solchen Konstellation kann ein Sender nicht mehr alle Kollisionen feststellen. Angenommen A und C senden gleichzeitig an B. Da A und C weit auseinander liegen, empfangen sie nicht das Signal der jeweils anderen Station, so dass die Kollision unbemerkt bleibt. Der Standard 802.11 sieht verschiedene Zugriffsverfahren vor, um solche Probleme zu umgehen.

Ein Konzept beruht auf dem Austausch von kurzen Protokollrahmen (MACA, Multiple Access with Collision Avoidance). Der Sender schickt zunächst eine Be-

nachrichtigung (RTS, Request To Send) an den Empfänger. Ist der Empfänger bereit, antwortet er mit einer Bestätigung (CTS, Clear To Send). Der CTS-Rahmen dient als Bestätigung, dass der Sender mit der Übertragung beginnen kann. Bleibt das CTS aus, schließt der Sender daraus – nach einem Timeout –, dass das RTS nicht angekommen ist, weil es zum Beispiel zu einer Kollision kam. Beide Protokollrahmen enthalten die Information über den Umfang der zu sendenden Daten. Daraus können benachbarte Stationen berechnen, wie lange der Transfer dauern wird und warten entsprechend lange mit eigenen Übertragungen.

Ähnlich wie bei Mobiltelefonsystemen mit Funkzellen organisiert man Netze nach 802.11 in Regionen. Die Rolle der Basisstationen übernehmen dann so genannte Anschlusspunkte (AP, Access Point). Die Anschlusspunkte können untereinander beispielsweise über Ethernet verbunden sein. Jeder mobile Knoten meldet sich bei einem Anschlusspunkt an und kommuniziert bis auf Weiteres nur über ihn. Der Standard definiert die Protokolle, nach denen sich Rechner anmelden. Ein Knoten kann einerseits selbst nach APs suchen (aktives Scanning). Andererseits schicken auch die APs Rahmen, mit denen sie ihre Dienste anbieten (passives Scanning). Bei mobilen Knoten kann es allerdings durchaus vorkommen, dass einer den Bereich seines APs verlässt. In einem solchen Fall findet er wiederum mit einer der Scanning-Methoden einen besser geeigneten AP. Idealerweise erfolgt dieser Wechsel zwischen zwei APs ohne Unterbrechung der Verbindung.

Neben der Kommunikation über mindestens einen gemeinsamen AP (infrastructure mode) können auch mehrere Stationen direkt miteinander Verbindung aufnehmen (Ad-hoc- oder Peer-to-Peer-Mode). Eine Sonderform sind so genannte Manets (mobile ad hoc networks), bei denen mehrere mobile Knoten untereinander ein gemeinsames Ad-hoc-Netz betreiben.

Der Aufbau und Betrieb von drahtlosen Netzen erfordert deutlich höheren Aufwand. Die beschränkte Reichweite der Knoten und ihre Beweglichkeit stellen neue Herausforderungen an die Kontrolle des Medienzugriffs. Auch die Anforderungen an die Sicherheit der Daten gegen Verfälschungen oder Abhören wachsen. Im Gegensatz zu einem leitungsgebundenen Netz ist es einfach, Pakete abzuhören oder eigene einzuschleusen. Die Inhalte sollten daher durch Verschlüsselung geschützt werden. Bei Netzen mit geschlossenem Benutzerkreis ist es sinnvoll, nur Geräte mit einer registrierten Netzwerkadresse zuzulassen. Der Standard 802.11i bietet Erweiterungen bezüglich Sicherheit und Authentifizierung.

4.4 Andere Netztechnologien

Neben diesen Technologien gibt es eine Reihe von Alternativen. Für Spezialfälle existieren angepasste Lösungen. So bedingen manche Anwendungen hohe Anforderungen an Robustheit und Zeitverhalten. Häufig sind mehrere Alternativen verfügbar. Nicht in allen Fällen entwickelte sich ein einheitlicher Standard.

LAN

- **Token Bus IEEE 802.4:** In diesem Protokoll wird ein Token-basiertes Verfahren auf einer Busarchitektur realisiert. Der Vorteil liegt in dem deterministischen Zeitverhalten. Diese Technik hat nur in speziellen Anwendungsbereichen Verbreitung gefunden. So basiert das von der Firma General Motors für die industrielle Fertigung entwickelte Manufacturing Automation Protocol (MAP) auf diesem Ansatz.
- **FDDI (Fiber Distributed Data Interface):** Eine ursprünglich für die Übertragung über Glasfaser entwickelte Technik, basierend auf einem doppelten Token Ring. Mit einer Übertragungsrate von 100 Mbit/s bei Ringlängen bis maximal 100 Kilometer wurde es häufig als Backbone (Englisch: Rückgrat) zur Vernetzung größerer Bereiche (Firmen, Uni-Campus) eingesetzt. Einzelne Rechner werden in der Regel nicht direkt, sondern über Konzentratoren an den Ring angeschlossen. Auch Ethernet-Netze können mittels FDDI/Ethernet-Bridges mit dem Ring verbunden werden. Durch die Entwicklung von Fast Ethernet und Gigabit Ethernet hat FDDI kaum noch Bedeutung.

Drahtlose Netze

- **Bluetooth:** ein Standard für die drahtlose Vernetzung von (kleinen) Geräten mit geringer Reichweite. Typisch ist die Vernetzung von mobilen Kleingeräten wie Mobiltelefonen und PDAs. Ein solches Netzwerk wird auch als Wireless Personal Area Network (WPAN) bezeichnet.

Netze für Peripheriegeräte

- **SCSI (Small Computer System Interface):** ein paralleler Bus für Peripheriegeräte wie Festplatten oder Scanner. SCSI wird vornehmlich im Highend-Bereich eingesetzt.
- **USB (Universal Serial Bus):** ein schneller serieller Bus zum Anschluss von Peripheriegeräten (Drucker, Scanner, Maus et cetera) an einen Rechner. In den Versionen 1.0 und 1.1 beträgt die Übertragungsrate bis zu 12 Mbit/s, bei der Version 2.0 bis zu 480 Mbit/s. Durch Einsatz eines USB-Hubs lassen sich an einen USB-Port bis zu 127 Geräte anschließen.
- **Firewire IEEE 1394:** eine schnelle serielle Schnittstelle mit Übertragungsrate bis zu 400 Mbit/s bei maximal 63 anschließbaren Geräten. Firewire wird beispielsweise als Verbindung zwischen Rechner und externer Festplatte oder Video-Kamera verwendet.

Feldbusse

Feldbusse sind für den Einsatz in der Automatisierungstechnik ausgelegt. Sie verbinden Steuergeräte mit Sensoren und Aktoren. Wichtig ist die Robustheit gegenüber Störungen. Die meisten Feldbusse garantieren eine Zeitspanne, innerhalb der Daten übermittelt werden (Echtzeitverhalten).

- **Profibus:** Ein offenes, flexibles System für Anwendungen in den Bereichen Prozessleittechnik, Gebäudeautomation et cetera. Es unterstützt die Kombinationen von Geräten in unterschiedlichen Konfigurationen (zum Beispiel Mono-Master, Multi-Master, Master-Slave). Ein Token-basiertes Protokoll regelt den Buszugriff.
- **CAN (Controller Area Network):** CAN benutzt eine Bustopologie mit Vielfachzugriffsverfahren und Prioritätsregeln. CAN wird auch zur Vernetzung von Komponenten in Kraftfahrzeugen eingesetzt. Neuere Entwicklungen für zeitkritische Anwendungen im Kfz-Bereich sind TTP (Time Triggered Protocol) und – basierend auf dem ByteFlight-Konzept von BMW – FlexRay.

Gebäudevernetzung

Darunter versteht man die Vernetzung der verschiedensten Sensoren und Geräte innerhalb eines Gebäudes. Die Vision ist eine einheitliche Steuerung von Geräten zur Klimasteuerung, Beleuchtung, Sicherheit, Kommunikation, Unterhaltung und so weiter. Derzeit wird die Technik überwiegend in gewerblichen Gebäuden eingesetzt, soll aber zunehmend auch in Privathaushalten Einzug halten.

- **EIB (Europäischer Installationsbus):** EIB ist ein europaweit einheitlicher Standard zur Vernetzung von Komponenten der Gebäudesystemtechnik.
- **Local Operating Network (LON), auch LONWORKS genannt:** universelles Automatisierungsnetzwerk, entwickelt von der Echelon Corporation [8].

Prof. Dr. Stephan Euler

tecCHANNEL-Links zum Thema	Webcode	Compact
Ethernet und Co.	a1494	S.38
Ethernet im Überblick	a717	–
10-Gigabit-Ethernet	a876	–
Wireless LANs im Überblick	a750	–
802.11: Standard für drahtlose Netze	p680	–
Reality Check: IEEE-802.11-WLANs	a1129	–
Bluetooth – der Kabel-Killer	a477	–
DECT: Die Alternative zu Bluetooth	p511	–
Grundlagen: Netzwerk-Verkabelung	a1262	–

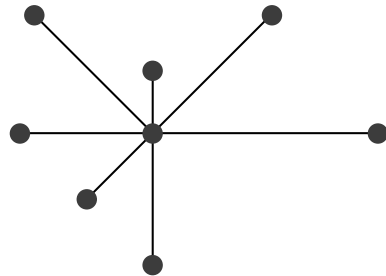
Mit den Webcodes gelangen Sie auf www.tecChannel.de direkt zum gewünschten Artikel. Geben Sie dazu den Code in das Feld „Webcode“ in der Titelleiste von www.tecChannel.de ein. Mit „p“ gekennzeichnete Webcodes beziehen sich auf Online-Artikel aus tecCHANNEL-Compact. Es ist möglich, dass Artikel zunächst exklusiv in tecCHANNEL-Compact veröffentlicht werden und daher erst später über den Webcode erreichbar sind.

5. Vermittlung

Im letzten Kapitel haben wir gesehen, wie Stationen direkt miteinander kommunizieren können. Die direkte Verbindung stößt jedoch an Kapazitätsgrenzen, wenn es gilt, viele Stationen zu verbinden oder eine hohe Datenrate zu gewährleisten.

Allgemein gesehen handelt es sich um eine Reihe von Teilnetzen, die zusammengeschaltet werden sollen. Im Extremfall kann jedes Teilnetz aus nur einer einzelnen Station bestehen. Man kann die Teilnetze beispielsweise in Form eines Sterns über ein zentrales Koppellement verbinden (Bild 5.1).

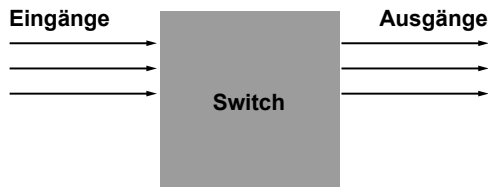
Bild 5.1: Netzwerk mit Sterntopologie und zentralem Koppellement.



© tecCHANNEL

Bei einem einfachen Ethernet-Hub werden alle Pakete über den gesamten Stern und damit an alle Knoten geschickt. Effizienter ist es, wenn das Koppellement die Adresse auswertet und die Pakete nur an die betroffenen Teilnetze schickt. Ein Gerät mit entsprechender Funktionalität ist ein Vermittler oder Switch. Der grundsätzliche Aufbau ist in Bild 5.2 dargestellt.

Bild 5.2: Prinzip eines Switchs. Das Gerät wertet die Adressen aus und schickt die Pakete nur an die betroffenen Teilnetze.



© tecCHANNEL

Ein Switch hat eine feste Anzahl von Ein- und Ausgängen. An diesen Anschlüssen – auch als Ports bezeichnet – werden Teilnetze oder Knoten angeschlossen. Dabei können verschiedene Typen von Ports vorhanden sein. Zum Beispiel kann ein Switch eine Reihe von Ports mit 10BaseT zum direkten Anschluss von Stationen und breitbandige Anschlüsse zur Vernetzung mit weiteren Switches haben. Empfängt der Switch an einem Eingangsport ein Paket, so liest er die Adresse und entscheidet dann, auf welchen Ausgangsport er es schickt.

Bevor wir näher auf Switches eingehen, sind in der folgenden Auflistung die unterschiedlichen Geräte zur Verbindung von Netzen zusammengestellt. Im Wesentlichen unterscheiden sich die Gerät nach der Anzahl der Anschlüsse sowie in der eingebauten „Intelligenz“.

Einfache Geräte arbeiten nur auf der Ebene des physikalischen Signals, während aufwendige Geräte eine komplette Analyse der Daten über alle Ebenen des ISO/OSI-Referenzmodells durchführen.

- **Repeater:** Ein Repeater verbindet zwei gleichartige Netze. Er arbeitet auf Ebene 1 des ISO/OSI-Referenzmodells und behandelt nur das physikalische Signal. Ein Repeater kann beispielsweise dann verwendet werden, wenn ein Ethernet-Strang die maximale Länge erreicht hat. Über einen Repeater kann dann ein weiterer Strang angeschlossen werden. Der Repeater regeneriert das Signal, bevor er es in den zweiten Strang überträgt.
- **Hub:** Ein Hub entspricht im Prinzip einem Repeater, jedoch mit mehreren Ports (Multiport-Repeater).
- **Bridge:** Eine Bridge stellt eine Verbindung zweier Netze her, wobei Ebene 1 und 2 des ISO/OSI-Referenzmodells berücksichtigt werden müssen. Bei der Weiterleitung werden die Adressinformationen ausgewertet und fehlerhafte Rahmen blockiert.
- **Switch:** Multiport-Bridge
- **Router:** Router verbinden Netzwerke unterschiedlicher Art. Falls erforderlich, können Sie Datenpakete in eine andere Form umsetzen. Sie arbeiten auf Schicht 3.
- **Gateway:** Als Schnittstelle zwischen unterschiedlichen Netzen (zum Beispiel TCP/IP mit einem herstellerspezifischen Protokoll wie DECnet) dienen Gateways. Sie arbeiten auf allen 7 Schichten.

Bei der Entwicklung von Switches stellen sich zwei Herausforderungen. Das erste Problem ist die Realisierung der Vermittlungsfunktionalität.

- Woher weiß der Switch, an welchen Ausgangsport ein Paket mit der Adresse xyz geschickt werden soll?

Der zweite Themenkomplex ist die effiziente Realisierung eines Switches. Intern ist zunächst ein schneller Datentransfer zu gewährleisten. Weiterhin muss der Switch auf jeder angeschlossenen Leitung ein sicheres Protokoll realisieren und beispielsweise auch Verzögerungen durch Kollisionen handhaben können. Die Kernfrage lautet dann:

- Wie kann ein leistungsfähiger und trotzdem preisgünstiger Switch in Hardware realisiert werden?

Im Folgenden beschreiben wir zunächst die Grundprinzipien der Vermittlung. Anschließend werden einige anschauliche Beispiele für die Realisierung von Switches vorgestellt.

5.1 Datagramme

In der Grundform der Paketvermittlung trägt jedes Datenpaket die Zieladresse. Anhand dieser Zieladresse wird das Paket durch das Netzwerk geleitet. Jeder Switch führt eine Tabelle mit der Information, auf welchen Port er Pakete mit gegebener Zieladresse weiterleitet (Weiterleitungstabelle oder Routing-Tabelle).

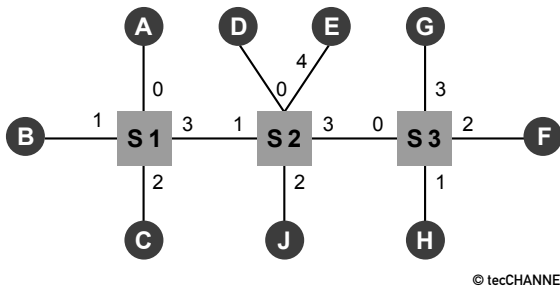


Bild 5.3: Netz mit neun Knoten A bis J und drei Switches.

Das Bild 5.3 zeigt ein Beispielnetz mit neun Knoten und Switches. Die Ports sind nummeriert. Der Einfachheit halber sind Ein- und Ausgänge zusammengelegt. Jeder Switch führt eine Tabelle, in der für jedes Ziel (Knoten) der richtige Port eingetragen ist. Entweder hängt der Knoten direkt an diesem Port oder der Port führt zum nächsten Switch in Richtung Ziel. Die Weiterleitungstabelle für Switch 2 ist in Tabelle 5.1 angegeben.

Tabelle 5.1: Weiterleitungstabelle für Switch 2

Zielknoten	Port
A	1
B	1
C	1
D	0
E	4
F	3
G	3
H	3
J	2

5.1.1 Lernende Bridges

Lernende Bridges nutzen eine einfache Strategie zum Aufbau der Weiterleitungstabellen. Bridges sind Geräte, die mehrere LANs verbinden, indem sie ankommenden Rahmen an die Ausgänge weitergeben. Wenn eine Bridge die Rahmen analysiert, kann sie feststellen, an welchem Port Rahmen von welchem Knoten ankommen. Dieser Port wird in die Weiterleitungstabelle für den Knoten eingetragen. Von da an werden Rahmen an diesen Knoten nur noch auf diesem Port ausgegeben. Durch diese Strategie wird für einen Knoten, sobald er Rahmen sendet, automatisch der richtige Eintrag in die Weiterleitungstabelle aufgenommen.

Am Anfang ist die Tabelle leer und wird erst im Laufe der Zeit gefüllt. Auch wenn die Tabelle das Netzwerk nicht vollständig erfasst, verbessert sich der Netzwerkverkehr. Fehlt für einen Zielknoten der Eintrag, so wird der Rahmen nach wie vor an alle Ports (außer dem Eingangsport) weitergeschickt. Um Änderungen im Netz zu erfassen, wird der Eintrag eines Knotens wieder gelöscht, sobald über eine gewisse Zeit keine Rahmen mehr von diesem Knoten gekommen sind.

Ein Problem entsteht bei der Verknüpfung von LANs über Bridges, wenn das Netz Schleifen enthält. Solche Schleifen entstehen entweder aus Versehen oder bewusst als Redundanz gegen Ausfall einer Bridge. Ohne Gegenmaßnahmen kann es dazu kommen, dass Rahmen ewig in den Schleifen kreisen. Eine Lösung dieses Problems stammt von R. Perlmann. Um Schleifen zu erkennen, tauschen die Bridges regelmäßig Informationen über den ihnen bekannten Teil des Netzes aus. Aus den Informationen berechnen sie dann einen Baum (Spanning Tree), der die jeweils kürzesten Verbindungen enthält. Eventuell vorhandene zusätzliche Verbindungen, die im vollständigen Netz zu Schleifen führen, werden im Baum nicht berücksichtigt. Während des Betriebs wird der Baum regelmäßig aktualisiert, um eventuelle Änderungen zu erfassen. Der Standard IEEE 802.1D „Media Access Control (MAC) Bridges“ beschreibt das Protokoll.

5.2 Virtuelle Verbindungen

Die Vor- und Nachteile der Nachrichtenübertragung mit Datagrammen wurden bereits diskutiert. Datagramme sind besonders geeignet, wenn nur kurze Nachrichten geschickt werden oder eine ausfallsichere Kommunikation wichtig ist. Andererseits wird häufig nicht nur eine einzelne Nachricht übermittelt, sondern die Kommunikation erfolgt über einen längeren Zeitraum (zum Beispiel Verbindung zwischen zwei Standorten einer Firma). In solchen Fällen ist die Datagramm-Übertragung nicht effizient, da für jedes Paket wieder der gesamte Vermittlungsprozess abläuft. Daher wickelt man die Kommunikation häufig über virtuelle Leitungen ab.

Virtuelle Leitungen (Virtual Circuit VC) simulieren mit Methoden der Paketvermittlung klassische Leitungsvermittlung. In einer Aufbauphase wird zunächst ein Weg vom Sender zum Empfänger festgelegt. Jeder dabei beteiligte Switch reser-

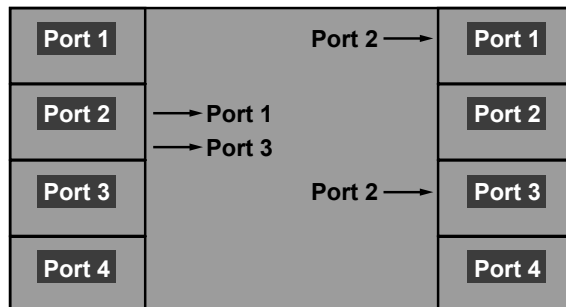
viert dann Kapazität für die Verbindung. Sobald die Verbindung steht, können Pakete über die Verbindung geschickt werden. Die Pakete benötigen nicht mehr die Zieladresse, sondern nur noch einen Bezeichner für die VC. Zum Ende der Verbindung sind die Switches zu informieren, so dass sie die VC wieder abbauen und die Ressourcen für neue Verbindungen frei machen.

Virtuelle Leitungen lassen sich entweder vom Netzsadministrator dauerhaft anlegen (Permanent Virtual Circuit PVC) oder von einem Knoten bei Bedarf aufbauen (Switched Virtual Circuit SVC). Man kann sich den Netzaufbau ähnlich wie die Weiterleitung eines Datagramms vorstellen. Der Sender schickt ein Paket mit dem Wunsch zum Verbindungsaufbau. Das Paket wird durch das Netz zum Empfänger geleitet. Beim Passieren des Pakets reserviert ein Switch die Ressourcen für die Verbindung. An dieser Stelle lassen sich Vorgaben für die Qualität der Verbindung treffen, beispielsweise bezüglich Bandbreite oder Verzögerungszeit. So ist für die Verbindung eine definierte Dienstgüte (Quality of Service, QoS) garantiert.

Ist die VC aufgebaut, braucht jeder Switch sich nur noch die Verbindung zwischen Ein- und Ausgang zu merken. Da über einen Port mehrere VC kommen können, reicht nicht die Portnummer, sondern eine Kennzeichnung für die VC ist zusätzlich erforderlich. Diese Kennzeichnung gilt nicht global für den VC, sondern nur innerhalb jedes Switches und dort wieder für einen Port.

In Bild 5.4 ist ein Beispiel für einen Switch mit vier Ein- und vier Ausgangsports dargestellt. Eingezeichnet sind zwei virtuelle Verbindungen; eine von Port 2 nach Port 1 und die zweite ebenfalls von Port 2 aus nach Port 3.

Bild 5.4: VC innerhalb eines Switches.



© tecCHANNEL

Um die Verbindungen zu kennzeichnen, vergeben die Switches eindeutige Identifikatoren. Das sind so genannte Virtual Circuit Identifier VCI. Für die Eingänge kann der Switch selbst die Nummerierung übernehmen. Sobald ein anderer Switch auf einem Port eine neue Verbindung anmeldet, vergibt der Switch eine freie VCI-Nummer. Diese meldet er dem Vorgänger-Switch, der sie dann als Ausgangs-VCI übernimmt. In Summe sind die Schritte für eine Verbindungsteilstrecke von Switch N zu Switch M:

1. Switch N sendet eine Anfrage an Switch M.
2. Switch M entscheidet, ob er die Anfrage akzeptiert.
3. Falls Switch M die Anfrage akzeptiert, wählt er einen freien VCI für seinen Eingangsport aus.
4. Switch M meldet seinen Eingangs-VCI an Switch N.
5. Switch N trägt den erhaltenen VCI dann als Ausgangs-VCI in seine eigene Tabelle ein.

Die Rückmeldung der VCI an den Vorgänger-Switch muss nicht unmittelbar erfolgen. Sie kann auf dem Weg einer Bestätigung vom Empfänger zurück zum Sender bei allen Switches nacheinander übergeben werden. Das Resultat trägt der Switch in eine Tabelle ein. Für das obige Beispiel könnte das Ergebnis folgendermaßen aussehen:

Tabelle 5.2: VC-Tabelle mit zwei Einträgen			
Eingangsport	Eingangs-VCI	Ausgangsport	Ausgangs-VCI
2	1	1	5
2	2	3	3

Techniken mit virtueller Leitungsvermittlung sind Frame-Relay und ATM. Frame-Relay ist bei der Übertragung in Weitbereichsnetzen im Einsatz. Eine wichtige Anwendung ist die Kopplung mehrerer LANs zu einem Virtual Private Network (VPN). Dazu werden vornehmlich PVC benutzt.

5.3 Design von Switches

Wie oben beschrieben, besteht ein Switch aus einer Reihe von Ein- und Ausgängen – den Ports – sowie einer Einheit zur Verteilung der Pakete – dem Schaltnetzwerk (Switching Fabric). Die Herausforderung beim Design ist, Pakete von möglichst vielen Ports möglichst schnell zu verteilen. Dazu ist intern eine entsprechend hohe Verarbeitungsgeschwindigkeit bereitzustellen.

Die Eingangsports nehmen Pakete in Empfang und speisen sie in das Schaltnetzwerk ein. Ausgangsports erhalten Pakete vom Schaltnetzwerk und schicken sie auf die Leitung. Diese zunächst trivial erscheinenden Aufgaben erweisen sich jedoch bei genauerer Analyse als komplex. Das Grundproblem liegt nämlich in der ungleichen Verteilung des Verkehrs. In aller Regel kommuniziert nicht stets ein Eingang mit genau einem Ausgang. Beispielsweise schickt ein Server Pakete an viele Knoten. Damit werden an dem Port viele Pakete ankommen, an dem der Server angeschlossen ist. Diese Pakete sind auf mehrere Ausgänge zu verteilen. Ein Extremfall ist ein Broadcast-Paket, das an einem Eingang ankommt und an alle Ausgänge weitergereicht werden muss.

Entsprechend gilt für die Ausgänge, dass auch dort Pakete von vielen Eingängen „gleichzeitig“ ankommen können und dann in eine Leitung einzuspeisen sind. Daher müssen Ports in der Lage sein, Pakete in einen Zwischenspeicher (Puffer) zu legen, bis sie weitergeschickt werden können. Bei dem Entwurf gilt es, einen guten Kompromiss zu finden: Die Puffer sollen möglichst selten überlaufen, aber andererseits auch nicht überdimensioniert sein.

Prinzipiell ist es möglich, die Puffer an den Eingängen oder an den Ausgängen zu platzieren. Zwischenspeicher an den Eingängen können allerdings schnell zu unnötigen Blockaden führen. Betrachten wir folgendes Beispiel von Paketfolgen an zwei Eingängen:

```
Port N:  Ziel 4  Ziel 4  Ziel 5   →
Port M:  Ziel 7  Ziel 6  Ziel 5   →
```

Die beiden nächsten Pakete an der Reihe kollidieren, da sie beide an Port 5 gerichtet sind. Bei Eingangspufferung würde eines der Pakete ausgewählt und die Ausgabe am zweiten Port verzögert. Damit werden auch weiter hinten liegende Pakete blockiert, obwohl ihr Ziel (4 oder 6) eventuell frei ist (Head-of-Line-Blocking). Daher erfolgt die Pufferung in der Regel an den Ausgängen.

5.3.1 Weiterleitung

Eine weitere Frage ist, wann ein Paket weitergeleitet werden soll? Eine Möglichkeit hierfür: erst das Paket vollständig abwarten, dann die Integrität prüfen und es – sofern alles damit in Ordnung ist – anschließend weitergeben (Store-and-Forward-Prinzip). So wird die weitere Verbreitung unvollständiger oder fehlerhafter Pakete verhindert. Daraus resultiert aber auch, dass die Verarbeitungszeit im Switch mindestens so groß ist wie die Übertragungsverzögerung für das Paket.

Die Alternative ist, ein Paket frühzeitig weiterzugeben. In der Regel steht die Zieladresse im vorderen Teil des Pakets. Sobald dieser Teil gelesen ist, kann schon die Weiterleitung aufgenommen werden (Cut-Through- oder On-the-fly-Prinzip). Allerdings nimmt man damit auch in Kauf, dass fehlerhafte Pakete in andere Netze übertragen werden.

Je nach Anwendungssituation kann eine der beiden Strategien besser sein. Bei einem lokalen Netz mit relativ geringer Auslastung lässt sich durch die beschleunigte Weiterleitung die Gesamtverzögerung reduzieren.

Ist umgekehrt das Netz stark belastet, so dass es häufiger zu Kollisionen kommt, dann wird die Lage durch die ungehinderte Verbreitung von unvollständigen Paketen noch verschärft.

Einige Hersteller bieten deshalb Geräte an, die dynamisch zwischen den beiden Betriebsweisen hin und her schalten können. Abhängig von etwa der Anzahl der festgestellten fehlerhaften Pakete wird dann zwischen der schnelleren oder der sichereren Option gewählt.

5.3.2 Knockout-Switch

Bei einem Knockout-Switch [9] werden alle ankommenden Pakete an alle Ausgangsports verteilt. An jedem Ausgangsport sind Paketfilter, die nur an den Port gerichtete Pakete passieren lassen. Die Pakete werden in eine Warteschlange eingereiht und von dort aus abgeschickt. Falls die Kapazität der Warteschlange nicht ausreicht, alle Pakete aufzunehmen, ist eine Auswahl zu treffen. Diese Aufgabe übernimmt ein Konzentrator, der aus L Paketen $N < L$ Pakete auswählt. Die günstige Realisierung beruht auf einfachen Auswahlbausteinen mit nur zwei Eingängen. Ein Baustein wählt zufällig aus den beiden ankommenden Paketen eines als „Sieger“ aus. Wie bei einem Turnier in K.-o.-Modus treten alle Pakete gegeneinander an, bis ein Sieger übrig bleibt. Die Verlierer kommen in eine zweite Runde, die nach dem gleichen Modus den zweiten Sieger ermittelt. Das Verfahren wird fortgesetzt, bis N Sieger feststehen. Die restlichen Pakete werden verworfen.

5.3.3 Batchter- und Banyan-Netzwerke

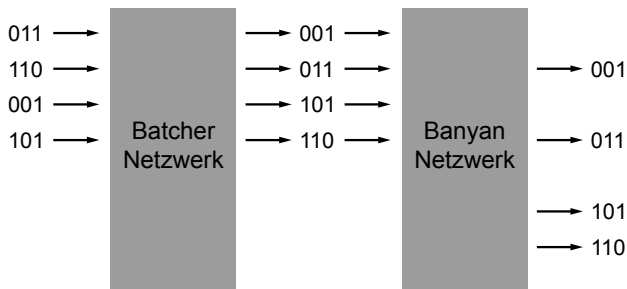
Schaltnetze lassen sich günstig als Verbund von gleichartigen kleinen, einfachen Elementen bauen. Solche Elemente haben zwei Eingänge sowie zwei Ausgänge und entscheiden nur zwischen zwei Möglichkeiten:

- Eingang 1 auf Ausgang 1 und Eingang 2 auf Ausgang 2
- Eingang 1 auf Ausgang 2 und Eingang 2 auf Ausgang 1

Das 2x2-Schaltelement trifft seine Entscheidung auf Grund von Informationen im Header des Pakets. Beispielsweise kann bei einer VC der Eingangsport aus der aktuellen Tabelle den Ausgangsport entnehmen und als internen Header an das Paket anfügen. Man spricht dann von eigenvermittelnden Schaltnetzen.

Der Entscheidungsprozess aus Sicht eines Pakets verläuft ähnlich wie die Suche in einem binären Baum. An jedem Knoten – Schaltelement – wird ein weiterführender Ast aus zwei Möglichkeiten ausgewählt. Bei einem Header mit n Bit erreicht ein Paket dann in n Schritten einen aus 2^n Endknoten (Ausgangsports). Da es aber mehrere Eingänge gibt, sind mehrere Wurzeln notwendig. Alle Wurzeln führen aber zu den gemeinsamen Blättern, wobei die Pakete auf dem Weg nicht zusammenstoßen dürfen. Eine trickreiche Lösung dieses Problems sind Banyan-Netzwerke, bei denen diese Vergleichsbäume in geschickter Weise ineinander verflochten sind. Bei n Eingängen werden $n/2$ Schalterwerke pro Stufe und $\log_2 n$ Stufen benötigt. Banyan ist der englische Name eines Baumes (*Ficus benghalensis*), bei dem Ableger von den Ästen aus neue Wurzeln bilden.

Banyan-Netzwerke funktionieren allerdings nur gut, wenn die eingehenden Pakete in aufsteigender Reihenfolge sortiert sind. Daher wird ein nach seinem Erfinder benanntes Batchter-Netzwerk vorgeschaltet, das zunächst die Pakete in die richtige Reihenfolge bringt. Das nachgeschaltete Banyan-Netzwerk bringt die Pakete dann an die richtigen Ausgangsports.



© tecCHANNEL

Bild 5.5: Das Batcher-Netzwerk sortiert die Pakete für das Banyan-Netzwerk in aufsteigender Reihenfolge vor, damit dieses effizient arbeiten kann.

Das kombinierte Netzwerk befördert alle Pakete an den richtigen Ausgangsport. Es kann allerdings stets nur ein Paket zu jedem Ausgang leiten. Daher werden zusätzliche Komponenten benötigt, die durch Pufferung und eventuelle parallele Strukturen Zielkollisionen so weit wie möglich vermeiden.

Prof. Dr. Stephan Euler

tecCHANNEL-Links zum Thema	Webcode	Compact
Vermittlung	a1495	S.49
Grundlagen Netzwerkkomponenten	a1092	–
Grundlagen: Layer-4/7-Switching	p1370	–
Ethernet im Überblick	a717	–

Mit den Webcodes gelangen Sie auf www.tecChannel.de direkt zum gewünschten Artikel. Geben Sie dazu den Code in das Feld „Webcode“ in der Titelleiste von www.tecChannel.de ein. Mit „p“ gekennzeichnete Webcodes beziehen sich auf Online-Artikel aus tecCHANNEL-Compact. Es ist möglich, dass Artikel zunächst exklusiv in tecCHANNEL-Compact veröffentlicht werden und daher erst später über den Webcode erreichbar sind.

6. Internet-Protokoll IP

Der Erfolg des Internets zeigt, dass eine weltweite Vernetzung unterschiedlichster Rechner und Systeme möglich ist und damit funktionierende Anwendungen angeboten werden können.

Das Internet ist kein einheitliches Netz, sondern ein Verbund vieler Netze mit unterschiedlicher Technologie. Insgesamt entstand ein riesiger Verbund von Knoten mit weiterhin starkem Wachstum [10]. Bei dem Aufbau des bisherigen Netzes und seinem weiteren Ausbau sind das Kernproblem:

- Heterogenität: Wie können die verschiedensten Netze einwandfrei zusammen funktionieren?
- Adressierung: Wie werden eindeutige und aussagefähige Adressen vergeben?
- Routing: Wie findet eine Nachricht den Weg durch das Netz?
- Skalierbarkeit: Wie kann das Netz das noch zu erwartende starke Wachstum aufnehmen?

Grundlage des Internets ist IP – das Internet Protokoll. IP beinhaltet die Adressvergabe und einen verbindungslosen, unzuverlässigen Datagramm-Dienst. Im OSI-Modell stellt IP die Schicht 3 (Vermittlungsschicht) dar. Unzuverlässig bedeutet, dass keine Garantie bezüglich der richtigen Zustellung von Paketen gegeben wird. Pakete können verfälscht eintreffen, verloren gehen und mehrfach oder in der falschen Reihenfolge ankommen. Das Netzwerk bemüht sich, die Pakete zuzustellen, aber um eventuelle Fehler müssen sich Sender und Empfänger selbst kümmern. Man sagt, das Netz arbeitet nach dem Prinzip Best-Effort.

6.1 Adressen

Eine IP-Adresse (in Version 4, IPv4) besteht aus 32 Bit. Üblicherweise schreibt man die Adressen als vier durch Punkte getrennte Dezimalzahlen, also zum Beispiel 120.56.222.94, wobei jede Zahl acht Bit repräsentiert. Sie kann damit maximal den Wert 255 annehmen. Insgesamt sind 2^{32} verschiedene Adressen zwischen 0.0.0.0 und 255.255.255.255 möglich.

Im Gegensatz zu den Ethernet-Adressen sind die IP-Adressen hierarchisch aufgebaut. Ethernet-Adressen sind nur hierarchisch bezüglich des Herstellers der Netzwerkkarte – eine Information, die allerdings beim Routing nicht hilft. Der erste Teil der IP-Adresse bezeichnet ein Netzwerk und der zweite Teil einen Knoten. Dadurch vereinfacht sich das Routing, da zunächst nur die Netzwerkadresse ausgewertet werden muss.

Netzwerke sind unterschiedlich groß. Beispielsweise haben Firmen verschieden große Netze. So unterscheidet sich der Bedarf eines weltweiten Großkonzerns sehr von dem einer kleinen Software-Firma. Deshalb wurde ein flexibles Schema mit mehreren Klassen eingeführt. Die Position der ersten Null in der Adresse legt

die Klasse fest. Ist das erste Bit Null (das heißt in der Hälfte der Fälle), so handelt es sich um eine Adresse der Klasse A. In diesem Fall sind die nächsten sieben Bit die Netzwerkadresse, und die restlichen 24 Bit bezeichnen den Knoten. Berücksichtigt man noch zwei Sonderfälle, verbleiben 126 Adressen für Netze der Klasse A. Diese 126 Netze belegen bereits die Hälfte aller zur Verfügung stehenden Adressen. In Tabelle 6.1 ist die Zuordnung der Bits zu den Feldern für die verschiedenen Klassen zusammengestellt.

Tabelle 6.1: Aufbau der IP-Adressen in verschiedene Klassen						
8	16		24	32	Bit	Klasse
0	Netz-ID		Host-ID			A
1	0	Netz-ID		Host-ID		B
1	1	0	Netz-ID		Host-ID	C
1	1	1	0	Multicast-Adresse		D

Betrachten wir als Beispiel die Adresse 212.201.24.18. Das erste Byte hat den Dezimalwert 212. Die Umwandlung in die Darstellung als Binärzahl ergibt 1101 0100. Die erste Null ist an der dritten Stelle von links. Es handelt sich also um eine Adresse aus der Klasse C.

Die Grundidee bei der Festlegung der Klassen war, dass es nur wenige sehr große Netze mit vielen Knoten, aber sehr viele kleine Netze mit wenigen Knoten gibt. Die jeweilige Anzahl der Netze und darin enthaltenen Knoten finden Sie in Tabelle 6.2. Dabei ist berücksichtigt, dass einige Adressen besondere Bedeutung haben. Solche mit dem Wert 0 beziehen sich auf das eigene Netz oder den eigenen Knoten. Sind andererseits alle Bits im Knotennamen gesetzt, so handelt es sich um eine Broadcast-Nachricht. Die Adresse 127 in der Klasse A wird für Tests benutzt, bei denen ein Knoten sein gesendetes Paket zurückerhält (Loop-Back).

Tabelle 6.2: Klasseneinteilung		
Klasse	Anzahl Netze	Anzahl Knoten
A	126	16.777.214
B	16.384	65.534
C	2.097.152	254

Die Netznummern werden vom Network Information Center (NIC) beziehungsweise jetzt durch die Internet Assigned Numbers Authority (IANA, www.iana.org/) verwaltet. In der Praxis hat sich die Einteilung als nicht optimal erwiesen. Es gibt einerseits zu wenige Netze der Klasse B für große Firmen oder Institutionen. Andererseits ist die Klasse B für die meisten Firmen überdimensioniert, während

die Klasse C zu klein ist oder zumindest zu wenig Reserven hat. Eine Lösung, die die starre Einteilung überwindet, ist Classless InterDomain Routing (CIDR). Wie der Name zeigt, ersetzt CIDR die feste Klasseneinteilung durch eine flexible Aufteilung der Bits zwischen Netzkennung und Knotennummer. Gleichzeitig wurden die Adressen den vier Weltzonen Europa, Nordamerika, Mittel-/Südamerika sowie Asien/Pazifik zugeordnet, um das Routing zu vereinfachen.

Eine andere Verfeinerung des Adressraums erreicht man durch Subnetzadressierung. Angenommen eine Firma betreibt mehrere LANs, die aber unter einer einheitlichen Netzadresse angesprochen werden sollen. Um die interne Weiterleitung zu vereinfachen, werden die einzelnen LANs als Subnetze behandelt. Dazu kommt ein Teil der Bits der Host-ID als Subnetz-ID zum Einsatz. Wie viele Bits für diese weitere Hierarchie zur Verfügung stehen, spezifiziert eine Subnetzmaske. Angeschlossene Knoten können dann an der Adresse erkennen, ob das Paket in ihrem eigenen LAN bleibt oder über einen Switch in ein anderes LAN soll.

Die Subnetzmaske hat die Form einer IP-Adresse. Gesetzte Bits kennzeichnen den Netzwerkteil inklusive Subnetzadresse. Beispielsweise bedeutet eine Subnetzmaske 255.255.255.0, dass nur die letzten acht Bit die Knotennummer beinhalten. Alle anderen Bits, bei denen die Subnetzmaske den Wert Eins hat, gehören zur Netzwerkadresse. Die genaue Interpretation – Netzadresse und Subnetzadresse – hängt von der Klasse der Adresse ab. Bei einer Adresse der Klasse B enthalten die ersten 16 Bit die Netzwerkadresse. Bei der Subnetzmaske 255.255.255.0 bilden dann die nächsten 8 Bit, die ursprünglich zur Host-Adresse gehörten, die Subnetzadresse.

Der Netzwerkteil einer Adresse lässt sich leicht durch bitweise UND-Verknüpfung mit der Subnetzmaske berechnen. Als Beispiel berechnen wir in Tabelle 6.2 für die Adresse 212.201.25.18 und die Subnetzmaske 255.255.252.0 die Adresse des Netzwerkteils.

Tabelle 6.3: Adressberechnung des Netzwerkteils		
IP-Adresse	212.201.25.18	11010100.11001001.00011001.00010010
Subnetzmaske	255.255.252.0	11111111.11111111.11111100.00000000
bitweise UND		11010100.11001001.00011000.00000000
Netzwerkteil	212.201.24.0	

Um das prinzipielle Problem von zu wenigen Adressen zu lösen, wird in der Version IPv6 deren Größe auf 16 Byte erhöht. Damit stehen selbst bei schlechter Ausnutzung des Adressraums rechnerisch mehr als 1000 IP-Adressen pro Quadratmeter der Erdoberfläche zur Verfügung.

Die IP-Adressen der einzelnen Knoten können – beispielsweise vom Systemadministrator – fest vergeben werden. Eine Alternative sind dynamische Adressen. Damit können insbesondere für temporäre Verbindungen zeitlich befristete Adres-

sen vergeben werden. Über das Protokoll DHCP (Dynamic Host Configuration Protocol) erhält ein Rechner von einem DHCP-Server eine IP-Adresse. Der DHCP-Server verwaltet die Nummern und sorgt für deren eindeutige Vergabe. Der Client erhält die Adresse nur für einen bestimmten Zeitraum (Lease). Vor dem Ablauf der Gültigkeitsdauer muss er sie dann wieder verlängern oder eine neue Lease beziehen. Beispiel:

```
ipconfig /all
```

Ethernet-Adapter „LAN-Verbindung“:

```
Verbindungsspezifisches DNS-Suffix: fh-friedberg.de
Beschreibung. . . . . : Accton EN2242 Series
                        MiniPCI Fast Ethernet Adapter
Physikalische Adresse . . . . . : 00-D0-59-6A-88-63
DHCP-aktiviert. . . . . : Ja
Autokonfiguration aktiviert . . . : Ja
IP-Adresse. . . . . : 212.201.26.252
Subnetzmaske. . . . . : 255.255.255.0
Standardgateway . . . . . : 212.201.26.1
DHCP-Server . . . . . : 212.201.26.1
...
Lease erhalten. . . . . : Dienstag, 22. Juni
➡2004 14:21:28
Lease läuft ab. . . . . : Dienstag, 22. Juni
➡2004 14:31:28
```

6.2 Paketformat

Tabelle 6.4 zeigt das Format der IP-Pakete (Datagramme) in der Version IPv4.

Tabelle 6.4: Das IPv4 Paket-Format im Detail						
0	4	8	16	19	31	Bit
Version	IHL	TOS	Gesamtlänge			
Ident			Flags	Offset		
TTL		Protocol	Checksum			
Quellenadresse						
Zieladresse						
Optionen + Pad						
Daten						

Die Darstellung ist in Vielfachen von 32 Bit organisiert. Die Bedeutung der einzelnen Felder ist:

- **Version:** IP Version (hier 4)
- **IHL:** Internet Header Length. Die Länge des Headers in Vielfachen von 32 Bit.
- **TOS:** Type of Service. In diesem Feld können Eigenschaften für die Übertragung spezifiziert werden.
- **Gesamtlänge:** Datagramm-Länge (Header plus Daten) in Bytes. Die maximale Größe beträgt 65.535 Byte.
- **Ident:** Identifikation des Datagramms. Notwendig, falls das Datagramm für die unteren Übertragungsschichten in kleinere Pakete – so genannte Fragmente – (zum Beispiel bei Ethernet maximal 1500 Byte) zerlegt werden muss. Jedes Fragment enthält den vollständigen IP-Header zusammen mit einem Anteil der Daten.
- **Flags:** Informationen über die Fragmentierung.
- **Offset:** Die laufende Nummer des ersten Bytes im Datenteil relativ zum ersten Byte des gesamten Datagramms.
- **TTL:** Time To Live. Zähler für die maximale Lebensdauer eines Datagramms. Der Sender setzt TTL auf einen Startwert (zurzeit 64). Jeder Router auf dem Weg dekrementiert TTL. Ist der Wert Null erreicht, wird das Datagramm vernichtet.
- **Protocol:** Spezifikation für das höhere Protokoll (zum Beispiel 6 für TCP und 17 für UDP).
- **Checksum:** Prüfsumme für den Header (Summe in Einerkomplement)
- **Quellenadresse und Zieladresse:** Die vollständigen IP-Adressen von Sender und Empfänger.
- **Optionen + Pad:** Mögliche zusätzliche Optionen und eventuelle Füllbits. Die Anzahl der Optionen berechnet sich aus der angegebenen Header-Länge.

6.3 Weiterleitung

Wenn ein Knoten – ein Rechner oder ein Router – ein Datagramm verschicken oder weiterleiten will, muss er die IP-Adresse analysieren. Zunächst prüft er die Netzwerkadresse. Stimmt die Netzwerkadresse mit seiner eigenen überein, erkennt der Knoten, dass der Zielknoten sich in seinem eigenen physikalischen Netzwerk befindet. Rechner sind normalerweise nur an einem physikalischen Netzwerk angeschlossen, während Router zwei oder mehr Netzschnittstellen haben. Router überprüfen daher, ob die Netzwerkadresse zu einem ihrer Netzanlüsse passt.

Befindet sich das Ziel im gleichen physikalischen Netzwerk, so kann der Knoten die zugehörige interne Adresse (zum Beispiel eine Ethernet-Adresse) ermitteln und das Datagramm direkt zustellen. Wie dies im Detail geschieht, sehen wir im nächsten Abschnitt.

Ist das Ziel außerhalb des eigenen Netzwerkes, wird das Datagramm an einen Router weitergeleitet. Dieser wirkt als Schnittstelle in die externen Netzwerke. Im Allgemeinen wird der Router selbst auch keine direkte Verbindung zu dem Ziel haben, sondern das Paket wieder an einen Router weiterreichen, bis irgendwann ein Router mit Zugang zu dem Zielnetz erreicht ist.

Die Weiterleitung basiert auf Tabellen. Ähnlich wie in Switches gibt es Weiterleitungstabellen für die IP-Adressen. Im einfachsten Fall enthält die Tabelle eine paarweise Zuordnung von Netzwerkadressen und nächsten Routern. Für die Weiterleitung des Datagramms schlägt der Knoten in seiner Weiterleitungstabelle nach, an welchen nächsten Knoten er Datagramme für die gegebene Netzwerkadresse schicken soll. Den ausgewählten Router nennt man Next-Hop-Router.

Die Weiterleitungstabelle ist in der Regel nicht vollständig. Daher gibt es noch einen Eintrag für einen Default-Router, an den alle nicht direkt zuzuordnenden Datagramme geschickt werden. Ein Rechner hat häufig nur Zugang zu einem Router. In diesem Fall ist die Weiterleitungstabelle leer beziehungsweise enthält nur den Eintrag für diesen Default-Router.

Insgesamt gesehen erfolgt die Zustellung in zwei Schritten. Zunächst wird das Datagramm in das richtige Netzwerk geschickt und anschließend dort an den Zielknoten zugestellt. Dadurch wird der Verwaltungsaufwand beträchtlich reduziert. Die Router müssen nicht für jeden Rechner in der Welt Einträge in ihre Weiterleitungstabellen aufnehmen, sondern nur noch für die Netze. Bleibt die Frage, wie die Router zu ihren Weiterleitungstabellen kommen? Dies ist Thema des übernächsten Abschnitts.

6.4 Zuordnung IP-Adresse zu Ethernet-Adresse

Wenn ein Knoten feststellt, dass der Zielknoten sich im gleichen Netzwerk befindet, so kann er das Datagramm direkt an dessen Adresse schicken. Sind etwa die beiden Knoten über Ethernet verbunden, so benötigt er dazu die Ethernet-Adresse des Zielknotens. Die Zuordnung von IP-Adressen zu Ethernet-Adressen erfolgt über das Adressauflösungsprotokoll (Address Resolution Protocol ARP).

Wieder ist die Information in einer Tabelle gespeichert. Jeder Knoten führt eine Zuordnungstabelle mit IP-Adressen und Ethernet-Adressen, die ARP-Tabelle oder auch ARP-Cache. Da die Adresszuordnung sich jederzeit ändern kann, wird die Tabelle dynamisch verwaltet.

Angenommen ein Knoten S möchte zum ersten Mal ein Datagramm an einen bestimmten Zielknoten Z schicken. Zu diesem Zeitpunkt gibt es noch keinen Eintrag für Z in der ARP-Tabelle. Daher schickt S eine Anfrage (Rundruf) an alle Knoten

im eigenen Netz. Bei Ethernet wird diese Anfrage als Broadcast gleichzeitig an alle geschickt. Die Anfrage enthält die Zieladresse und die eigene Adresse von S. Jeder Knoten empfängt die ARP-Anfrage. Knoten Z erkennt seine eigene Adresse und reagiert, indem er eine Antwortnachricht mit seiner eigenen Ethernet-Adresse zurückschickt. Gleichzeitig fügt Z die Adresse von S in seine eigene ARP-Tabelle ein, in der Annahme, dass er bald Datagramme an S schickt. S erhält die Antwort, erweitert seine ARP-Tabelle und schickt das Datagramm. Bei weiteren Datagrammen nutzen S und Z die Einträge in ihren ARP-Tabellen.

Um die ARP-Tabelle aktuell zu halten, wird jeder Eintrag nach 15 Minuten automatisch entfernt. Findet in der Zwischenzeit eine weitere Kommunikation mit dem Knoten statt, verlängert sich die Lebensdauer des Eintrags entsprechend. Sie verlängert sich auch, wenn ein Knoten eine ARP-Abfrage von einem anderen Knoten sieht. Diese Aktualisierung nehmen alle Knoten vor, die einen Eintrag für den Sender haben. Nicht betroffene Knoten ignorieren die ARP-Abfrage und erzeugen keinen neuen Eintrag.

Mit dem Programm `arp` kann man die Übersetzungstabellen anschauen und manuell ändern. Es lassen sich Einträge löschen oder einfügen. Manuell eingefügte Einträge sind permanent und werden auch bei Zeitüberschreitung nicht aus dem Cache gelöscht. Ein Beispiel:

```
arp -a
```

```
Schnittstelle: 212.201.26.252 on Interface 0x1000003
  Internetadresse      Physikal. Adresse      Typ
  212.201.26.1         00-20-18-58-b8-1f      dynamisch
```

Das Gegenstück zu ARP ist RARP (Reverse Address Resolution Protocol). Damit können Computer ohne eigenen Massenspeicher von einem RARP-Server ihre IP-Adresse erfragen.

6.5 Internet Control Message Protocol

Das Internet Protokoll wird ergänzt durch das Internet Control Message Protocol (ICMP). Über ICMP tauschen Hosts und Router Nachrichten aus. Dazu gehören unter anderem Fehlermeldungen, Informationen zur Steuerung der Übertragung sowie Echo-Anfragen und Antworten. Um welche Art von Nachricht es sich handelt, wird in einem Typfeld im Kopf des Pakets spezifiziert. Abhängig vom konkreten Typ folgen dann die dazu gehörenden Daten. Eine ICMP-Nachricht wird im Nutzdatenfeld eines IP-Pakets übertragen.

Eine einfache Anwendung auf der Basis von ICMP ist der Befehl Ping. Ping steht für Packet InterNet Groper (das Verb *grope* bedeutet greifen oder suchend nach etwas tasten). Gleichzeitig ist Ping auch die Bezeichnung für einen kurzen Impuls, wie er bei Echolot oder Radarsuchgeräten ausgesendet wird.

Die gleiche Funktion hat Ping im Netzwerk: Es schickt eine ICMP-Nachricht mit einer Echo-Anforderung an einen Host und wartet auf das Echo. Ein Beispiel:

```
Ping www.w3.org [128.30.52.25] mit 32 Bytes Daten:
```

```
Antwort von 128.30.52.25: Bytes=32 Zeit=160ms TTL=55
Antwort von 128.30.52.25: Bytes=32 Zeit=150ms TTL=55
Antwort von 128.30.52.25: Bytes=32 Zeit=150ms TTL=55
Antwort von 128.30.52.25: Bytes=32 Zeit=150ms TTL=55
```

```
Ping-Statistik für 128.30.52.25:
```

```
  Pakete: Gesendet = 4, Empfangen = 4, Verloren = 0 (0%
  ↗Verlust),
```

```
Ca. Zeitangaben in Millisek.:
```

```
  Minimum = 150ms, Maximum = 160ms, Mittelwert = 152ms
```

In diesem Fall wurden vier Pakete geschickt und bei allen kam eine Antwort. Das Programm gibt die jeweils benötigte Zeit aus und berechnet aus den einzelnen Werten eine kleine Statistik. Eine solche erfolgreiche Ausführung zeigt, dass die Netzwerkverbindung zu dem angegebenen Rechner funktioniert. Damit ist Ping ein nützliches Prüfwerkzeug. Falls Fehler auftreten, kann man damit schnell prüfen, ob die Kommunikation bis zur Ebene von IP zu Stande kommt. So lassen sich elementare Fehler wie zum Beispiel ein ausgeschalteter Zielknoten oder ein falscher Knotenname aufdecken.

6.6 Routing – eine kurze Einleitung

In der bisherigen Diskussion hatten wir behandelt, wie ein Router Pakete weiterleitet. Grundlage sind Weiterleitungstabellen, die die benötigte Information beinhalten. Wir haben gesehen, wie damit das Problem der Weiterleitung effizient gelöst wird. Weit gehend ausgeblendet hatten wir bisher, wie die Weiterleitungstabellen gefüllt werden. Die zu Grunde liegende Frage ist: Wie wird die Verbindung zwischen zwei Knoten S und E festgelegt?

Wenn man das Netzwerk als Graphen sieht – eine Menge durch Kanten verbundener Knoten – ist die entsprechende Frage nach der kürzesten Verbindung zwischen zwei Knoten. In Bild 6.1 ist ein Netzwerk als Graph dargestellt. Er repräsentiert allerdings nur die Verbindungen zwischen den Routern. In der Realität gehört dann zu einem Router wiederum ein lokales Netz, in dem Pakete mit den beschriebenen Methoden weiter verteilt werden.

Der Einfachheit halber wird nur zwischen verbundenen und nicht verbundenen Knoten unterschieden. Im Allgemeinen sind die Kosten (Latenz, Bandbreite, Gebühren) für verschiedene Verbindungen (Kanten) unterschiedlich. Dies ist zu berücksichtigen, indem man entsprechende Maßzahlen an die Kanten anbringt. Die Suche nach der optimalen Verbindung muss dann die entstehenden Kosten berücksichtigen.

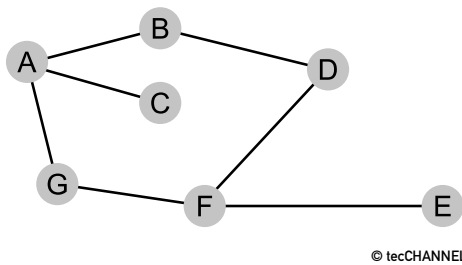


Bild 6.1: Ein Netzwerk mit Knoten A bis G als Graph dargestellt.

Die Suche nach dem kürzesten Pfad in einem Graphen ist ein Standardproblem der Informatik. Eine klassische Anwendung ist die Bestimmung einer optimalen Route zum Beispiel für ein Auto-Navigationssystem. Die Kosten der Kanten sind in diesem Fall die entsprechenden Fahrzeiten, bestimmt durch Entfernung und Straßentyp. Zur effizienten Lösung des Problems existieren Standardlösungen. Grundvoraussetzung ist allerdings die Kenntnis des Graphen.

Die genaue Kenntnis des Graphen – das heißt das Wissen, welche Knoten aktiv sind, welche Kanten es gibt sowie die eventuellen Kosten einer Verbindungsstrecke – ist bei einem großen Netz mit ständigen Veränderungen kaum zentral verfügbar. Daher beruhen die erforderlichen Routing-Verfahren auf verteilten Algorithmen. Jeder Knoten teilt anderen Knoten sein Wissen über das Netz mit. Aus diesen Informationen erstellt jeder Knoten für sich eine Sicht des Netzes. Im Idealfall konvergieren die Algorithmen, so dass nach einer gewissen Zeit alle Knoten eine einheitliche Sicht haben.

Die Aufgabe des Routings hängt sehr von dem betrachteten Netzwerk ab. Solange es sich um ein weitgehend einheitliches Netz unter gemeinsamer Verwaltung handelt, ist die Information über den Graphen relativ leicht zu erhalten. Betrachtet man demgegenüber das weltweite Internet, so ist eine einheitliche Sicht nicht mehr möglich. Verschiedene Provider haben unterschiedliche Sichtweisen der Kosten für bestimmte Verbindungen. So wird zum Beispiel ein Provider bestrebt sein, möglichst häufig eigene Verbindungen zu benutzen, selbst wenn der resultierende Pfad nicht der kürzeste ist.

Im Folgenden wird daher zwischen Intradomain- und Interdomain-Routing unterschieden. Man betrachtet ein abgeschlossenes System mit einheitlicher Administration als autonomes System (AS). Beispiele für autonome Systeme sind das interne Netzwerk eines Großunternehmens oder das Netzwerk eines Service Providers. Dementsprechend unterscheidet man zwischen internem (intradomain) und externem (interdomain) Routing.

Die Vorgehensweise bei Intradomain-Routing behandeln wir im Folgenden hauptsächlich am Beispiel des Distanzvektor-Routings. Bei der Betrachtung des Interdomain-Routings beschränkt sich die Darstellung auf einige allgemeine Eigenschaften des im Internet eingesetzten Protokolls.

6.7 Distanzvektor-Routing

Der Algorithmus des Distanzvektor-Routings basiert auf den Informationen der Knoten über ihre unmittelbaren Nachbarn. Betrachten wir lediglich die Anzahl der Kanten bis zu einem anderen Knoten (Hops): Ein Knoten erstellt zunächst eine Tabelle, in der alle direkten Nachbarn den Abstand 1 und sämtliche anderen Knoten den Abstand ∞ haben. In dem obigen Beispiel hat Knoten A die Entfernungstabelle 6.5.

Tabelle 6.5: Entfernungstabelle von Knoten A

Knoten	A	B	C	D	E	F	G
A	–	1	1	∞	∞	∞	1

In gleicher Weise erstellen auch alle anderen Knoten ihre initiale Entfernungstabelle. Insgesamt erhält man die Tabelle 6.6.

Tabelle 6.6: Anfangswerte aller Knoten-Entfernungstabellen

Knoten	A	B	C	D	E	F	G
A	–	1	1	∞	∞	∞	1
B	1	–	∞	1	∞	∞	∞
C	1	∞	–	∞	∞	∞	∞
D	∞	1	∞	–	∞	1	∞
E	∞	∞	∞	∞	–	1	∞
F	∞	∞	∞	1	1	–	1
G	1	∞	∞	∞	∞	1	–

Die Tabelle existiert nur in verteilter Form. Jeder Knoten sieht nur seine eigene Zeile. Im nächsten Schritt schickt jeder Knoten seine Information an seine Nachbarn. Knoten A erfährt so von B, dass B mit einem Hop D erreichen kann. Jetzt weiß A, dass er D über B mit zwei Hops erreicht. Knoten A ergänzt seine Tabelle und trägt den nächsten Hop (Port) für jeden Knoten ein (Tabelle 6.7).

Tabelle 6.7: Entfernungstabelle Knoten A nach einer Iteration

Knoten	A	B	C	D	E	F	G
A	–	1	1	2	∞	2	1
Port	–	B	C	B	–	G	G

Wenn alle Knoten die Informationen von ihren Nachbarn eingetragen haben, ergeben sich die Werte in Tabelle 6.8. Nach der ersten Iteration kennt Knoten G die Entfernungen und die nächstgelegenen Knoten zu allen anderen Knoten im Netzwerk. Knoten E fehlen demgegenüber noch Informationen über A, B und C.

Tabelle 6.8: Entfernungstabellen nach einer Iteration

Knoten	A	B	C	D	E	F	G
A	–	1	1	2	∞	2	1
B	1	–	2	1	∞	2	2
C	1	2	–	∞	∞	∞	2
D	2	1	∞	–	2	1	2
E	∞	∞	∞	2	–	1	2
F	2	2	∞	1	1	–	1
G	1	2	2	2	2	1	–

Im nächsten Schritt verschicken die Knoten diese erweiterten Tabellen – genauer gesagt jeweils ihre eigene Zeile – an ihre Nachbarn. Die Knoten werten die Informationen aus und ergänzen ihre eigenen Tabellen. Dieses Verfahren – Versenden des aktuellen Kenntnisstandes und Aktualisieren der Tabellen – wird kontinuierlich fortgesetzt. Die Knoten prüfen dabei, ob sie zu einem Knoten eine kürzere Verbindung finden, als bisher in der Entfernungstabelle eingetragen. Für unser Beispielnetz ergibt sich Tabelle 6.9.

Tabelle 6.9: Entfernungstabellen nach vier Iterationen

Knoten	A	B	C	D	E	F	G
A	–	1	1	2	3	2	1
B	1	–	2	1	3	2	2
C	1	2	–	3	4	3	2
D	2	1	3	–	2	1	2
E	3	3	4	2	–	1	2
F	2	2	3	1	1	–	1
G	1	2	2	2	2	1	–

Auf diese Weise bauen die Knoten die Information für ihre Weiterleitungstabellen auf. Zwar hat kein Knoten die vollständige Sicht über das komplette Netzwerk. Jeder der Knoten weiß nun aber genug, um die Pakete an den jeweils richtigen Folgeknoten zu schicken.

Die Knoten senden ihre Informationen periodisch oder wenn sie eine Änderung festgestellt haben. Zum Beispiel kann ein Knoten feststellen, dass einer seiner Nachbarn keine periodischen Aktualisierungen mehr sendet. Er wird dann davon ausgehen, dass die Verbindung zu diesem Knoten gestört ist und die Einträge in seiner Tabelle, die über diesen Knoten laufen, mit der Distanz unendlich belegen. Diese Information schickt er an seine anderen Nachbarn, die daraufhin ebenfalls ihre Tabellen aktualisieren.

Im Normalfall konvergiert der Algorithmus relativ schnell wieder zu einem stabilen Zustand. Allerdings kann es zu Problemen kommen, wenn im Netz Schleifen vorhanden sind. Es gibt Verfeinerungen des Verfahrens, die das Verhalten in solchen Fällen verbessern. Ein verbreitetes Protokoll, mit dem Router ihre Weiterleitungsinformationen austauschen, ist das Routing Information Protocol RIP. RIP implementiert den beschriebenen Distanzvektor-Algorithmus. Die Informationen werden durch Pakete im RIP-Format verschickt. Die Aktualisierung erfolgt nach jeweils 30 Sekunden.

Ein alternatives Verfahren ist das Link-State-Routing. Im grundsätzlichen Unterschied zum Distanzvektor-Routing verschickt dabei ein Knoten nur die Informationen über seine Nachbarn, nicht sein gelerntes Wissen über weiter entfernte Knoten. Diese eingeschränkte Information sendet jeder Router allerdings an alle anderen. Er setzt dazu ein als zuverlässiges Fluten bezeichnetes Verfahren ein.

6.8 Interdomain-Routing

Das Internet besteht aus einer Anzahl von autonomen Systemen (AS). Dabei gibt es verschiedene Klassen:

- AS mit nur einer Verbindung zu einem anderen AS: zum Beispiel eine kleine Firma mit einem internen Netz und nur einem Router (Border-Gateway) als Schnittstelle nach extern.
- AS mit mehreren externen Verbindungen: zum Beispiel eine große Firma mit mehreren Standorten. Innerhalb des AS findet aber nur lokaler Verkehr statt.
- AS mit mehreren externen Verbindungen und der Bereitschaft, Pakete für andere AS zu übermitteln: zum Beispiel ein Backbone-Provider.

Aufgabe des Interdomain-Routings ist es, die Erreichbarkeit zwischen den AS zu gewährleisten. Die Ansprüche sind dabei wesentlich geringer als beim Intradomain-Routing. Wesentliches Ziel ist es, irgendeinen Weg ohne Schleifen zu finden. Die Festlegung eines optimalen Pfads ist kaum möglich, da dann beispielsweise alle AS die gleiche Kostenmetrik verwenden müssten. Außerdem sind Kriterien wie „möglichst lange im eigenen Netz“ oder „nur durch vertrauenswürdige Netze“ schwer zu formalisieren.

Im Internet wird das Border Gateway Protocol BGP eingesetzt. BGP ist ausgelegt, um das Routing in einem Verbund von AS zu ermöglichen. Jedes AS erhält eine eindeutige Nummer (16 Bit). Jedes AS benennt mindestens einen BGP-Sprecher.

Die BGP-Sprecher tauschen untereinander die Routing-Information in Form von vollständigen Weglisten aus. Ein Sprecher könnte beispielsweise mitteilen, dass er das Netzwerk 178.53.66 auf dem Weg <AS15, AS356, AS27> erreichen kann. Mit entsprechenden Aktualisierungspaketen verschickt ein BGP-Sprecher die ihm bekannten Wege oder meldet sie auch wieder ab.

Durch die Kombination von Intradomain- und Interdomain-Routing wird der große, heterogene Verbund des Internets überschaubar. Ein Router innerhalb eines AS einer Firma braucht beispielsweise nur das Border-Gateway zu kennen. Er schickt dann alle nicht intern zustellbaren Pakete an diesen Router. Umgekehrt benötigt auch der Router eines Backbone-Providers nicht alle Detailkenntnisse. Es ist ausreichend, wenn er die Netze den einzelnen AS zuordnen kann.

6.9 Domain Name System DNS

Zur besseren Übersichtlichkeit verwendet man statt der numerischen IP-Adressen aussagekräftige Namen für Knoten. Die Auflösung der Namen erfolgt durch das Domain Name System DNS. Die Namen sind dabei hierarchisch organisiert, wobei die einzelnen Namensteile durch Punkte getrennt sind. Die Reihenfolge ist nach zunehmender Größe geordnet.

Ein Beispiel ist premium.tecChannel.de. Der Knoten premium gehört zur Domäne tecChannel, die wiederum Teil von de ist. Die Top Level Domains sind entweder die Länderkürzel oder Zuordnungen wie edu (Education) oder mil (Military).

Die gesamte Hierarchie ist in so genannte Zonen eingeteilt, wobei es für jede eine verantwortliche Instanz gibt. Jede Zone stellt mindestens zwei Name-Server bereit, die auf Anfrage Informationen über die Adressen erteilen.

Wenn eine Anwendung die IP-Adresse eines Knotens benötigt, schickt sie eine Anfrage mit dem Namen an einen lokalen Name-Server. Wenn dieser den Namen noch nicht kennt, leitet er eine Abfrage an den Name-Server in seiner Zone weiter. Von dort erhält er entweder die gewünschte Zieladresse oder die Adresse eines weiteren Name-Servers.

Im zweiten Fall fragt der lokale Name-Server bei dem weiteren Name-Server nach. Dieser Prozess wird fortgesetzt, bis ein Server gefunden ist, der die gewünschte Adresse angeben kann.

Zur Verringerung des Abfrageverkehrs merken sich die Name-Server in einem Cache für eine gewisse Zeit die erfragten Adressen. So kann eine neue Frage nach der Adresse direkt beantwortet werden.

Die Basis des Domain Name System ist eine einfache Datenbank mit Ressourcendatensätzen. Ein Datensatz enthält dabei ein Typfeld, mit dem verschiedene Arten von Informationen spezifiziert werden können. Das Domain Name System bietet dadurch Möglichkeiten wie die Definition von Alias-Namen oder die Festlegung des Mail-Servers.

6.10 Internet-Standards

Das Internet ist ein loser Verbund vieler einzelner Rechner. Für den Endbenutzer ist in der Regel nur sein Internet-Provider relevant. Gegen Gebühren erhält er von ihm den Zugang und die Möglichkeit zur Datenübertragung. Die Infrastruktur des Internets kann er kostenfrei nutzen. Ob eine angefragte Adresse in der Nachbarschaft oder am anderen Ende der Welt liegt, spielt keine Rolle.

Gleichzeitig ist das Internet ein wesentlicher Teil der globalen Infrastruktur der Informationsgesellschaft. Die Ausgestaltung und die Weiterentwicklung des Internets wirft damit eine Vielzahl von technischen, wirtschaftlichen, politischen und kulturellen Fragen auf. Das Zusammenspiel verschiedenster Systeme setzt klare Standards voraus. Diese bedürfen einer ständigen Anpassung an die technische Weiterentwicklung. Gleichberechtigter Zugang zu neuen Standards ist eine wesentliche Voraussetzung für einen fairen Wettbewerb. Fragen nach erlaubten Inhalten und die Garantie eines freien und sicheren Flusses von Informationen berühren unmittelbar unsere Lebenssituation.

Im Vergleich zu anderen Bereichen ist die Entwicklung des Internets geprägt von einem Geist der Offenheit und Freiwilligkeit. Weder staatliche Einrichtungen noch Firmen mit marktbeherrschender Stellung bestimmen das Internet. Vielmehr entwickelte sich eine Reihe von Mechanismen zur selbstbestimmten Regulation und Weiterentwicklung. Die Dachorganisation für die verschiedenen Aktivitäten zur Koordination des Internets ist die Internet Society (ISOC, www.isoc.org/). Ihr Ziel ist zusammengefasst im Mission Statement „To assure the open development, evolution and use of the Internet for the benefit of all people throughout the world.“ Die Mitgliedschaft in der ISOC ist kostenfrei und steht jedem Interessenten offen. Neben der internationalen ISOC gibt es nationale Vereine. So wurde 1995 aus der 1992 gegründeten Deutsche Interessengemeinschaft Internet (DIGI e.V.) unter dem Namen ISOC.DE das „German Chapter“ der Internet Society.

Technische Themen werden in speziellen Gruppen behandelt. Allgemeine grundlegende Fragen zur Architektur des Internets sind Thema des Internet Architecture Board (IAB). Spezielle Fragen wie Protokolle werden in den verschiedenen Working Groups der Internet Engineering Task Force (IETF) behandelt. Es gibt keine formale Mitgliedschaft in der IETF. Wer in einer der zahlreichen Arbeitsgruppen mitwirken möchte, lässt sich einfach in den entsprechenden Email-Verteiler aufnehmen. Erste Versionen von technischen Dokumenten haben den Status von „Internet Drafts“. Sie werden in den Arbeitsgruppen zur Diskussion gestellt. Hat ein Internet Draft einen stabilen Stand erreicht, so kann er als Request for Comments (RFC) publiziert werden. Die einzelnen RFC werden fortlaufend nummeriert. Der beispielsweise im Juni 2004 aktuellste ist RFC 3846 „Mobile IPv4 Extension for Carrying Network Access Identifiers“.

Über die Seite www.ietf.org/rfc.html hat man Zugang zu allen RFCs. Wichtig ist auch dieser Stelle die Idee des freien Zugriffs auf die Dokumente sowie die Möglichkeit, die beschriebenen Protokolle lizenzfrei nutzen zu können.

Neben technischen Dokumenten wie Spezifikationen oder Ähnlichem sind auch mehr beschreibende Dokumente unter den RFC. Ein Beispiel ist RFC 1118 „Hitchhikers guide to the Internet“ mit einer allgemeinen Einführung in die Internet-Technologie. RFC 2828 „Internet Security Glossary“ ist ein umfangreiches Glossar von mehr als 200 Seiten.

Aus einigen RFC werden Internet-Standards. Der Ablauf der Standardisierung ist im Detail in RFC 2026 beschrieben. Durchgeführt wird dieser Prozess von der Internet Engineering Steering Group (IESG). Die IESG wirkt außerdem als Schnittstelle zu anderen Standardisierungsgremien wie ITU.

Einige Namen und Nummern im Internet müssen fest und eindeutig vergeben werden. Dazu zählen etwa die IP-Netzwerkadressen oder die Endungen für die Top Level Domains. Derartige Werte wurden ursprünglich von der Internet Assigned Numbers Authority (IANA) zentral vergeben und verwaltet. Mittlerweile hat Internet Corporation for Assigned Names and Numbers (ICANN) diese Aufgabe übernommen. Dabei wurde die Zuständigkeit auf vier Regional Internet Registries (RIRs) verteilt [11]. Das RIR für den europäischen Bereich ist das RIPE Network Coordination Centre (RIPE NCC, www.ripe.net).

Prof. Dr. Stephan Euler

tecCHANNEL-Links zum Thema	Webcode	Compact
Internet Protokoll IP	a1496	S.58
Dynamic Host Configuration Protocol	a206	–
Domain Name System	a205	–
Hypertext Transfer Protocol	a208	–
So funktionieren TCP/IP und IPv6	p209	–
Grundlagen Netzwerkkomponenten	a1092	–
Grundlagen: Layer-4/7-Switching	p1370	–

7. UDP und TCP

Bisher haben wir betrachtet, wie Pakete von einem Rechner zu einem anderen Rechner geschickt werden. Die eigentliche Kommunikation erfolgt zwischen Anwendungen, das heißt Prozessen auf den jeweiligen Betriebssystemen.

Beispiele für Anwendungen sind Terminalprogramme wie Telnet, Datentransfer über FTP, E-Mail-Programme sowie Internet-Browser.

Diese Anwendungen kommunizieren mit einem entsprechenden Partner auf einem entfernten Rechner. Man spricht daher von Ende-zu-Ende-Protokollen und der Transportschicht. Diese Protokolle sind Mittler zwischen der Vermittlungsschicht und den Anwendungen. Sie müssen mit den Möglichkeiten der Vermittlungsschicht ein den Anforderungen der Anwendungen genügendes Protokoll realisieren. So ist beispielsweise für den Dateitransfer eine zuverlässige Verbindung über das unzuverlässige Best-Effort-IP erforderlich.

Allerdings stellen verschiedene Anwendungen unterschiedliche Anforderungen. So benötigt FTP eine zuverlässige Verbindung zur effizienten Übertragung großer Datenmengen. Ein Terminalprogramm liefert demgegenüber nur kleine Datenmengen (ein Zeichen für einen Tastendruck), die aber möglichst sofort geschickt werden sollen. Einige Anwendungen wie eine Video-Konferenz können gelegentliche Datenverluste akzeptieren, benötigen aber eine große Bandbreite mit einer über die Zeit relativ konstanten Qualität.

Erst auf der Ebene der Ende-zu-Ende-Protokolle können derartige Anforderungen konkretisiert werden. Durch diese Arbeitsteilung wird das Protokoll der Vermittlungsebene IP entlastet. IP stellt nur die notwendige Basisfunktionalität zur Verfügung und kann damit als Grundlage für unterschiedlichste Anwendungen dienen. Würde umgekehrt IP beispielsweise einen Schutz gegen Paketverluste gewährleisten, müssten verloren gegangene Pakete erneut geschickt werden. Durch die daraus resultierende Verzögerung wäre das Protokoll für Echtzeitanwendungen weniger geeignet.

Im Folgenden behandeln wir die Transportprotokolle UDP und TCP, die die Prototypen Datagramm-Dienst und stromorientierten Dienst über IP darstellen. Im Kapitel 10 wird ein Beispiel für einen Anfrage/Antwort-Dienst vorgestellt.

7.1 UDP

Das UDP – User Datagram Protocol – ist ein einfaches Datagramm-Protokoll. Im Prinzip erweitert es lediglich den Host-zu-Host-Dienst um die Adressierung einer Anwendung. Auf einem Rechner können gleichzeitig mehrere Anwendungen oder mehrere Instanzen einer Anwendung aktiv sein. Daher wird eine eindeutige Kennung für jede Anwendung benötigt.

Das Betriebssystem stellt zu diesem Zweck Kommunikationspunkte – die Ports – zur Verfügung. Eine Anwendung meldet sich an einem Port an und erhält von

nun an vom Betriebssystem alle Nachrichten für diesen Port. Dieser ist eine Abstraktion eines Kommunikationspunktes. Wie der Port konkret realisiert wird, bleibt dem Betriebssystem überlassen. Wesentlich ist, dass die UDP-Pakete die entsprechende Portadresse enthalten. Für diese Adressen stehen im UDP-Header 16 Bit zur Verfügung, wobei der Header sowohl Quell- als auch Zielport enthält. Zur Vollständigkeit ist der Aufbau der UDP-Pakete in Tabelle 7.1 dargestellt. Neben den Portadressen enthält der Header nur noch Felder für eine optionale Prüfsumme sowie die Längeninformation.

Ein UDP-Paket wird dann für die Übermittlung in ein IP-Paket gepackt. Dieses trägt die IP-Adresse des Zielrechners. Dort angekommen, wird das IP-Paket entpackt, und die Portadressen werden dem UDP-Header entnommen.

Insgesamt bildet das Paar <IP-Adresse, Portadresse> die vollständige Adresse der Anwendung. Das IP-Paket enthält sowohl für Sender als auch Empfänger diese beiden Informationen.

Tabelle 7.1: UDP Paket-Format		
0	16	31
Quellport	Zielport	
Prüfsumme	Länge	
Daten		

Jeder der beiden Prozesse muss in irgendeiner Form die Portadressen des jeweiligen Partners kennen. Einige Dienste verwenden daher feste, öffentlich bekannte Portnummern. In Tabelle 7.2 sind für einige Anwendungen die Portnummern aufgelistet. Eine umfangreiche Liste finden Sie auf der Online-Seite www.iana.org/assignments/port-numbers. In dem tecCHANNEL-Artikel „Ports im Überblick“ (**webcode: p901**) sind die wichtigsten Portnummern und -funktionen enthalten. In manchen Fällen benutzen Anwendungen derartige wohl bekannte Portnummern nur, um den Kontakt aufzunehmen und für die weitere Kommunikation einen eigenen Port auszuhandeln.

Tabelle 7.2: Reservierte Portnummern	
FTP-Daten	20
FTP-Steuerung	21
Telnet	23
DNS	53
WWW	80

7.2 TCP

UDP ist letztlich nur ein Multiplexer zwischen verschiedenen Anwendungen und dem IP-Protokoll. Es bietet keine funktionalen Erweiterungen gegenüber dem unzuverlässigen Paketdienst. Viele Anwendungen benötigen aber eine zuverlässige Verbindung. Das wichtigste Protokoll für diese Fälle ist TCP (Transmission Control Protocol). TCP realisiert einen zuverlässigen, verbindungsorientierten Byte-Strom. Anwendungen, die auf TCP aufsetzen, werden dadurch von vielen Implementierungsdetails entlastet. Wesentliche Eigenschaften von TCP sind:

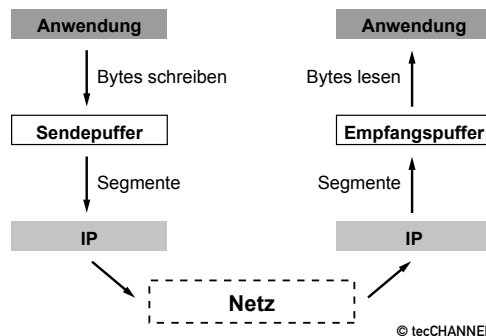
- Verbindungsaufbau
- zuverlässige Übertragung
- Vollduplex-Byte-Strom
- Flusskontrolle (Schutz des Empfängers vor zu vielen Daten)
- Überlastkontrolle (Schutz des Netzes vor zu vielen Daten)
- Multiplex für mehrere Anwendungen (analog zu UDP)

Die große Herausforderung für TCP ist, über einen unzuverlässigen Paketdienst und ein komplexes, sich ständig veränderndes Netz eine zuverlässige Verbindung mit möglichst hohem Datendurchsatz und geringer Verzögerung zu realisieren. TCP benutzt den uns aus Kapitel 3 bekannten Sliding-Window-Algorithmus. Der Basisalgorithmus wird allerdings an mehreren Punkten erweitert, um der wesentlich schwierigeren Situation Rechnung zu tragen.

7.2.1 Segmentierung

Aus Sicht der Anwendung stellt TCP einen Byte-Strom dar. Das heißt, die Anwendung kann einzelne Bytes schreiben oder lesen. Andererseits überträgt IP Pakete. Es ist wenig effizient, jedes Byte stets in einem eigenen IP-Paket zu verschicken. Das Verhältnis Nutzdaten zu Header-Daten ist dann sehr ungünstig. Daher setzt TCP den Byte-Strom in eine Folge von so genannten Segmenten um. Den gesamten Ablauf für eine Richtung zeigt Bild 7.1

Bild 7.1: Segmentierung des Byte-Stroms in TCP für eine Richtung.



Auf der Sendeseite sammelt TCP zunächst die geschriebenen Bytes in einem Puffer. Aus dem Puffer werden dann Bytes zu Segmenten zusammengefasst. Im Sinne der Netzwerknutzung sind möglichst große Segmente wünschenswert, um ein gutes Verhältnis von Nutzdaten zu Header-Daten zu erzielen.

Andererseits führen große Segmente zu längeren Wartezeiten. Außerdem gibt es im zu Grunde liegenden Netz eine Beschränkung der Paketgröße. Insgesamt verwendet TCP folgende Kriterien für das Versenden des nächsten Segments:

- Die maximale Segmentgröße (Maximum Segment Size, MSS) ist erreicht. Zweckmäßigerweise wird MSS so gewählt, dass IP das Segment nicht fragmentieren muss.
- Die Anwendung wünscht explizit den sofortigen Versand (Push-Operation). Beispiel: Terminalanwendungen wie Telnet erfordern die wenig effiziente, sofortige Weitergabe jedes Tastendrucks.
- Der Timer für die Zeit zum Aufsammeln ist abgelaufen.

Das Format der TCP-Segmente zeigt Tabelle 7.3. Die Information über die Segmentierung steckt im Feld Sequence Number. Diese Zahl bezeichnet den Index des ersten Bytes im Datenfeld. Der Header enthält keine Information zu der Länge des Datenfelds. Die Anzahl der Daten berechnet der Empfänger aus der Gesamtlänge des IP-Pakets abzüglich der beiden Header.

Tabelle 7.3: TCP Segment-Format				
0	4	10	16	31
Source Port		Destination Port		
Sequence Number				
Acknowledgement Number				
HdrLen	0	Flags	Advertised Window	
Checksum Urgent Pointer				
Optionen + Pad				
Daten				

Wie bei UDP enthält der Header die beiden Portadressen, die Länge des Headers sowie eine Prüfsumme. In dem Feld Flags stehen Steuerinformationen. Hier kann beispielsweise das Flag URG gesetzt werden, um den ersten Teil der Daten als dringend zu kennzeichnen.

Der Zeiger Urgent Pointer deutet dann auf das erste Byte, das nicht mehr zu dem dringenden Teil gehört. Die Bedeutung der anderen Felder werden wir im weiteren Verlauf des Kapitels behandeln.

7.2.2 Verbindungsaufbau

TCP ist ein verbindungsorientierter Dienst. Die Kommunikation beginnt mit einer Phase zum Aufbau einer virtuellen Verbindung. Diese existiert nur auf der Ebene von TCP und nicht im physikalischen Netz. Sie geht von einer Anwendung auf einem Rechner zu einer anderen Anwendung auf einem zweiten Rechner. Die beiden Endpunkte sind jeweils durch Portadresse und IP-Adresse gegeben. Insgesamt ist die Verbindung damit durch die vier Werte <IP-Adresse1, Portadresse1, IP-Adresse2, Portadresse2> definiert.

Eine wesentliche Aufgabe der beiden Partner ist es, sich über die Segmentnummern zu einigen. Die Designer von TCP hatten entschieden, dass für eine neue Verbindung zufällige Startwerte zu wählen sind. Damit soll verhindert werden, dass eventuelle Irrläufer aus einer vorherigen Verbindung der beiden Anwendungen die neue Verbindung stören. Durch die zufälligen Startwerte lassen sich diese alten Pakete mit großer Wahrscheinlichkeit an ihren nicht mehr passenden Segmentnummern erkennen. Zum Verbindungsaufbau dient ein Drei-Wege-Handshake. Dabei werden die Flags im Header zum Informationsaustausch genutzt. Der Ablauf ist wie folgt:

1. Anwendung A schickt ein Segment mit dem Flag SYN und seiner Segmentnummer NA.
2. Anwendung B bestätigt den Eingang durch ein Segment mit den Flags SYN und ACK. Gleichzeitig setzt B die Acknowledge Number auf NA+1, um anzugeben, welches Byte als Nächstes zu erwarten ist. Schließlich wird die eigene Segmentnummer NB zufällig ausgewählt und eingetragen.
3. Anwendung A erhält die Bestätigung und schickt daraufhin ebenfalls eine Bestätigung mit ACK und NB+1 als Acknowledge Number.

7.2.3 Sliding Window

Für eine zuverlässige Verbindung verlangt der Sender für jedes geschickte Segment eine Bestätigung. Diese kann in einem eigenen Segment stehen oder Teil eines Segments mit Daten sein. Wir haben bereits in der Diskussion zum einfachen Stop-and-Wait-Algorithmus gesehen, dass das Netzwerk schlecht ausgelastet ist, wenn erst nach Erhalt der Bestätigung das nächste Paket auf die Reise geht. Diese Situation ist bei einer Verbindung im Internet noch verschärft. Die Antwortzeiten sind hier recht groß, insbesondere kann ein Paket sich zum Beispiel durch Überlastung einer Teilstrecke deutlich verzögern. Daher muss der Sender beim Ausbleiben einer Bestätigung lange warten, bis er sicher ist, dass das Segment verloren gegangen ist.

TCP nutzt daher den Sliding-Window-Algorithmus, und ein Sender schickt mehrere Segmente direkt nacheinander. Anhand der eingehenden Bestätigungen kann er verfolgen, welche Segmente angekommen sind. Bleibt eine Bestätigung aus, wiederholt er die alten Segmente ab dem letzten bestätigten Segment.

Schätzung der RTT

Im Internet mit den unterschiedlichsten Arten von Verbindungen, die noch dazu zeitlichen Schwankungen unterworfen sind, ist die Wahl der Wartezeit auf Bestätigungen schwierig. Ein fester Wert für den Timeout müsste an Verbindungen mit großer RTT ausgerichtet sein und wäre damit in vielen Fällen zu pessimistisch.

Daher wird bei TCP die Laufzeit adaptiv geschätzt. Der Sender misst dazu die Zeit vom Versand bis zum Erhalt der Bestätigung und benutzt diesen Wert als aktuelle Schätzung RTT_a . Diese aktuelle Schätzung kann sehr stark schwanken. Um diese Schwankungen auszugleichen, wird der Schätzwert für RTT als gleitender Mittelwert zwischen dem alten Wert und dem aktuellen Wert in der Form

$$RTT = \alpha \times RTT + (1 - \alpha) \times RTT_a$$

gemittelt. Der Koeffizient α bestimmt, wie schnell sich die Schätzung an Änderungen adaptiert. Bei einem kleinen Wert von α geht der alte Wert nur zu einem geringen Anteil in die Berechnung ein, so dass sich Änderungen sehr schnell auswirken. Umgekehrt bewirkt ein großer Wert, das heißt α nahe an 1, dass die Änderungen langsam erfolgen.

Typische Werte liegen im Bereich von 0,8 bis 0,9. TCP benutzt dann als Wert für den Timeout das Doppelte der geschätzten RTT. Der beschriebene Algorithmus stammt aus der ursprünglichen TCP-Spezifikation. Mittlerweile sind Verfeinerungen des Algorithmus im Einsatz, die unter anderem die Schwankung der Messwerte als Maß für die Zuverlässigkeit der Schätzung benutzen.

Flusskontrolle

Auf Seiten des Empfängers besteht die Gefahr, dass die ankommenden Daten nicht schnell genug von der Anwendung aus dem Empfangspuffer gelesen werden. Dann füllt sich der Puffer immer mehr, und irgendwann ist kein Platz mehr für weitere Daten. Der Empfänger kann den Empfang neuer Segmente nicht mehr bestätigen, und der Sender muss sie später nochmals schicken. Ohne besondere Vorkehrungen wäre in einem solchen Fall viel unnötiger Netzverkehr die Folge. Daher verfügt TCP über einen Mechanismus, mit dem der Empfänger die Übertragungsgeschwindigkeit des Senders steuern kann.

Der Empfänger benutzt dazu das Feld Advertised Window im TCP-Header. In seinen Bestätigungsmeldungen (oder eigenen Datensegmenten) trägt er in diesem Feld ein, wie viele Bytes er zurzeit noch aufnehmen kann. Der Sender schickt dann bis zur nächsten Bestätigung nur noch maximal so viele Bytes. Wenn beim Empfänger der Platz abnimmt, weil die lokale Anwendung die Daten momentan nur sehr langsam oder gar nicht mehr liest, reduziert er die Größe im Feld Advertised Window und bremst damit den Sender.

Im Extremfall – wenn der Puffer vollständig gefüllt ist – meldet er den Wert 0 und signalisiert damit, dass er zurzeit keine Daten mehr aufnehmen kann. In diesem Fall schickt der Sender trotzdem weiterhin in regelmäßigen Abständen Segmente mit nur einem Daten-Byte. Solange der Empfänger nicht in der Lage ist, Daten

aufzunehmen, ignoriert er die Segmente. Hat er allerdings wieder Platz, kann er das Byte übernehmen und in der Bestätigung ein größeres Advertised Window angeben. Auf diese Art wird der Transfer wieder aufgenommen, ohne dass beim Empfänger eine spezielle Vorgehensweise implementiert sein muss. Dieses Konzept, bei dem die Intelligenz beim Sender liegt, wird mit dem allgemeinen Ausdruck *smart sender/ dumb receiver rule* bezeichnet.

7.2.4 Überlastkontrolle

Ohne weitere Maßnahmen kann TCP leicht zu einer Überlastung des Netzes führen. Dass TCP im Fall der Netzüberlastung, wenn Segmente verloren gehen, mit erhöhter Aktivität reagiert, indem es die Segmente erneut sendet, ist dann besonders kritisch. Damit verschlimmert sich eine Überlast im Netz noch weiter, und es kann zu einem Kollaps kommen. Um die Situation zu verbessern, wurde Ende der achtziger Jahre eine Überlastkontrolle in TCP eingeführt. Sie basiert auf der Schätzung der Netzbelastung durch einen Sender. Auf der Basis der beobachteten Paketübertragungen und Paketverluste steuert ein Sender selbstständig seine Übertragungsgeschwindigkeit.

Ein prinzipielles Problem dabei ist, dass er nur indirekt Informationen über das Netz erhält. Um das Optimum zu ermitteln, muss er die Grenzen herausfinden. Diese entdeckt er nur, indem er sie überschreitet. Er muss zunächst die Übertragungsrate möglichst weit erhöhen, bis er irgendwann an ein Limit stößt. Dann nimmt er die Rate zurück und nähert sich sozusagen vorsichtig von unten wieder dem Optimum. Durch diese gezielte Überschreitung des Optimums wird zusätzlich Verkehr erzeugt. Daher ist es wichtig, nach einer Überschreitung schnell wieder auf einen sicheren Wert zurückzugehen. Umgekehrt kann das Herantasten an das Optimum langsam erfolgen.

Auf diesen Gedanken beruht das Konzept *Additive Increase / Multiplicative Decrease*. Der Sender führt ein Überlastfenster, das festlegt, wie viele Daten maximal bis zur nächsten Bestätigung gesendet werden dürfen. Diese Größe schöpft er nur aus, wenn der Empfänger entsprechende Aufnahmekapazität signalisiert hat. Geht ein Segment verloren, vermutet der Sender eine Überlastung im Netz. In diesem Fall reduziert er sein Überlastfenster auf die Hälfte.

Umgekehrt zeigt ihm ein ACK an, dass ein Segment gut angekommen ist. Wenn sämtliche Segmente in einem Sliding-Window-Intervall bestätigt wurden, erhöht er das Überlastfenster um einen kleinen Betrag. Das Fenster wächst langsam an, bis irgendwann erneut das Limit erreicht ist. Dann wird es wieder auf die halbe Größe reduziert. Dieser Zyklus des langsamen Anwachsens und schnellen Schrumpfens wiederholt sich immer wieder. Die Priorität liegt auf dem Schutz des Netzes vor Überlast.

Das typische Verhalten des Überlastfensters zeigt Bild 7.2. Die Werte wurden mittels eines einfachen Simulationsprogramms erzeugt. In der Simulation wird eine Kapazität vorgegeben. Das Überlastfenster wächst bis zum Erreichen der Kapazi-

tätsgrenze linear an. Nach jedem Überschreiten wird das Fenster wieder auf die Hälfte des letzten Wertes reduziert. Der Wert für die Kapazitätsgrenze – im Bild jeweils als Punkt eingetragen – wird in der Simulation nach jedem Zeitschritt um einen zufälligen Wert verändert.

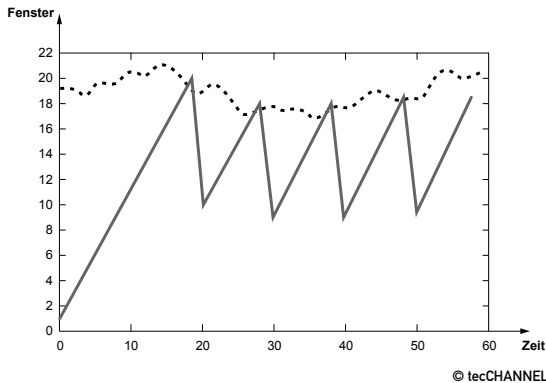


Bild 7.2: Regelung der Fenstergröße durch Lastkontrolle bei simulierter Kapazitätsgrenze (gestrichelte Linie).

Eine besondere Situation besteht am Anfang der Verbindung. Hier hat der Sender noch keinerlei Informationen über das Netz. Deshalb befindet er sich in einem Dilemma: Beginnt er mit der vollen Fenstergröße, riskiert er eine Überlastung. Andererseits ist ein langsames, lineares Steigern unter Umständen zu pessimistisch. Daher verwendet der Sender einen Slow-Start-Ansatz (im Gegensatz zu Full-Start mit der vollen Fenstergröße). In diesem Fall erhöht er das Fenster bereits nach jedem ACK. Wenn er vor dem Timeout n Segmente geschickt hat, wird im Erfolgsfall das Fenster auch n mal erhöht und nicht nur einmal wie im oben beschriebenen Additive-Increase-Ansatz. In Summe verdoppelt sich dadurch die Anzahl der gesendeten Segmente pro RTT. Das Verfahren ist sozusagen das Gegenstück zum Multiplicative Decrease mit umgekehrter Tendenz.

Prof. Dr. Stephan Euler

tecCHANNEL-Links zum Thema	Webcode	Compact
UDP und TCP	a1497	S.73
Ports im Überblick	p901	–
So funktionieren TCP/IP und IPv6	p209	–
Ethernet im Überblick	a717	–
Grundlagen Netzwerkkomponenten	a1092	–
Grundlagen: Layer-4/7-Switching	p1370	–
Ethernet im Überblick	a717	–

8. ATM-Netzwerktechnologie

ATM ist aus der Konvergenz von Telefonie und digitaler Datenübertragung entstanden. Die verbindungsorientierte Netzwerktechnik kommt heute vor allem im WAN zum Einsatz und bietet dort neben QOS zahlreiche weitere Vorteile.

In den späten 80er Jahren begannen die Telefongesellschaften mit der Digitalisierung ihrer Netze. Ziele waren zum einen die Verbesserung des Telefondienstes durch eine bessere Qualität und ein höherer Komfort mit neuen Leistungsmerkmalen. Zum anderen wollten die TK-Anbieter auf Grund des absehbaren Zusammenwachsens von klassischem Telefondienst und neuen Datenanwendungen auf eine einheitliche Netzstruktur umstellen. Weiterhin sollte die neue Telefontechnik kompatibel zu bestehenden Installationen sein und ohne neue Verkabelung auskommen. Diese Zielsetzung führte zunächst zur Entwicklung des ISDN-Netzes (Integrated Services Digital Network).

Das ISDN-Netz besteht aus Daten- und Signalisierungskanälen. Jeder Datenkanal (B-Kanal) hat eine Bandbreite von 64 Kbit/s. Diese Bandbreite ist für die Übertragung von Sprache in guter Telefonqualität ausgelegt. Sie berechnet sich aus einer Abtastrate von acht kHz und einer Kodierung der Abtastwerte mit acht Bit. Die Signalisierung etwa für den Verbindungsauf- und -abbau erfolgt über einen getrennten Kanal (D-Kanal). Dieser Kanal kann auch zur Übertragung von Daten benutzt werden. Zur Auswahl stehen dem Kunden ein Basisanschluss (zwei B-Kanäle und ein D-Kanal mit 16 Kbit/s) oder ein Primär-Multiplexanschluss (30 B-Kanäle und ein D-Kanal mit 64 Kbit/s).

Als Erweiterung oder Ergänzung für Anwendungen mit höherem Bedarf an Bandbreite konzipierten die TK-Gesellschaften B-ISDN (Broadband-ISDN). Es sollte sowohl den ansteigenden Datenverkehr aufnehmen als auch eine Grundlage für neuartige Dienste, wie hochwertiges Bildtelefon oder „Video-on-Demand“, darstellen. Für einen solchen allgemeinen Einsatz ist eine Einteilung in feste Kanäle nicht mehr effizient. Daher wurde als Netztechnologie ATM – Asynchronous Transfer Mode – ausgewählt. B-ISDN hat es allerdings nie zur Marktreife gebracht. Auf Grund seiner guten Eigenschaften bei hohen Übertragungsraten ist ATM auch für andere Einsatzgebiete attraktiv. Daher kommt es heute vor allem im WAN-Bereich oder zur Verbindung mehrerer LANs zum Einsatz.

Das folgende Kapitel gibt eine Einführung in die Prinzipien und Anwendung von ATM. Der Schwerpunkt liegt auf den Aspekten der Netzwerkkommunikation. Eine ausführliche Darstellung unter stärkerer Betonung der Telekommunikationsanwendungen findet sich beispielsweise in [12] und [13]. Aktuelle Informationen bietet die Internet-Seite des ATM-Forums (<http://www.atmforum.com/>).

Das ATM-Forum wurde 1991 zur Unterstützung der ATM-Entwicklung und zur Förderung der ATM-Verbreitung gegründet. Mittlerweile sind etwa 150 Firmen Mitglieder dieser Organisation. Ziel ist die Weiterentwicklung der ATM-Spezifikationen, Abstimmung mit anderen Standardisierungsgremien sowie der Informationsaustausch zwischen Anbietern und Nutzern von ATM-Geräten.

8.1 ATM-Grundlagen

Die Anforderungen an die Technologie für B-ISDN waren vielfältig. Sie sollte eine einheitliche Plattform sowohl für Anwendungen aus dem Telefonbereich als auch für die Datenkommunikation zwischen Rechnern bieten. Gleichzeitig sollte die Technologie kostengünstig in der Realisierung und skalierbar für wachsende Bandbreiten sein. Wie in Kapitel 2 diskutiert, sind entsprechend den unterschiedlichen Anforderungen der Anwendungen zwei Vermittlungsverfahren üblich: die Leitungsvermittlung und die Paketvermittlung. ATM kann man als einen Mittelweg zwischen beiden Techniken sehen.

Der Datenverkehr von ATM beruht auf Daten, wobei aber feste Verbindungen zwischen den Teilnehmern bestehen. Eine Besonderheit ist, dass alle Pakete die gleiche konstante Länge haben. In der Terminologie von ATM werden diese Pakete als Zellen bezeichnet. Die Verbindung zwischen zwei Knoten besteht aus einem konstanten Strom von derartigen Zellen, die lückenlos aufeinander folgen. Stehen zu einem Zeitpunkt keine Daten zum Versand an, so wird eine spezielle Leerzelle in den Strom eingefügt.

Die feste Länge der Zellen verkürzt nicht unbedingt die Verarbeitungszeit pro Zelle in einem Vermittlungsknoten. Der große Gewinn gegenüber der allgemeinen Paketvermittlung liegt in der Übersichtlichkeit und Berechenbarkeit. Man kann die Situation an den Eingängen eines Switches gut mit der Bedienung an Supermarktkassen oder Flughafenschaltern vergleichen. Wenn die Zeitdauer der Abfertigung eines Kunden nicht genau vorhergesagt werden kann, ist die Auswahl der „schnellsten“ Schlange schwierig. Es kann passieren, dass die gewählte Schlange sehr langsam abgearbeitet wird. Kunden, die sich später in einer anderen Schlange angestellt haben, kommen in diesem Fall schneller voran. Würde die Abfertigung eines jeden Kunden exakt gleich lange dauern, wäre die Auswahl der Warteschlange einfach und die Wartezeit damit vorab berechenbar.

Durch die Berechenbarkeit lassen sich auch leichter Aussagen zur Qualität einer Verbindung treffen (Dienstgüte, Quality of Service, QOS). Ein Switch kann Anforderungen wie eine Mindestbandbreite oder eine Obergrenze für die Verzögerung garantieren, indem er eine entsprechende Anzahl von Zellen pro Zeiteinheit für die Verbindung reserviert. Wie der Name ATM besagt, sind diese Reservierungen asynchron. Die Lage der Zellen einer Verbindung in dem Strom ist nicht festgelegt. Demgegenüber wären bei einer synchronen Verbindung die Zellen für die verschiedenen Verbindungen in einem festen Raster angeordnet.

Die Vereinfachung der Vermittlung ermöglicht eine kostengünstige Realisierung von Vermittlungsgeräten. Insbesondere erleichtert die feste Zellengröße die Parallelisierung von Aufgaben. Die Weiterleitung einer Zelle ist eine klar definierte Aufgabe, für die ein Entwickler ohne viel Aufwand eine autonome Verarbeitungseinheit entwerfen kann. Durch die damit ermöglichte Parallelisierung eignen sich ATM-Switches besonders für einen großen Datendurchsatz.

ATM ist verbindungsorientiert. Die Kommunikation zwischen zwei Knoten erfolgt als virtuelle Verbindung, wie in Kapitel 5 beschrieben. Zu Beginn der Kommunikation wird ein Weg als virtuelle Verbindung aufgebaut. In dieser Signalisierungsphase können die beteiligten Komponenten auch Verkehrsparameter für die Dienstgüte aushandeln.

Die virtuellen Verbindungen sind dabei in zwei Hierarchien organisiert. Ein virtueller Kanal (Virtual Channel, VC) stellt eine unidirektionale Verbindung bereit. Mehrere virtuelle Kanäle zwischen zwei Knoten bündelt ein so genannter virtueller Pfad (Virtual Path, VP). So können für eine Multimedia-Verbindung zwischen zwei Teilnehmern getrennte Kanäle für Audio- und Video-Signale in einem gemeinsamen Pfad liegen. Diese zweistufige Organisation erleichtert die Aufgabe der Switches, da diese nur die Kennung für den Pfad auswerten müssen.

8.2 ATM-Zellengrößen

Nachdem die Vorteile einer festen Zellengröße einsichtig sind, stellt sich die Frage nach der optimalen Zellengröße. Kleine Zellen haben den Vorteil geringer Verzögerungszeit. Dafür ist bei vorgegebener Größe des Headers das Verhältnis zwischen Nutzdaten und Zellengröße schlechter als bei großen Zellen. Große Zellen sind jedoch bei kurzen Nachrichten, wie etwa Bestätigungsmeldungen, nur zu einem geringen Teil mit Nutzdaten gefüllt.

Diese Überlegungen zeigen, dass es keine optimale Zellengröße für alle Fälle gibt. Vielmehr erweisen sich bei unterschiedlichen Anwendungen auch unterschiedliche Zellengrößen als jeweils optimal. Nicht umsonst wird bei der Datenübertragung in Rechnernetzen mit ihren vielfältigen Anwendungen eine variable Länge der Pakete bevorzugt.

Bei der Festlegung der Größe der ATM-Zellen orientierte man sich an der damals wichtigsten Anwendung, der Übertragung von Telefongesprächen. Die Kunden erwarten beim Telefonieren zumindest die vom Analognetz gewohnte Qualität. Neben dem Frequenzgang und dem Rauschanteil ist die Verzögerungszeit dabei ein entscheidender Faktor. Um die konkrete Festlegung für die Paketlänge zu verstehen, muss man jedoch etwas weiter ausholen.

Die Sprachsignale eines Teilnehmers werden vom Telefon des Gesprächspartners zum Teil reflektiert. Der Sprecher nimmt seine Worte deshalb als zeitverzögertes Echo wahr. Dieser Effekt entsteht unter anderem dadurch, dass das Mikrofon in einem Telefonhörer auch die Signale des Lautsprechers leise mit aufnimmt und zurücksendet. Diese Echos stören normalerweise nicht. Sie tragen im Gegenteil sogar zu einem natürlichen, räumlichen Gesprächseindruck bei.

Mit steigender Laufzeit werden diese Echos allerdings zunehmend als störend wahrgenommen und beeinträchtigen ab 150 ms das Gespräch. Daher enthalten Verbindungen mit langer Laufzeit, etwa bei der Übertragung via Satellit (Latenzzeit >200 ms), in der Regel spezielle Echokompensatoren.

In den Standardisierungsgremien von ATM machten die Vertreter aus Amerika und Europa unterschiedliche Vorschläge zur Länge der Zellen. In den USA ist auf Grund der geographischen Ausdehnung der Einsatz von Echokompensatoren ohnehin erforderlich. Daher plädierten die amerikanischen Vertreter für eine Länge von 64 Daten-Bytes. Die meisten europäischen Länder setzen demgegenüber keine Echokompensatoren ein. Bei dem europäischen Vorschlag von 32 Daten-Bytes wäre die Verzögerung innerhalb der akzeptablen Zeit geblieben. Zumindest bei der Kommunikation im eigenen Landesnetz wollte man so die damals teuren Kompensatoren einsparen. Die Gremien konnten sich jedoch nicht auf einen dieser beiden Vorschläge einigen und wählten als Kompromiss den mittleren Wert von 48 Daten-Bytes.

8.3 ATM-Zellen-Header

Zu dem Feld von 48 Bytes für Nutzdaten kommt ein Header mit fünf Bytes, so dass eine ATM-Zelle insgesamt 53 Bytes lang ist. Prinzipbedingt gibt es bei ATM zwei Arten von Schnittstellen:

- Network Node Interface (NNI): bezeichnet die Schnittstelle zwischen zwei ATM-Switches.
- User Node Interface (UNI): bezeichnet die Schnittstelle zwischen einem ATM-Switch und einem Endpunkt.

Je nach Schnittstelle unterscheiden sich die Felder im Header geringfügig. Folgende Tabelle zeigt den Header für ein NNI.

Tabelle 8.1: ATM-Header für ein Netzwerk-Interface				
12 Bit	16 Bit	3 Bit	1 Bit	8 Bit
VPI	VCI	PTI	CLP	HEC

Die beiden ersten Felder enthalten die Kennung für Pfad und Kanal (Virtual Path Identifier, Virtual Channel Identifier). Das PTI-Feld (Payload Type Identifier) spezifiziert die Art der Zelle. So kann zwischen Zellen mit Benutzerdaten und Zellen zur Steuerung unterschieden werden. Mit dem CLP-Bit (Cell Loss Priority) können Zellen mit niedriger Priorität markiert werden. Kommt es in einem Switch zu einem Überlauf, so werden zuerst Zellen mit gesetztem CLP-Bit, also Zellen mit niedriger Priorität, verworfen.

In dem Feld HEC (Header Error Control) schließlich steht die CRC-Prüfsumme über die ersten vier Bytes. Damit kann ein Switch Zellen mit fehlerhaftem Header erkennen. Die Prüfsumme dient zusätzlich einem ganz anderen Zweck, nämlich der Synchronisation auf die Zellengrenzen. Da die ATM-Zellen keine speziellen Anfangs- oder Endmarkierungen enthalten, sieht ein Empfänger zunächst nur ei-

nen stetigen Bitstrom. Um die Grenzen der einkommenden Zellen zu lokalisieren, entnimmt der Adressat zufällig Gruppen von fünf aufeinander folgenden Bytes.

Jede dieser Gruppen testet er bezüglich einer korrekten Prüfsumme. Falls die Prüfung zu einem korrekten Ergebnis führt, handelt es sich mit großer Wahrscheinlichkeit um einen Header. Diese Hypothese kann der Empfänger durch Untersuchung des nächsten potenziellen Headers nach 53 Bytes untermauern. Wenn er eine vorgegebene Anzahl von aufeinander folgenden Headern korrekt erkannt hat, kann er mit hoher Wahrscheinlichkeit davon ausgehen, dass er die Synchronisation der Zellen ermittelt hat. Umgekehrt erkennt er den Verlust der Synchronisation, sobald er einen Schwellwert ungültiger Header in Folge ermittelt. In diesem Fall beginnt er an einer beliebigen Stelle im Bitstrom und sucht mit dem beschriebenen Verfahren nach gültigen Headern.

8.4 ATM-Anpassungsschicht

ATM bietet einen Transport von Zellen fester Größe. Es gibt jedoch keine Garantie für die Korrektheit der übermittelten Nutzdaten. So kann es durchaus vorkommen, dass komplette Zellen unterwegs verloren gehen. Allerdings ist gewährleistet, dass die Zellen stets in der korrekten Reihenfolge ankommen. Anders als bei einer reinen Paketvermittlung kann auf der virtuellen Verbindung keine Zelle eine andere überholen.

Zwischen dem Transportdienst mit Zellen und den höheren Schichten ist eine Anpassung notwendig. Eine Anpassungsschicht (ATM Adaptation Layer, AAL) übernimmt diese Umsetzung von Daten der höheren Netzwerkschichten auf die ATM-Schicht.

ATM soll verschiedenartigste Dienste mit unterschiedlichen Anforderungen transportieren. Die Umsetzung kann daher nicht von einer einzigen Schnittstelle gewährleistet werden. Stattdessen existieren mehrere AAL-Typen, die jeweils für eine bestimmte Art von Dienst ausgelegt sind. Im Detail unterscheidet man bei dem AAL zwischen zwei Subschichten:

- Convergence Sublayer (CS): Anpassung an die höheren Schichten (Konvergenzschicht).
- Segmentation and Reassembly (SAR): Aufteilung größerer Dateneinheiten auf mehrere Zellen beim Sender (Segmentation) und Rekonstruktion (Reassembly) beim Empfänger.

Zu den Aufgaben der Konvergenzschicht gehören die Erkennung von Zellenverlusten und die Synchronisation zwischen Sender und Empfänger. Dazu benötigte zusätzliche Informationen werden in den Datenteil der ATM-Zellen eingetragen. Man kann sich dies als Minizelle, eingebettet in die ATM-Zelle, vorstellen. Es sei betont, dass dieser Prozess nur an den jeweiligen Endpunkten einer Verbindung notwendig ist. Das ATM-Netz selbst transportiert die Zellen unabhängig von der Bedeutung der Werte im Datenteil.

Ursprünglich waren vier verschiedene AAL-Typen vorgesehen:

- AAL 1: Verbindungsorientierte Echtzeitdienste mit fester Bandbreite (zum Beispiel Telefongespräche)
- AAL 2: Verbindungsorientierte Echtzeitdienste mit variabler Bandbreite. Hierunter fallen beispielsweise komprimierte Video-Signale, bei denen der Kompressionsgrad, und damit die Datenrate, nicht konstant sind, sondern von Bildinhalt und Bildähnlichkeit abhängen.
- AAL 3: Verbindungsorientierte Dienste für die Datenübertragung ohne Echtzeitanforderungen
- AAL 4: Verbindungslose Dienste für die Datenübertragung ohne Echtzeitanforderungen

Die Typen 3 und 4 waren zur Datenübertragung zwischen Rechnern (Dateitransfer) vorgesehen. Im Laufe der Zeit zeigte sich, dass beide Typen sehr ähnlich sind. Daher wurden sie zu einem gemeinsamen Typ AAL 3/4 verschmolzen. Das Protokoll beinhaltet auch die Möglichkeit zum Multiplexen mehrerer Anwendungen in einer Verbindung.

Aus Sicht der Computerindustrie war der Standard AAL 3/4 unnötig komplex. Daher entwickelte sie einen eigenen Vorschlag für die ATM-Datenübertragung mit dem bezeichnenden Namen Simple Efficient Adaptation Layer (SEAL). Im Prinzip werden dabei lediglich die Daten eines Pakets der höheren Schicht auf entsprechend viele ATM-Zellen aufgeteilt und um Fülldaten, die Längeninformation sowie einen Fehlerschutz mittels CRC-32 ergänzt. Diese Informationen stehen in der letzten Zelle des Pakets. Gekennzeichnet ist diese spezielle Abschlusszelle (End of Message, EOM) durch ein gesetztes drittes Bit im PTI-Feld des Headers. Diesen Vorschlag hat das ATM-Forum als AAL 5 übernommen.

8.4 Übergänge zwischen LAN und ATM

In großen Netzen (WAN) oder als schnelle Verbindung zwischen Routern hat ATM weite Verbreitung gefunden. Mit den beschriebenen Vorzügen und Datenraten von 25 Mbit/s bis 2,4 Gbit/s stellt es eine geeignete Transporttechnologie für diese Einsatzgebiete dar. Im Gegensatz dazu ist der erwartete Durchbruch von ATM-fähigen Endgeräten weit gehend ausgeblieben. Durch Fortschritte in der Technologie und dem vermehrten Einsatz von Punkt-zu-Punkt-Verkabelungen bietet Ethernet bei günstigeren Preisen einen ähnlich guten Datendurchsatz.

Daher ist weiterhin mit einer Koexistenz von LAN und ATM-Systemen zu rechnen. Eine Möglichkeit zur Verschmelzung ist die Nachbildung eines LANs durch ein ATM-System. In diesem Fall können alle vorhandenen LAN-Applikationen unverändert weiterverwendet werden. Ein großes Problem bei der LAN-Emulation durch ATM ist jedoch das Fehlen einer einfachen Möglichkeit für Broadcasts und Multicasts [14].

Insbesondere bei der Initialisierung sind Broadcasts ein wichtiges Mittel, um Informationen über andere Knoten zu erhalten. Ein Beispiel dafür ist das Adressauflösungsprotokoll ARP (siehe Kapitel 6). Dieses Problem löst das LANE- (LAN-Emulation-) Protokoll, das dafür einen speziellen Server voraussetzt.

Dieser so genannte BUS (Broadcast and Unknown Server) erhält Multicast-Verbindungen zu allen angeschlossenen Clients aufrecht. Ein Client schickt seine Multicast-Anfrage an den BUS, der dann die Verteilung übernimmt und die Antworten an den Client zurückgibt. Der gesamte Ablauf ist recht komplex. Er beinhaltet unter anderem auch die Umsetzung der MAC-Adressen in ATM-Adressen. Eine detaillierte Beschreibung findet sich beispielsweise in [14] (www.cse.ohio-state.edu/~jain/papers/ftp/mcast.pdf).

Stephan Euler

tecCHANNEL-Links zum Thema	Webcode	Compact
ATM-Netzwerktechnologie	a1498	S.81
Voice over IP	a990	–
IP-VPNs als Managed Service	p1426	–
Dienstgüte bei ATM und IP	a425	–

Mit den Webcodes gelangen Sie auf www.tecChannel.de direkt zum gewünschten Artikel. Geben Sie dazu den Code in das Feld „Webcode“ in der Titelleiste von www.tecChannel.de ein. Mit „p“ gekennzeichnete Webcodes beziehen sich auf Online-Artikel aus tecCHANNEL-Compact. Es ist möglich, dass Artikel zunächst exklusiv in tecCHANNEL-Compact veröffentlicht werden und daher erst später über den Webcode erreichbar sind.

9. Sockets

Eine elementare Schnittstelle zur Programmierung von Netzerkanwendungen bieten die so genannten Sockets (Englisch für Steckdose). Als API (Application Program Interface) stellen Sockets Funktionen zum Aufbau von Verbindungen sowie zur Kommunikation zwischen Applikationen bereit.

Sockets kapseln die Details der Netzerkommunikation. Der Programmierer kann sich so auf die Realisierung der Anwendung konzentrieren. Im Idealfall funktionieren die Anwendungen dadurch unabhängig von Programmiersprache, Betriebssystem und Netzwerk-Interface.

Das Socket API wird von allen gängigen Betriebssystemen unterstützt. Sockets setzen auf die Protokolle einer Transportschicht wie TCP oder UDP auf. Aus Sicht der Anwendung ist ein Socket ein Zugang zum Netzwerk. Im Folgenden diskutieren wir die Programmierung am Beispiel der Programmiersprache C. Die Verwendung in anderen Programmiersprachen erfolgt ähnlich. Am Ende dieses Kapitels zeigen wir eine Implementierung in der Sprache Perl. Einige Beispiele für die Programmiersprache Java finden sich in Kapitel 12.

Das Socket API stellt zwei Verfahren zur Verfügung: verbindungsorientierte und verbindungslose Kommunikation. Im ersten Fall müssen die beiden beteiligten Rechner zuerst eine Verbindung aufbauen, bevor sie Daten austauschen können. In beiden Fällen jedoch übernimmt einer der beiden Rechner – der Server – die Aufgabe, zunächst die Netzressource aufzubauen. Anschließend wartet er auf Verbindungswünsche des anderen Rechners – des Clients. Ein Server kann im Allgemeinen mehrere Clients bedienen. Sockets eignen sich nicht nur für die Kommunikation zwischen Rechnern. Sie können auch die Kommunikation zwischen mehreren Applikationen auf einem Rechner erledigen.

9.1 Verbindungsorientierte Kommunikation

Den prinzipiellen Ablauf des Aufbaus einer Verbindung und des anschließenden Datenaustausches zeigt die Tabelle auf der nächsten Seite. Für jeden Schritt ist der Name der entsprechenden Funktion in der Programmiersprache C angegeben.

Eine Besonderheit ist bei der Verwendung von Sockets unter Windows (Windows-Sockets) zu beachten: Um den Service benutzen zu können, muss zunächst die Funktion WSASStartup aufgerufen werden. Dabei kann der Programmierer die gewünschte Version einstellen. Dazu ein Beispiel:

```
long WinsockStartup()
{
    long rc;

    WORD wVersionRequested;
    WSADATA wsaData;
```

```

wVersionRequested = MAKEWORD(2, 1);

rc = WSASStartup( wVersionRequested, &wsaData );
return rc;
}

```

In diesem Beispiel wird die Version 2.1 spezifiziert. Nach dem Aufruf stehen in der Struktur `wsaData` Informationen über die gefundene Implementierung der Windows-Sockets.

Tabelle 9.1: Funktionen für verbindungsorientierte Sockets

SERVER		CLIENT	
Endpunkt anlegen	<code>socket()</code>		
Adresse festlegen	<code>bind()</code>		
Warteschlange starten	<code>listen()</code>		
Auf Anmeldung warten	<code>accept()</code>		
		Endpunkt anlegen	<code>socket()</code>
		Verbinden	<code>connect()</code>
Daten schreiben	<code>send()</code>	Daten lesen	<code>recv()</code>
Daten lesen	<code>recv()</code>	Daten schreiben	<code>send()</code>

9.1.1 Funktion „socket“

Zu Beginn muss man das zu verwendende Protokoll spezifizieren. Dies erfolgt mit der Funktion

```
SOCKET socket( int family, int type, int protocol );
```

Das erste Argument spezifiziert die Protokollfamilie (Adressfamilie). Mögliche Werte sind unter anderem:

- `AF_UNIX` Unix-internes Protokoll
- `AF_INET` Internet-Protokoll IPv4
- `AF_INET6` Internet-Protokoll IPv6
- `AF_APPLETALK` AppleTalk

Diese Werte sind als Konstanten in entsprechenden Header-Dateien definiert. Das Präfix `AF_` steht für Adressfamilie. Gleichwertig dazu existieren die Konstanten mit den gleichen Endungen und dem Präfix `PF_` für Protokollfamilie. Das zweite Argument `type` gibt die Verbindungsart an. Die wichtigsten Möglichkeiten sind:

- `SOCK_STREAM` Stromorientierte Verbindung
- `SOCK_DGRAM` Datagrammverbindung

Eine weitere Option ist `SOCK_RAW`. Damit werden beispielsweise im Programm Ping ICMP-Pakete über einen Socket versandt. Das Argument `protocol` wird in den meisten Fällen auf 0 (NULL) gesetzt. Dann wird automatisch aus den beiden anderen Argumenten das passende Protokoll ausgewählt. So resultiert aus der Kombination `AF_INET` mit `SOCK_STREAM` eine TCP-Verbindung. `AF_INET` mit `SOCK_DGRAM` definiert eine UDP-Verbindung. Ansonsten kann man auch explizit ein Protokoll auswählen (z.B. `IPPROTO_TCP`).

Bei Erfolg liefert die Funktion einen Bezeichner für einen Socket zurück. Eigentlich handelt es sich lediglich um einen Integer-Wert – sozusagen um den Index des Sockets. Zur besseren Lesbarkeit des Programms kann man den Datentyp `SOCKET` verwenden. Das folgende Codefragment zeigt beispielhaft einen Aufruf, um einen Socket für TCP zu erzeugen:

```
SOCKET sockListen;
sockListen=socket(AF_INET,SOCK_STREAM,NULL);
if (sockListen == INVALID_SOCKET) {
    printf("Error: Cannot create Socket\n");
}
```

9.1.2 Funktion „bind“

Im nächsten Schritt muss der Server dem Socket einen Namen geben, unter dem er später erreichbar sein wird. Dazu dient die Funktion `bind` in der Form

```
int bind( SOCKET s, sockaddr *p_addr, int addrlen );
```

Das erste Argument ist der soeben angelegte Socket. Zur Übergabe der spezifischen Details wird eine Struktur `SOCKADDR` verwendet. Das zweite Argument besteht in einem Zeiger auf die Struktur, das dritte gibt deren Größe an. Für TCP verwendet man die Struktur `SOCKADDR_IN` mit folgenden Elementen

```
struct sockaddr_in{
    short          sin_family;
    unsigned short sin_port;
    struct in_addr sin_addr;
    char          sin_zero[8];
};
```

Das erste Element bestimmt die Adressfamilie. Das dritte Element ist wiederum eine Struktur, die die IP-Adresse enthält. Für unsere Zwecke reicht es aus, in dieser Unterstruktur ein Element auf `INADDR_ANY` zu setzen, um ein beliebiges Netzwerk-Interface an dem eigenen Rechner zu erlauben. Entscheidend ist das zweite Element `sin_port`. Hier wird die so genannte Portnummer eingetragen – der Name oder die Adresse für den Socket. Damit wird das System informiert, dass alle an diesem Port eingehenden Nachrichten an das Programm weitergereicht werden sollen. Bestimmte Dienste sind mit festen Portnummern verknüpft (zum Beispiel Webserver Port 80). Unter Unix können normale Benutzer nur Portnummern oberhalb von 1024 verwenden.

Bei der Angabe der Portnummer gilt es, die für das TCP-Protokoll passende Darstellung einer Integer-Zahl zu beachten. Verschiedene Systeme unterscheiden sich in der Reihenfolge von Low-Byte und High-Byte. Das Makro `htons` (Host to network, short) wandelt Integer-Werte vom Typ `short` entsprechend um. Genauer gesagt tauscht es, falls erforderlich, Low-Byte mit High-Byte. Mit diesen Bestandteilen kann man in folgender Weise die Portnummer 5432 an den Socket binden:

```
#define  SERVER_PORT                5432

long    rc;
SOCKADDR_IN addr;
int  addrlen = sizeof(addr);

addr.sin_family = AF_INET;
addr.sin_port = htons(SERVER_PORT);
addr.sin_addr.s_addr = INADDR_ANY;

rc = bind(sockListen, (SOCKADDR*)&addr, sizeof(addr));
if (rc == SOCKET_ERROR) {
    printf("Error: Cannot bind Socket\n");
}
```

9.1.3 Funktion „listen“

Der Socket ist jetzt angelegt und mit einer Adresse versehen. Nun kann dem System mitgeteilt werden, dass der Server bereit ist, Anfragen zu bearbeiten. Die Funktion dazu ist

```
int listen( SOCKET sock, int backlog);
```

Das erste Argument ist wieder der Socket. Das zweite Argument gibt an, wie viele Anfragen die Warteschlange maximal halten soll. Ist der Server gerade beschäftigt, speichert die Warteschlange neu eintreffende Anfragen und gibt sie nach und nach zur Bearbeitung weiter.

Die Größe der Warteschlange ist durch die Systemressourcen beschränkt. Eine sinnvolle Spezifikation ist `SOMAXCONN`. Damit wird automatisch die Maximalzahl eingestellt. Der entsprechende Abschnitt im Programm lautet:

```
rc = listen(sockListen, SOMAXCONN);
if (rc == SOCKET_ERROR) {
    printf("Error: Cannot listen\n" );
}
```

9.1.4 Funktion „accept“

Wenn sich Anfragen in der Warteschlange befinden, können sie mit der Funktion `accept` angenommen werden. Die Definition lautet:

```
SOCKET accept( SOCKET s, sockaddr *p_addr, int * p_len );
```

Die Funktion `accept` wartet auf eine Anfrage für den angegebenen Socket. Erfolgt die Anfrage, gibt die Funktion einen neuen (Verbindungs-) Socket zurück. Dieser neue Socket wird dann für den Datenaustausch genutzt. Der alte Socket steht dadurch weiterhin für die Annahme weiterer Verbindungen bereit. Eine verbreitete Technik unter UNIX ist, mit dem neuen Socket einen Kindprozess zu starten. Der Kindprozess führt die gewünschte Aufgabe aus, während der Elternprozess bereits wieder neue Anfragen annehmen kann [15].

Das zweite Argument enthält nach der Rückkehr aus der Funktion Informationen über den Client. Dazu füllt `accept` die Struktur, auf die der angegebene Zeiger deutet. Die Länge der Struktur wird an die im dritten Argument über Zeiger angegebene Stelle geschrieben. Beide Argumente können auch NULL sein. In diesem Fall wird keine Information über den Client übermittelt.

9.1.5 Funktion „send“ und „recv“

Über den Verbindungs-Socket können Client und Server Daten austauschen. Aus Sicht der Applikation ähnelt der Verbindungs-Socket sehr stark einer Dateischnittstelle. In der Tat kann man auch mit den Systemfunktionen `read` und `write` sowohl Dateien als auch Sockets lesen beziehungsweise beschreiben.

Eine andere Möglichkeit besteht in der Verwendung der beiden Funktionen `recv` und `send`:

```
int recv( SOCKET s, char * buf, int len, int flags );  
int send( SOCKET s, char * buf, int len, int flags );
```

Diese Funktionen benutzen einen Puffer, um Daten zu übertragen. Bei `recv` spezifiziert `len` die Länge des Puffers. Demgegenüber wird bei `send` die Anzahl der zu übertragenden Zeichen mitgeteilt. Das vierte Argument übernimmt optionale Spezifikationen für die Ausführung. In den meisten Fällen ist der Standardwert 0 ausreichend. Beide Funktionen geben zurück, wie viele Werte sie tatsächlich empfangen beziehungsweise geschickt haben.

9.1.6 Einfacher Server

Mit den beschriebenen Funktionen lässt sich bereits ein Server realisieren. Als Client dient im Beispiel zunächst die Anwendung `telnet`. Mit dem Befehl `telnet IP-Adresse Portnummer` wird eine Verbindung hergestellt. Ohne physikalisches Netzwerk kann man zum Test auf dem gleichen Rechner die Loopback-Adresse `localhost` (127.0.0.1) verwenden. In diesem Fall gibt die Netzwerkkarte die Nachrichten direkt zurück.

Der folgende Programmcode implementiert einen einfachen Server. Der Server wartet auf Clients, schickt bei Anmeldung eine kurze Begrüßung und schließt danach unmittelbar wieder die Verbindung. Die Funktion `gethostname` wird dabei

verwendet, um den Namen des eigenen Rechners zu erfragen. Dieser Name wird dann in den Begrüßungstext eingebaut. Zu Gunsten einer besseren Übersicht verzichtet das Beispiel auf eine Fehlerbehandlung. Die Version enthält spezielle Aufrufe für Windows-Sockets. Die Socket-Funktionen stehen in der Bibliothek ws_32.lib. Diese muss beim Aufruf des Links zusätzlich mit angegeben sein.

Dieses und alle weiteren Listings des Compacts können Sie auch bequem über den Webcode a1541 von tecChannel.de downloaden.

```
#include <stdio.h>
#include <winsock2.h>

#define SERVER_PORT 1234
long WinsockStartup();

void main()
{
    long rc; /* Variable fuer Rueckgabewerte */
    SOCKET sockListen;
    SOCKET sockConnected;
    SOCKADDR_IN addr;
    int addrlen = sizeof(addr);
    char welcome[200];
    char hostname[200];

    addr.sin_addr.s_addr = 0;
    addr.sin_family = AF_INET;
    addr.sin_port = htons(SERVER_PORT);

    /* Nur unter Windows erforderlich */
    rc = WinsockStartup();

    /* Willkommenstext erstellen */
    gethostname( hostname, sizeof hostname );
    sprintf( welcome, "Willkommen bei host %s \r\n",
    ➡hostname);

    /* Socket anlegen und aktivieren */
    sockListen=socket(AF_INET,SOCK_STREAM,NULL);
    bind(sockListen, (SOCKADDR*)&addr, sizeof(addr));
    listen(sockListen, SOMAXCONN);
    /* In Endlosschleife auf Clients warten
     * Begrueßung schicken und
     * Verbindung wieder abbauen */
    for( ;; ) {
        printf( "Warten auf naechste Verbindung... \n" );
        sockConnected = accept( sockListen,
                                (SOCKADDR*)&addr, &addrlen);
        rc = send( sockConnected,
                   welcome, strlen(welcome) + 1, NULL);
```

```

        printf("%d Bytes geschickt\n", rc );
        /* Windows Socket beenden */
        shutdown( sockConnected, SD_SEND );
        closesocket( sockConnected );
    }
}

/* *****
* Starte Socket Service unter Windows
*/
long WinsockStartup()
{
    long rc;

    /* gewünschte Version */
    WORD wVersionRequested;
    /* Datenstruktur fuer Info ueber Version */
    WSADATA wsaData;

    wVersionRequested = MAKEWORD(2, 1); /* Short Wert aus
➡ 2 Bytes */
    rc = WSASStartup( wVersionRequested, &wsaData );
    return rc;
}

```

9.1.7 Funktion „connect“

Zur Realisierung eines eigenen Clients benötigt man noch die Funktion

```
int connect( SOCKET s, sockaddr *p_addr, int addrlen );
```

Der Client kann unmittelbar einen Socket mittels connect verbinden. In diesem Fall erhält er automatisch vom System eine freie Portadresse. Bei Erfolg (Rückgabewert 0) steht dann bereits ein Verbindungs-Socket zur Verwendung mit recv und send bereit. Die notwendigen Informationen über den Server – IP-Adresse und Portnummer – stehen in der Struktur, auf die der Zeiger p_addr deutet.

Der folgende Code zeigt einen einfachen Client. Dieser Client sucht nach einem Server auf Port 1234 auf dem lokalen Rechner. Nach jedem Versuch wartet er 500 ms bis zum nächsten Versuch. Gelingt die Verbindung zu einem Server, so wird eine Nachricht gelesen und der Text ausgegeben.

```

#include <stdio.h>
#include <winsock2.h>

#define SERVER_PORT          1234
#define RECV_BUF_MAXLEN     256

long WinsockStartup();
void main()

```

```

{
    SOCKET sock;
    SOCKADDR_IN addr;
    long rc;
    char recvBuf[RECV_BUF_MAXLEN+1];
    char hostAddress[] = "127.0.0.1";

    rc = WinsockStartup();

    // socket anlegen
    sock=socket(AF_INET, SOCK_STREAM, NULL);

    /* Informationen fuer Verbindung */
    addr.sin_family      = AF_INET;
    addr.sin_port        = htons(SERVER_PORT);
    addr.sin_addr.s_addr = inet_addr(hostAddress);

    // socket mit Server verbinden, endlos immer wieder
    ➔ probieren
    while(
        connect(sock, (SOCKADDR*)&addr, sizeof(SOCKADDR))
            == SOCKET_ERROR)
    {
        printf(".");    // Lebenszeichen ausgeben
        Sleep( 500 );   // 500ms warten
    }
    // Server gefunden
    printf("Verbunden mit %s ...\n", hostAddress);
    // Nachricht abholen und ausgeben
    rc = recv(sock, recvBuf, RECV_BUF_MAXLEN, NULL);
    recvBuf[rc] = '\0';
    printf("%d Byte empfangen: %s\n", rc, recvBuf );
}

```

9.2 Verbindungslose Kommunikation

Für eine verbindungslose Kommunikation sind weniger Schritte notwendig, da der Aufbau der Verbindung entfällt. Auf Seiten des Servers wird ein Socket angelegt und mit einem Port verbunden.

Der Client legt ebenfalls einen Socket an, den er mit einem Port verknüpfen kann. In den meisten Fällen ist dies aber nicht notwendig, da die Portnummer vom System automatisch beim ersten Verschicken eines Pakets zugewiesen wird.

Da mit einem Socket jetzt kein festes Ziel verbunden ist, muss die Zieladresse bei jedem Paket explizit mit angegeben werden. Daher wird anstelle der Funktion `send` die Funktion `sendto` verwendet. Die entsprechende Funktion zum Lesen ist `recvfrom`. Mit dieser Funktion kann ein Programm – ähnlich wie bei der Funktion `accept` beschrieben – Informationen über den Absender abfragen. Allerdings spielt

beim Lesen eines Pakets der Unterschied zwischen verbundenen und unverbundenen Sockets eine weniger große Rolle. Die Informationen über den Absender sind dabei optional. Wenn dafür kein Bedarf besteht, kann man auch die schon bekannte Funktion `recv` verwenden.

Tabelle 9.2: Funktionsaufrufe für verbindungslose Sockets

SERVER		CLIENT	
Endpunkt anlegen	<code>socket()</code>		
Adresse festlegen	<code>bind()</code>		
		Endpunkt anlegen	<code>socket()</code>
		Adresse festlegen	<code>bind()</code>
Daten lesen	<code>recvfrom()</code>	Daten schreiben	<code>sendto()</code>
Daten schreiben	<code>sendto()</code>	Daten lesen	<code>recvfrom()</code>

Der Ablauf soll wieder am Beispiel eines Programmpaares für Server und Client erläutert werden. Als minimale Funktionalität empfängt der Server Pakete und schickt jeweils eine Bestätigung an den Absender zurück. Damit eine Kommunikation überhaupt zu Stande kommen kann, verwendet der Server eine feste Portadresse. Der Client kann dann über den Rechnernamen und diese Portadresse ein Paket an den Server schicken. Mit diesem Paket werden die entsprechenden Adressen des Clients übermittelt. Daher kann der Client seine Portadresse frei wählen (oder vom System auswählen lassen). Der Server entnimmt die Informationen dem Paket und kann dann seine Bestätigung schicken. Für diesen Zweck stehen die beiden letzten Argumente der Funktion

```
int recvfrom( SOCKET s, char * buf, int len, int flags,
             sockaddr *p_addr, int *p_len );
```

zur Verfügung. Das vorletzte Argument enthält einen Zeiger auf eine Struktur, die die Informationen aufnehmen soll. Die tatsächliche Länge wird in die durch den Zeiger `p_len` angegebene Stelle geschrieben. Das Beispielprogramm wertet die Struktur aus, um die Angaben über den Absender auszugeben. Anschließend sendet der Server mit der Funktion

```
int sendto( SOCKET s, char * buf, int len, int flags,
           sockaddr *p_addr, int len );
```

eine Bestätigung (der Text OKAY). Bei dem Aufruf der Funktion `sendto` wird die Struktur mit den Informationen über den Absender wiederum zur Spezifikation der Zieladresse verwendet. Insgesamt hat der Server folgende Form:

```
/* Server für UDP */
#include <stdio.h>
#include <winsock2.h>
```

```

#define SERVER_PORT    5432
#define BUF_MAXLEN     256
long WinsockStartup();

void main()
{
    int    rc; /* Variable fuer Rueckgabewerte */
    int    al;

    SOCKET sock;
    SOCKADDR_IN addr;
    SOCKADDR_IN sender;

    char buffer[BUF_MAXLEN];

    addr.sin_addr.s_addr = 0;
    addr.sin_family      = AF_INET;
    addr.sin_port        = htons(SERVER_PORT);

    /* Nur unter Windows erforderlich */
    rc = WinsockStartup();

    /* Socket anlegen und binden */
    sock = socket(AF_INET, SOCK_DGRAM, NULL);
    bind(sock, (SOCKADDR*)&addr, sizeof addr);

    /* In Endlosschleife auf Pakete warten
     * */
    for( ;; ) {
        printf( "Warten auf naechstes Paket... \n" );
        al = sizeof sender;
        rc = recvfrom( sock, buffer, sizeof buffer, 0,
                       (SOCKADDR*)&sender, &al);
        printf("%d Bytes von ", rc );
        printf("Host: %s ", inet_ntoa ( sender.sin_addr ) );
        printf("Port: %d ", ntohs( sender.sin_port ) );
        printf("<%s>\n", buffer );
        // Bestaetigung schicken
        sendto( sock, "OKAY", 5, 0, (SOCKADDR*)&sender, al);
    }
}

```

Das Programm verwendet die beiden Funktionen `inet_ntoa` und `ntohs`. Die Funktion `inet_ntoa` wandelt die interne Darstellung der IP-Adresse in der Struktur in eine ASCII-Zeichenkette um. `ntohs` (Network to host short) ist das Gegenstück zu der bereits bekannten Funktion `htons`. Wie der Name sagt, konvertiert sie einen short-Wert aus dem standardisierten Netzwerkformat in die auf dem lokalen Rechner verwendete Zahlendarstellung.

Nach dem Starten wartet der Server auf Pakete. Wenn er ein Paket erhalten hat, zeigt er die Informationen über das Paket an und schickt ein Paket mit einer Bestätigung. Der folgende Code zeigt einen dazu passenden Client:

```
/* Client für UDP */
#include <stdio.h>
#include <winsock2.h>

#define SERVER_PORT      5432
#define BUF_MAXLEN      256

long WinsockStartup();

int main()
{
    SOCKET sock;
    SOCKADDR_IN addr;
    long rc;
    char buffer[BUF_MAXLEN];
    char hostAddress[] = "127.0.0.1";

    rc = WinsockStartup();

    // socket anlegen
    sock=socket (AF_INET, SOCK_DGRAM, NULL);

    // zuerst alles auf 0 setzten
    memset(&addr,0,sizeof(SOCKADDR_IN));
    addr.sin_family      = AF_INET;           // AdressFamilie
    addr.sin_port        = htons(SERVER_PORT); // PortNummer
    addr.sin_addr.s_addr = inet_addr(hostAddress); // host
    for(;;) {
        sendto( sock, "test", 5, 0, (SOCKADDR*)&addr, sizeof
➡SOCKADDR);
        rc = recvfrom( sock, buffer, sizeof buffer, 0, 0, 0);
        if( rc > 0 ) printf("%s\n", buffer);
        else        printf(".");
        Sleep( 500 );
    }
    return 0;
}
```

Der Client schickt ein Paket an den Server und wartet anschließend auf eine Bestätigung. Eine besondere Situation entsteht, wenn der vorhergehende `sendto`-Aufruf fehlschlug, da der Empfänger nicht erreichbar war. Der Socket erhält dann die ICMP-Meldung Port Unreachable. In diesem Zustand wird die Funktion `recvfrom` unmittelbar beendet und gibt einen entsprechenden Fehlerwert zurück. Das Programm erkennt durch den Vergleich des Rückgabewertes auf kleiner 0 solche Fehler und meldet dies durch die Ausgabe eines Punkts.

9.3 Realisierung in Perl

Wie in der Einleitung erwähnt, ist die Socket-Schnittstelle nicht auf die Sprache C beschränkt, sondern steht auch in anderen Programmiersprachen zur Verfügung. Das Socket-API kapselt die Netzwerkkommunikation soweit, dass auch Anwendungen in verschiedenen Programmiersprachen über Sockets Informationen austauschen können. Als ein Beispiel sei ein UDP-Client für den oben beschriebenen Server in der Sprache Perl (Practical Extraction and Report Language) [16] angegeben. Der folgende Code zeigt einen Client, der Eingaben von der Tastatur annimmt, jede Zeile an den Server schickt, die jeweilige Antwort empfängt und am Bildschirm ausgibt.

```
# UDP-Client
use Socket;

$port = 5432;
$dest = 'localhost';

# UDP-Socket anlegen
socket( S, PF_INET, SOCK_DGRAM, 0) || die "socket: $!";

# Adresse des Servers
$hisiaddr = inet_aton($dest)           || die "unknown host";
$hispaddr = sockaddr_in($port, $hisiaddr);

# Eingaben einlesen, an Server schicken und Antwort ausgeben
while( <STDIN> ) {
    send(S, $_, 0, $hispaddr)          || die "send: $!";
    $ok = '';
    recv(S, $ok, 10, 0)                 || die "recv: $!";
    print "$ok\n";
}
```

Im Wesentlichen entsprechen der Ablauf und die Funktionsaufrufe dem des C-Programms. Eine Vereinfachung ergibt sich aus der Tatsache, dass Perl Zeichenketten als eigener Typ behandelt. Daher wird im Aufruf von send keine Längenangabe benötigt. Es genügt, die Variable \$_ mit der aktuellen Zeile zu übergeben.

Prof. Dr. Stephan Euler

tecCHANNEL-Links zum Thema	Webcode	Compact
Sockets	a1499	S.88
Sourcecode und Listings zum Download	a 1541	–
.NET-Grundwissen für Admins und Programmierer	p1208	–

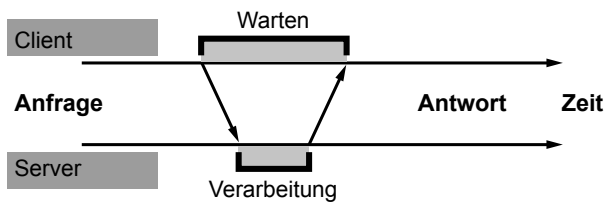
10. Remote Procedure Call RPC

Häufig basieren Netzerkwendungen auf einem Frage-und-Antwort-Schema. Ein Prozess (der Client) auf einem Rechner beauftragt einen zweiten Prozess (den Server) auf einem anderen Rechner, eine bestimmte Aktion auszuführen und ein dazu gehörendes Ergebnis zu melden. In diesem Szenario steht weniger die Datenübertragung als die Aufteilung der Arbeit im Mittelpunkt.

Betrachten wir als Beispiel einen Datenbank-Server. Ein Client fragt nach allen Einträgen zu einem bestimmten Nachnamen. Der Ablauf ist:

1. Der Client schickt eine Abfrage mit dem Nachnamen und wartet danach.
2. Der Server sucht nach entsprechenden Einträgen.
3. Der Server sendet die Liste mit gefundenen Einträgen zurück.
4. Der Client empfängt die Antwort und arbeitet damit weiter.

Folgendes Bild zeigt den Ablauf in der Darstellung als Zeitstrahl.



© tecCHANNEL

Bild 10.1: Ablauf in einem Frage-und-Antwort-Schema.

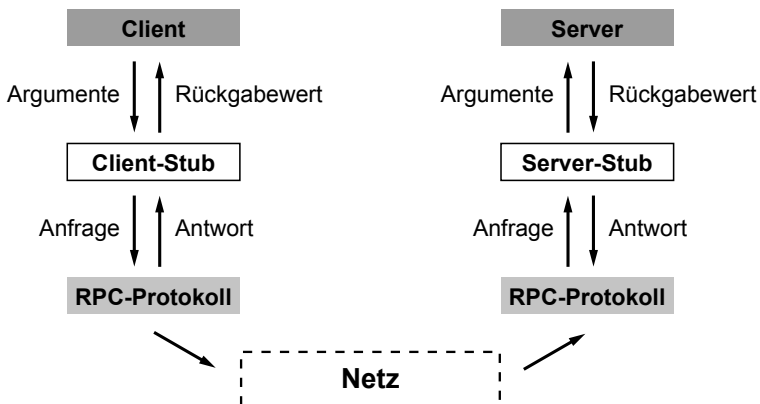
Bei dem Ansatz mit Remote Procedure Call RPC betrachtet man diesen Ablauf wie den Aufruf eines Unterprogramms. Der Client entspricht dem übergeordneten Modul, das eine Funktion mit einer Argumentliste und Rückgabewerten aufruft. Der RPC-Formalismus unterstützt diese Sichtweise durch eine Reihe von Werkzeugen. Mit deren Hilfe erfolgt die Umsetzung in den auf unterer Schicht ablaufenden Austausch von Datenpaketen weit gehend automatisch. Im Allgemeinen können die beiden Prozesse in verschiedenen Programmiersprachen geschrieben worden sein und auf verschiedenen Betriebssystemen laufen. Daher ist es notwendig, eine plattformunabhängige Repräsentation der Daten einzusetzen.

Im Folgenden wird zunächst eine kurze Beschreibung der grundlegenden Mechanismen von RPC gegeben. RPC ist allerdings eher ein allgemeines Konzept als ein gemeinsames Protokoll. Es gibt eine ganze Reihe von Implementierungen (Sun RPC, Microsoft RPC et cetera), die sich in den Details unterscheiden. Daneben stehen in anderen Programmiersprachen ähnliche Möglichkeiten zur Verfügung. Java bezeichnet dieses Konzept als Remote Methode Invocation (RMI). Im Folgenden werden Details der Realisierung an Hand eines einfachen Beispiels für

ein Telefonbuch besprochen. Die Entwicklung erfolgt mit Visual C unter Windows 2000. Die Darstellung versteht sich als Einführung in die Programmierung mit RPC anhand der Basisfunktionalitäten. Für weitere Möglichkeiten sei auf die umfangreiche Dokumentation verwiesen [17].

10.1 Grundlagen

RPC soll möglichst weit gehend alle netzwerkrelevanten Details übernehmen. Im Idealfall ist für den Anwender nicht sichtbar, dass die Anwendung über ein Netzwerk läuft. Die Kapselung erfolgt durch so genannte Stubs (Englisch: (Baum)Stumpf, Stummel). Die Stubs wirken als Mittler zwischen der Anwendung und dem RPC-Protokoll. Ein entsprechender Generator oder Compiler erzeugt die Stubs für die jeweilige Programmiersprache automatisch aus einer abstrakten Definition des Interface. Der Client ruft den Client-Stub als normale Prozedur auf und übergibt die notwendigen Argumente. Im Stub erfolgen die Umsetzung auf eine netzwerktaugliche Darstellung und der Aufruf des RPC-Protokolls.



© tecCHANNEL

Bild 10.2: Stubs setzen die Programmzugriffe auf das RPC-Protokoll um.

10.2 RPC-Beispiel

Als Beispiel soll ein Server für Anfragen nach Telefonnummern realisiert werden. Im ersten Schritt der RPC-Entwicklung muss der Programmierer einen eindeutigen Bezeichner für das so genannte Interface erzeugen. Dazu steht das Programm uuidgen zur Verfügung. Eventuell ist zunächst ein Aufruf von vcvars32.bat nötig, damit die Suchpfade entsprechend gesetzt sind.

Folgender Aufruf erzeugt dann eine Datei mit einem eindeutigen Identifikator (Universally Unique Identifier, UUID):

```
uuidgen /i /otelbu.idl
```

Im Beispiel resultierte daraus die Datei telbu.idl mit diesem Inhalt:

```
//file telbu.idl
[
  uuid(49db6843-b369-4551-86f6-b3254a9b8a1a) ,
  version(1.0)
]
interface INTERFACENAME
{

}
```

In diese IDL-Datei (IDL: Interface Definition Language) trägt man als Nächstes einen Namen für das Interface sowie die vorgesehenen Prozeduren ein. Das Interface bekommt den Namen telbu als Abkürzung von Telefonbuch. Zunächst ist nur eine einzige Prozedur testKomm als Test für die Kommunikation vorgesehen. Diese Prozedur erhält als Argument eine Zeichenkette, die dann als Bestätigung für eine geglückte Übertragung auf der Seite des Servers ausgegeben wird. Damit erhält man:

```
//file telbu.idl
[
  uuid(49db6843-b369-4551-86f6-b3254a9b8a1a) ,
  version(1.0)
]
interface telbu
{
  void testKomm([in, string] unsigned char * testString);
}
```

In der Deklaration sind zusätzliche Attribute für den Parameter eingetragen. Die Angabe „in“ legt fest, dass es sich um einen Eingabewert handelt, den die Prozedur nicht überschreiben darf. Das zweite Attribut string deklariert den Parameter als eine gemäß der C-Konvention mit '\0' abgeschlossene Zeichenkette. Damit kann der Compiler die Funktion strlen einsetzen, um die Länge des zu übertragenden Datenbereichs zu bestimmen. Weiterhin benötigt man ein so genanntes Application Configuration File (ACF).

```
//file: telbu.acf
[implicit_handle (handle_t telbu_IfHandle)
] interface telbu
{
}
```

Aus diesen beiden Dateien erzeugt der MIDL-Compiler (MIDL steht für Microsoft IDL) mit dem Aufruf `midl telbu.idl` den C-Code für den Server- und Client-Stub. Die beiden Dateien erhalten die Namen `telbu_s.c` und `telbu_c.c`. Weiterhin legt er eine Header-Datei `telbu.h` an.

10.3 RPC-Server

Nach diesen Vorbereitungen lässt sich der Server implementieren. Den RPC-Mechanismus kann der Programmierer auf verschiedene Protokolle aufsetzen. Das gewünschte Protokoll wird in Form einer Zeichenkette spezifiziert. Die Zeichenkette besteht aus einer Kennung für das RPC-Protokoll: `ncacn` oder `ncadg` für strom- oder paketorientierte Kommunikation. Anschließend folgen die Kennungen für ein Vermittlungsprotokoll, wie etwa IP, und ein Transportprotokoll, zum Beispiel TCP. Die einzelnen Bestandteile sind durch `_`-Zeichen getrennt.

Damit ist die Spezifikation für TCP/IP `ncacn_ip_tcp`. Ein anderes Beispiel ist `ncacn_np` für die Kommunikation über named pipes. Das Protokoll zusammen mit dem Namen des Endpunkts wird mit der Funktion `RpcServerUseProtseqEp` angegeben. Anschließend wird mit `RpcServerRegisterIf` das Interface registriert. Schließlich wartet mit `RpcServerListen` der Server auf Anfragen. Insgesamt erhält man folgenden Code für den Server. Alle Listings finden Sie auch online zum Download unter dem Webcode a1541 auf tecChannel.de.

```
/* file: telbus.c */
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include "telbu.h"

void main()
{
    RPC_STATUS status;
    unsigned char *pszProtocolSequence = "ncacn_ip_tcp";
    unsigned char *pszSecurity         = NULL;
    unsigned char *pszEndpoint         = "5432";
    unsigned int   cMinCalls            = 1;
    unsigned int   cMaxCalls            = 20;
    unsigned int   fDontWait            = FALSE;

    printf( "RpcServerUseProtseqEp\n" );
    status = RpcServerUseProtseqEp(pszProtocolSequence,
                                   cMaxCalls,
                                   pszEndpoint,
                                   pszSecurity);

    if (status) exit(status);
    printf( "RpcServerRegisterIf\n" );
```

```

status = RpcServerRegisterIf(telbu_v1_0_s_ifspec,
                             NULL,
                             NULL);
if (status) exit(status);

printf( "RpcServerListen\n");
status = RpcServerListen(cMinCalls,
                        cMaxCalls,
                        fDontWait);

if (status) exit(status);

} // end main()

/*****
/*      MIDL allocate and free      */
*****/

void __RPC_FAR * __RPC_USER midl_user_allocate(size_t len)
{
    return(malloc(len));
}

void __RPC_USER midl_user_free(void __RPC_FAR * ptr)
{
    free(ptr);
}

```

Zusätzlich werden zwei Routinen zum Reservieren und Freigeben von Speicher benötigt. Die beiden Routinen sind hier lediglich Hüllen (wrapper) um die entsprechenden Funktionen malloc und free aus der Standardbibliothek von C.

10.4 RPC-Client

Der Client muss zunächst die RPC-Verbindung spezifizieren. Er benutzt dazu die Funktion RpcStringBindingCompose, um aus einer Anzahl von Parametern eine Zeichenkette zu konstruieren. Abhängig vom Protokoll müssen nicht unbedingt alle Parameter gesetzt werden. So benötigt das Protokoll über named pipes keine Angabe der Netzwerkadresse, da es nur zwischen Prozessen auf einem Rechner verwendet werden kann.

Diese Zeichenkette wird dann an RpcBindingFromStringBinding übergeben. Bei Erfolg ist das Interface aktiviert, und das zweite Argument enthält einen gültigen so genannten Handle auf das Interface. Danach kann der Client die Prozeduren aufrufen. Die Aufrufe sind im Beispiel in einen speziellen RpcTryExcept Block eingebettet, der eventuelle Laufzeitfehler auffängt. Der Client ruft in diesem Beispiel zehn Mal die Prozedur testKomm auf. Am Ende werden noch die belegten Ressourcen wieder freigegeben. Das Programm hat folgende Form:

```

/* file: telbuc.c */
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include "telbu.h"

void main()
{
    RPC_STATUS status;
    unsigned char *pszUuid          = NULL;
    unsigned char *pszProtocolSequence = "ncacn_ip_tcp";
    unsigned char *pszNetworkAddress  = "127.0.0.1";
    unsigned char *pszEndpoint        = "5432";
    unsigned char *pszOptions         = NULL;
    unsigned char *pszStringBinding    = NULL;
    unsigned char *testString         = "Hallo Server";
    unsigned long ulCode;

    status = RpcStringBindingCompose(pszUuid,
                                     pszProtocolSequence,
                                     pszNetworkAddress,
                                     pszEndpoint,
                                     pszOptions,
                                     &pszStringBinding);
    if (status) exit(status);
    printf("pszStringBinding: %s\n", pszStringBinding );

    status = RpcBindingFromStringBinding(pszStringBinding,
                                         &telbu_IfHandle);
    if (status) exit(status);

    RpcTryExcept {
        int i;
        // mehrfach die Test-Prozedur aufrufen
        for( i=0; i<10; i++ ) {
            printf("%d ", i);
            testKomm(testString);
            Sleep( 1000 );
        }
    }
    RpcExcept(1) {
        ulCode = RpcExceptionCode();
        printf("Runtime exception 0x%x = %ld\n", ulCode, ul-
Code);
    }
    RpcEndExcept

    status = RpcStringFree(&pszStringBinding);
    if (status) exit(status);

```

```

    status = RpcBindingFree(&telbu_IfHandle);
    if (status) exit(status);

    exit(0);
} // end main()

/*****
/*      MIDL allocate and free      */
*****/

void __RPC_FAR * __RPC_USER midl_user_allocate(size_t len)
{
    return(malloc(len));
}

void __RPC_USER midl_user_free(void __RPC_FAR * ptr)
{
    free(ptr);
}

```

10.5 Compilieren und Linken

Die Aufrufe zum Übersetzen und Linken der Programme sind vorteilhaft in einer Make-Datei zusammengefasst. Der folgende Text in der Datei makefile wird dann durch nmake ausgewertet, und die notwendigen Aufrufe von Compiler und Linker werden gestartet. Das Makefile enthält auch die Abhängigkeiten von der IDL-Datei. Nach einer Änderung an dieser Datei erzeugt ein Aufruf von nmake automatisch die gesamte Verarbeitungskette. Am Ende erhält man die beiden Anwendungen telbus und telbuc für Server und Client. Die beiden Anwendungen kann man in zwei Eingabefenstern ausführen.

Noch ein wichtiger Hinweis: nmake erwartet bei der Angabe der Abhängigkeiten in makefile jeweils ein Tabulatorzeichen nach dem Doppelpunkt. Dieses Zeichen darf nicht durch Leerzeichen ersetzt werden.

```

#makefile for telbuc.exe and telbus.exe
#link refers to the linker
#conflags refers to flags for console applications
#conlibs refers to libraries for console applications

!include <ntwin32.mak>

all : telbuc.exe telbus.exe

# Make the client side application telbuc
telbuc.exe : telbuc.obj telbu_c.obj
    $(link) $(linkdebug) $(conflags) -out:telbuc.exe \
        telbuc.obj telbu_c.obj \

```

```

        rpcrt4.lib $(conlibs)

# telbuc main program
telbuc.obj : telbuc.c telbu.h
        $(cc) $(cdebug) $(cflags) $(cvars) $*.c

# telbuc stub
telbu_c.obj : telbu_c.c telbu.h
        $(cc) $(cdebug) $(cflags) $(cvars) $*.c

# Make the server side application telbus
telbus.exe : telbus.obj proc.obj telbu_s.obj
        $(link) $(linkdebug) $(conflags) -out:telbus.exe \
        telbus.obj telbu_s.obj proc.obj \
        rpcrt4.lib $(conlibsmt)

# telbu server main program
telbus.obj : telbus.c telbu.h
        $(cc) $(cdebug) $(cflags) $(cvarsmt) $*.c

# remote procedures
proc.obj : proc.c telbu.h
        $(cc) $(cdebug) $(cflags) $(cvarsmt) $*.c

# telbus stub file
telbu_s.obj : telbu_s.c telbu.h
        $(cc) $(cdebug) $(cflags) $(cvarsmt) $*.c

# Stubs and header file from the IDL file
telbu.h telbu_c.c telbu_s.c : telbu.idl telbu.acf
        midl telbu.idl

```

Mit der erfolgreichen Ausführung von `nmake` werden die beiden Dateien `telbus.exe` und `telbuc.exe` erzeugt. Nach dem Start wartet der Server auf Anfragen von Client-Prozessen. Der Client ruft in einer Schleife mehrfach die Prozedur `testkomm` auf. Bei jedem Aufruf gibt der Server die erhaltene Zeichenkette aus.

10.6 Abfrage-Prozedur

Nach diesen Vorbereitungen kann man die eigentliche Funktionalität des Telefonbuchbeispiels angehen. Zunächst wird eine Struktur mit zwei Zeichenketten zur Aufnahme der Daten `Name` und `Nummer` eingeführt. Der folgende Codeabschnitt legt ein Feld der Größe 100 an. Er füllt die beiden ersten Einträge und setzt einen Zähler `count` auf die Anzahl zwei:

```

\\ in Datei proc.c
struct {
    char *name;

```

```
char *number;
} tb[100] = {"Maier",    "123 567",
            "Schmidt", "33 33 22"};
int count = 2;
```

Bei der Übergabe von Argumenten muss der Software-Entwickler die Besonderheiten der Netzwerkkommunikation beachten. So ist es wenig sinnvoll, einen Zeiger zu übergeben. Die Information, dass die Daten an der angegebenen Speicheradresse stehen, hat für einen anderen Prozess auf einem anderen Rechner keine Bedeutung. Vielmehr müssen die Daten in geeigneter Form übertragen werden. Der RPC-Compiler übernimmt die Aufgabe, aus den Definitionen in der IDL-Datei den entsprechenden Code zu generieren.

Allerdings sind dabei die beschriebenen Beschränkungen zu beachten. Es ist konkret nicht möglich, wie sonst in C üblich, das Ergebnis einer Anfrage nach einer Telefonnummer als Zeiger auf das Resultat zurückzugeben. Eine Möglichkeit zur Übergabe des Resultats ist jedoch, das Ergebnis in ein Feld vorgegebener Größe zu schreiben. Die IDL-Datei hat dann folgende Form:

```
//file telbu.idl
[
  uuid(49db6843-b369-4551-86f6-b3254a9b8a1a),
  version(1.0)
]
interface telbu
{
  #define STRSIZE 100
  cpp_quote("#define STRSIZE 100")
  void testKomm([in, string] unsigned char * testString);
  int getNumber( [in, string] unsigned char * name,
                 [out ] unsigned char number[STRSIZE]);
}
```

Die Prozedur `getNumber` enthält zwei Parameter:

1. `name` mit dem Namen, für den der Client eine Telefonnummer sucht.
2. `number` als Feld, in das der Server das Ergebnis der Suche einträgt.

Der Integer-Rückgabewert ist als Ergebnismeldung vorgesehen. Das Feld `number` ist mit einer festen Größe definiert. Zur besseren Übersicht ist die Größe über eine symbolische Konstante `STRSIZE` festgelegt. Die Anweisung `define` ist nur für die Auswertung durch den Generator `midl` gültig. Die Anweisung `cpp_quote` schreibt eine Kopie der Definition der Konstanten in die Datei `telbu.h`. Dadurch ist diese Definition auch in den C-Programmen zugänglich. Die Prozedur wird durch folgende C-Funktion realisiert:

```
\\ in Datei proc.c
int getNumber( unsigned char *name, unsigned char *number ){
    int i;
    strcpy( number, "");
}
```

```

for( i=0; i<count; i++ ) {
    if( strstr( tb[i].name, name ) ) {
        strcpy( number, tb[i].number);
        return i;
    }
}
return -1;
}

```

Mit der Bibliotheksfunktion `strstr` wird geprüft, ob die gesuchte Zeichenkette in einem der Namensfelder enthalten ist. Findet sich ein entsprechender Eintrag, kopiert der Server die zugehörige Telefonnummer in das Ergebnisfeld und beendet die Funktion. Als Rückgabewert dient der Index des Eintrags. Findet er keinen passenden Eintrag, signalisiert er dies durch den Rückgabewert -1.

In den Client wird eine Schleife mit Abfragen wie folgt eingefügt. Der Client liest dabei jeweils ein Wort von der Kommandozeile ein und ruft damit die Prozedur zur Suche auf.

```

/* file: telbuc.c */
...
unsigned char number[STRSIZE], name[STRSIZE];
...
while( scanf("%s", name) ) {
    if( getNumber( name, number ) < 0 ) {
        printf("Kein Eintrag\n");
    } else {
        printf( "%s\n", number );
    }
}
...

```

Prof. Dr. Stephan Euler

tecCHANNEL-Links zum Thema	Webcode	Compact
Remote Procedure Call RPC	a1500	S.100
Alle Listings zum Download	a1541	–
Das bringt Visual Studio 2005 „Whidbey“	p1440	–
Die .NET CLR – Version 2.0	a1457	–
.NET-Grundwissen für Admins und Programmierer	p1208	–
C# für Ein- und Umsteiger	p1227	–

11. Netzerkanwendungen

Auf die von Ende-zu-Ende-Protokollen wie TCP und UDP bereitgestellten Dienste bauen verbreitete Anwendungen wie E-Mail oder WWW für die Benutzer auf. Einige davon, wie etwa telnet, nutzen direkt den Transportdienst, während andere zusätzliche eigene Kommunikationsprotokolle einführen. In diesem Kapitel wird am Beispiel einiger weit verbreiteter Anwendungen diskutiert, wie diese die Funktionalität der Transportschicht nutzen.

11.1 World Wide Web WWW

Die wohl derzeit am meisten genutzte Anwendung über Rechnernetze ist das World Wide Web WWW. An diesem Beispiel lassen sich gut einige grundlegende Konzepte darstellen.

Auf Seite des Benutzers steht ein Anwendungsprogramm (User Interface), das Informationen in textueller und grafischer Form darstellt und Eingaben des Benutzers annimmt. Dieses Anwendungsprogramm ist der Webclient oder Webbrowser wie etwa Opera, Firefox oder der Internet Explorer. Auf der anderen Seite steht ein Webserver, der in der Regel über ein Netzwerk zu erreichen ist. Der Webserver reagiert auf Anfragen des Clients und schickt die angeforderten Informationen sowie Bestätigungsmeldungen.

Die Kommunikation zwischen Server und Client erfolgt über TCP. Es stellt einen zuverlässigen, bidirektionalen Byte-Strom bereit. Die Kommunikation erfolgt aber in Form von Anfragen und Antworten. Daher benötigt man ein Protokoll, das über den TCP-Strom einen geeigneten Anfragen- und Antwortmechanismus realisiert. Dieses Protokoll nennt sich HTTP: HyperText Transport Protocol.

HTTP ist ein einheitliches Anwendungsprotokoll, mit dem Clients von unterschiedlichen Herstellern mit beliebigen Webservern kommunizieren können. Die Clients und Server können dabei auf unterschiedlichen Hard- und Software-Plattformen laufen. Auch die Darstellung der Information durch den Client kann sich nach den jeweiligen Gegebenheiten stark unterscheiden und text- oder grafikbasierend sein. Aber das gemeinsame Anwendungsprotokoll ermöglicht über all diese Grenzen hinweg die Interoperabilität zwischen Server und Client.

HTTP legt in seinem Anwendungsprotokoll fest, welche Anfragen möglich sind und wie die Antworten darauf aussehen. Der Inhalt der ausgetauschten Informationen ist hierbei irrelevant. HTTP ist lediglich für den Transport zuständig. Die Informationsdarstellung ist in dem Begleitprotokoll HTML (HyperText Markup Language) geregelt. In HTML ist der Aufbau der Seite mit Formatierungsinformationen wie etwa Textart, Textgrößen, Farben sowie die Einbindung von Elementen wie Bilder oder Video-Clips beschrieben. Weiterhin kann ein Dokument Verweise (Links) auf andere Dokumente enthalten. Erweiterungen wie Java Applets, Javascript oder Flash können in die Dokumente integriert werden.

Diese Unterteilung in Client-Anwendung, Server-Anwendung, Anwendungsprotokoll und eventuelles Begleitprotokoll findet man bei vielen Anwendungen wieder. In manchen Fällen ist die Trennung weniger deutlich sichtbar, wenn wie bei FTP die Anwendung den gleichen Namen wie das Anwendungsprotokoll trägt.

11.1.1 Das Protokoll HTTP

Das Protokoll HTTP besteht aus Anfrage- und Antwortnachrichten. HTTP tauscht diese Nachrichten in lesbarer Form als ASCII-Texte aus. Es ist daher ohne Weiteres möglich, über eine telnet-Verbindung selbst HTTP-Anfragen an einen Server zu schicken.

Als Beispiel soll der Ablauf für die elementare Operation zum Lesen einer HTML-Datei dienen. Der Standort der Datei ist als Universal Resource Locator URL spezifiziert. Ein solcher URL ist `http://www.fh-friedberg.de/index.html`. Der Ablauf ist dann wie folgt:

1. Der Client extrahiert aus dem URL den Knotennamen `www.fh-friedberg.de` des Servers.
2. Über DNS ermittelt der Client die IP-Adresse.
3. Er baut eine TCP-Verbindung zu Port 80 des Servers auf.
4. Der Client schickt die Abfrage für die Datei `index.html`.
5. Der Server antwortet und überträgt die Datei.
6. Die TCP-Verbindung wird abgebaut. (Ab der Version 1.1 besteht die Möglichkeit, die TCP-Verbindung aufrechtzuerhalten, um weitere Elemente über die gleiche Verbindung laden zu können.)
7. Der Browser analysiert den HTML-Code der Seite und zeigt den Text an. Gegebenenfalls holt er weitere benötigte Elemente wie etwa Bilder.

Die Anfragenachricht für die Datei `index.html` hat die Form:

```
GET http://www.fh-friedberg.de/index.html HTTP/1.1
```

Sie besteht aus drei Teilen:

1. dem Befehl GET
2. dem URL
3. einer Kennung der verwendeten Version des Protokolls.

Im Allgemeinen können auf die erste Zeile der Abfrage noch mehrere weitere Zeilen mit Optionen folgen. Das Ende der Nachricht wird durch eine Leerzeile markiert. Folgende Tabelle enthält wichtige Anfrageoptionen in HTTP. Die Optionen ermöglichen im Wesentlichen das Lesen, Schreiben und Löschen von Dokumenten auf dem Server.

Tabelle 11.1: Anfrageoptionen in HTTP

GET	Lesen eines Dokuments
HEAD	Lesen des Headers eines Dokuments
PUT	Schreiben eines Dokuments
POST	Anhängen von Daten an bestehendes Dokument
DELETE	Löschen eines Dokuments
OPTIONS	Abfrage von verfügbaren Optionen
TRACE	Zu Testzwecken wird die Anfrage wie erhalten an den Client zurückgeschickt

In der ersten Zeile der Antwortnachricht sendet der Server seine Versionsnummer für HTTP und einen Ergebniscode mit einem erläuternden Text. Daran anschließend können optionale Informationszeilen folgen.

Das Format dabei ist „Informationsart : Informationstext“. Typische Informationen sind eine Beschreibung des Inhaltes oder das Datum der letzten Änderung. Bei der Abfrage GET folgt dann der Inhalt des angeforderten Dokuments. Als Beispiel erhält man mit der oben angegebenen Abfrage die Antwort

```
HTTP/1.0 200 OK
Server: WEBULA/1.2.3
Date: Wed, 19 Jun 2002 08:51:23 CEST
Last-Modified: Wed, 05 Jun 2002 11:00:27 CEST
MIME-Version: 1.0
Content-Type: text/html
Content-length: 78
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//
EN">
<HTML>
<HEAD>
...
</HTML>
```

Die erste Zeile enthält in diesem Beispiel die HTTP-Version der Nachricht und den Code 200 mit der Erläuterung OK als Bestätigung für eine erfolgreiche Transaktion. Eine Übersicht über die Statuscodes in HTTP gibt folgende Tabelle. Die nächsten Zeilen enthalten Informationen über den Server und Datumsangaben.

Wichtig für die weitere Bearbeitung durch den Client sind die Einträge Content Type und Content Length. Content Type beschreibt den MIME-Typ der im Datenbereich übermittelten Datei. Im Header-Feld Content Length gibt der Server die Länge der Daten in Byte an. Der Einsatz des Feldes ist dabei nicht zwingend vorgeschrieben. Das RFC gibt jedoch die Empfehlung, das Feld immer zu senden. Getrennt durch eine Leerzeile folgt dann der HTML-Code der Seite.

Tabelle 11.2: Typen der Statuscodes in HTTP

Code	Typ	Erklärung
2xx:	Erfolg	Aktion empfangen, verstanden und ausgeführt
200		OK
204		No Content: Methode war erfolgreich, jedoch folgt keine Antwort im Rest der Nachricht
3xx:	Redirection	Weitere Aktionen sind erforderlich
301		Moved Permanently: Die angeforderte Seite ist dauerhaft umgezogen
4xx:	Client-Fehler	Anfrage enthält fehlerhafte Syntax oder ist nicht ausführbar
401		Unauthorized: keine Berechtigung für die angeforderte Seite
5xx:	Server-Fehler	Eine gültige Anfrage führt zu einem Fehler im Server
500		Internal Server Error

11.1.2 Universal Resource Identifier

HTTP spricht Dokumente über eine Adresse in Form eines Universal Resource Locator (URL) an. Der URL ist einer der beiden Unterkategorien des Universal Resource Identifier (URI). Deren zweite Unterkategorie sind die Universal Resource Names (URN), zu denen etwa E-Mail- oder News-Adressen gehören. Der URI ist ein allgemeines Schema, um Ressourcen im Netzwerk zu adressieren. Die Spezifikation ist in der RFC 2396: „Uniform Resource Identifiers (URI): Generic Syntax“ beschrieben.

Die allgemeine Form des URI ist „Schema:schemaspezifischer Teil“. Der Schemateil beschreibt dabei den Dienst, im obigen Beispiel also http. Daneben sind noch ftp, news, file, telnet oder mailto gängig. Browser erlauben in der Regel auch die Eingabe von anderen Diensten als HTTP und starten dann die entsprechende Anwendung automatisch.

Der zweite Teil hat bei HTTP die allgemeine Form „//user:password@host:port/pfad“. In den meisten Fällen wird aber nur der Pfad benötigt. Weiterhin gelten Vereinbarungen wie etwa die Verwendung von index.html als Default-Wert für den Dateinamen.

Neben dem eigentlichen Namen kann man weitere Optionen an den URI anhängen. Dies ist eine einfache Möglichkeit für den Client, Informationen an den Server zu schicken. Der Server entnimmt dann die Optionen dem erweiterten URI und erzeugt eine Seite mit der angeforderten Information. Als Beispiel ist der folgende URI die Anfrage an eBay nach Artikeln mit dem Schlüsselwort Friedberg (Zeilenwechsel und Leerzeichen sind nur zur besseren Übersicht eingefügt):

```
http://search.ebay.de/search/search.dll?MfcISAPICommand=
➔GetResult
  &ht=1 &shortcut=4
  &SortProperty=MetaEndSort
  &maxRecordsPerPage=50
  &st=2 &ebaytag1code=77
  &query=friedberg
```

Der Teil bis zu dem Fragezeichen spezifiziert die Anwendung. Der Rest wird dann beim Aufruf als Argument an diese Anwendung übergeben. Im Beispiel besteht das Argument aus einer Liste mit Eigenschaften und den dazugehörigen Werten. Die Anwendung wird vom Browser aktiviert. Sie liest die Argumente, führt auf dem Server eine entsprechende Aktion aus und gibt das Resultat als HTML-Text an den Browser zurück.

11.1.3 HTTP-Cache

Wenn für jedes Lesen einer Webseite der komplette Inhalt neu über das Netz geladen wird, entsteht sehr viel Datenverkehr. In vielen Fällen ist das neue Lesen aller Elemente aber gar nicht erforderlich, da sich deren Inhalt nicht geändert hat. Daher ist es sinnvoll, die gelesenen Seiten und deren Elemente, etwa darin enthaltene Bilder, in einem temporären Speicher, den Cache (Englisch: geheimes Lager), abzulegen. Bei einem erneuten Aufruf der Seite kann der Browser dann die Kopie aus dem Zwischenspeicher verwenden. Dadurch entlastet der Cache nicht nur das Netz, sondern beschleunigt auch den Seitenaufbau.

Die Zwischenspeicherung kann an mehreren Stellen erfolgen. Zunächst kann der Browser Daten lokal auf der Festplatte ablegen. Weiterhin kann in einem lokalen Netz ein Knoten auch einen gemeinsamen Cache anbieten. Dadurch nutzen mehrere Clients die Kopie gemeinsam, was den Datenverkehr weiter verringert.

Knoten mit dieser Funktionalität nennt man Proxy (Englisch: Stellvertreter). Neben dem Caching übernehmen Proxies weitere Aufgaben wie die Regelung von Zugriffsrechten oder die Umsetzung von Protokollen. So kann ein Proxy für einen Client, der nur HTTP unterstützt, eine Umsetzung auf FTP vornehmen, damit dieser einen FTP-Server abfragen kann.

Schließlich kann auch ein Internet Service Provider (ISP) in seinem Netz einen oder mehrere Knoten mit Cache installieren. Dadurch entlastet er seine Leitungen und reduziert gleichzeitig die Kosten für den Datenverkehr in fremde Netze.

Das Design von HTTP unterstützt Caching durch mehrere Elemente. Bevor eine Seite aus dem Cache verwendet wird, muss sichergestellt sein, dass die Seite noch aktuell ist. Dazu weist der Server beim Verschicken jeder Seite in dem Optionsfeld Expires ein Gültigkeitsdatum zu. Bis zu diesem Datum kann der Cache davon ausgehen, dass die Seite auf jeden Fall aktuell ist.

Bei Zugriffen nach dem Gültigkeitsdatum prüft der Proxy über die Abfrageoption HEAD oder eine spezielle Variante von GET, ob sich die Seite geändert hat. In

diesem Fall fordert er sie erneut an. Damit der Cache nicht über eine voreingestellte Größe wächst, entfernt der Proxy zudem Seiten aus dem Cache, auf die längere Zeit kein Browser mehr zugegriffen hat.

Ein Seiteneffekt des Caching-Mechanismus betrifft die häufig benutzten Zähler für Zugriffe auf Seiten. Wenn noch eine aktuelle Version in dem Cache eines Proxys liegt, wird der Zugriff eines zweiten Benutzers lokal bedient, ohne dass der Webserver dies bemerkt. Durch diesen Effekt kann der Zugriffszähler einen zu niedrigen Wert enthalten.

11.2 E-Mail

Elektronische Post – E-Mail – ist eine einfache Anwendung. Die Aufgabe ist es, ein Dokument – etwa einen einfachen Text, ein Bild, ein Video oder eine Kombination mehrerer Elemente – von einem Sender zu einem Empfänger zu schicken. Auch hier findet sich wieder die Unterteilung in Client-Anwendung, Server-Anwendung, Anwendungsprotokoll und Begleitprotokoll. Im Folgenden sollen das Anwendungsprotokoll Simple Mail Transfer Protocol (SMTP) und das Protokoll Multi-Purpose Internet Mail Extension (MIME) zum Format der Nachricht näher betrachtet werden.

Auf Seiten des Clients ist die Situation etwas komplizierter als bei einem Webbrowser. Anders als bei WWW ist E-Mail eine asynchrone Anwendung. Neue Mails können zu beliebigen Zeiten eintreffen. Man benötigt daher einen Mechanismus, um die Anwendung für das Benutzer-Interface (Benutzeragent oder E-Mail-Reader) von der Zustellung zu entkoppeln. Dabei gibt es zwei unterschiedliche Strategien.

Falls eine permanente Verbindung zum Mail-Server (etwa einem Rechner in einem lokalen Netz) besteht, kann ein im Hintergrund laufender Prozess als Postamt arbeiten. Dieser Mail-Daemon nimmt Mails an und legt sie im Postfach des Benutzers ab. Gleichzeitig informiert er den Benutzer über die Ankunft einer neuen E-Mail. Der Benutzer kann dann mit seinem Mail-Reader die Nachricht lesen. Erhält der Mail-Daemon umgekehrt ausgehende Nachrichten, schickt er sie sofort an den Mail-Server.

Wenn allerdings nur temporär eine Verbindung zum Mail-Server vorhanden ist, benötigt man ein Protokoll, um gezielt Nachrichten abzuholen und abzugeben. Ein typischer Anwendungsfall dafür ist die manuelle Einwahl vieler Privathaushalte über ein Modem. Hier kommt meist das Mail-Fetching-Protokoll Post Office Protocol Version 3 (POP3) zum Einsatz.

Bei der E-Mail-Anwendung ist die Trennung zwischen Client und Server weniger deutlich als beim WWW. Letztlich stellen beide Seiten die gleichen Funktionalität – Empfangen und Senden von E-Mails – bereit. Insofern ergibt sich die Unterscheidung eher aus der jeweiligen Rolle aus Sicht des Benutzers.

11.2.1 Das Protokoll SMTP

Das Protokoll zum Austausch von E-Mails über das Internet ist das Simple Mail Transfer Protocol SMTP. Wie HTTP ist es ein textbasiertes Protokoll auf der Basis von TCP mit Anfragen und Antworten. Ein kleines Beispiel für das Versenden einer E-Mail sieht wie folgt aus (Eingaben sind zur besseren Lesbarkeit mit >> gekennzeichnet):

```
>> telnet monet 25
Trying 212.201.24.18...
Connected to monet.
Escape character is '^]'.
220 monet.fh-friedberg.de ESMTP
>> HELO monet.fh-friedberg.de
250 monet.fh-friedberg.de
>> MAIL FROM <stephan.euler@mnd.fh-friedberg.de>
250 ok
>> RCPT TO <stephan.euler@t-online.de>
553 sorry, that domain isn't in my list of allowed rcpthosts
➡ (#5.7.1)
>> RCPT TO <stephan.euler@mnd.fh-friedberg.de>
250 ok
>> DATA
354 go ahead
>> Hallo
>> Hier ist eine kleine Testmail.
>> .
250 ok 1024507580 qp 27898
>> QUIT
221 monet.fh-friedberg.de
```

Der Server bestätigt zunächst den erfolgreichen Verbindungsaufbau. Anschließend schickt der Client Anfragen mit den gewünschten Optionen. Der Server bestätigt die Anfragen mit einem dreistelligen Code und einem erläuternden Text. In dem Beispiel ist die erste Adresse nicht erlaubt, und die Eingabe führt zu einem Fehler. Nach der Option DATA folgt der Nachrichtentext, der durch eine Zeile mit nur einem Punkt beendet wird.

11.2.2 Nachrichtenformat und MIME

Zu Beginn der Entwicklung konnte man über elektronische Post lediglich einfache Textnachrichten verschicken. Eine Nachricht besteht aus zwei Teilen: einem Kopf (Header) und einem Rumpf (Body). Jede Kopfzeile enthält einen definierten Typ und einen dazugehörigen Wert, getrennt durch einen Doppelpunkt. Kopfzeilen werden mit der Kombination der Zeichen für Zeilenende CR und LF (kurz <CRLF>) abgeschlossen. Eine Leerzeile trennt den Kopf vom darauf folgenden Rumpf. Beispiele für Kopfzeilen sind:

```

Return-Path: <Manfred.Merkel@mnd.fh-friedberg.de>
Message-Id: <3C57CD5A.3020609@mnd.fh-friedberg.de>
Date: Wed, 30 Jan 2002 11:39:22 +0100
From: merkel <Manfred.Merkel@mnd.fh-friedberg.de>
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; ...
To: Stephan.Euler@mnd.fh-friedberg.de
Subject: [Fwd: Termin morgen]

```

Die Kopfzeilen enthalten unter anderem Informationen über Sender und Empfänger, Thema und Versanddatum. Gateways auf dem Weg der Nachricht zum Empfänger fügen weitere Informationen etwa über die Weiterleitung hinzu:

```

Received: from mx0.gmx.net by merkur.hrz.uni-giessen.de for
Stephan.Euler@mnd.fh-friedberg.de; Mon, 4 Mar 2002 11:34:21

```

In der Regel steht die eigentliche Nachricht im Rumpf. Lediglich in Spezialfällen kann die gesamte Information bereits im Betreff enthalten sein (beispielsweise bei subscribe, um sich für eine Mailing-Liste anzumelden).

Moderne E-Mail-Systeme erlauben den Versand von nahezu beliebigen Objekten. Der Text kann formatiert sein (zum Beispiel als HTML-Code), und Bilder, Musikstücke oder Ähnliches können als Anhang (Attachment) mitgeschickt werden. Als Standard zur Darstellung der unterschiedlichen Objekte dient die Multi-Purpose Internet Mail Extension (MIME). MIME umfasst dabei:

- Weitere Typen für die Kopfzeile wie etwa die MIME-Version.
- Definition von Inhaltstypen und -untertypen; zwei Beispiele sind: image/gif, text/richtext.
- Einen Typ Multipart, um mehrere Objekte zu einer gemeinsamen E-Mail zu verbinden. Das folgende Beispiel zeigt eine E-Mail mit einem kurzen Text und einer MS-Word-Datei als Anhang.

```

MIME-Version: 1.0
Content-Type: Multipart/Mixed;
  Boundary="__Next_1024570278_Part4__"

--__Next_1024570278_Part4__
Content-Type: Text/Plain;
  Charset="ISO-8859-1"
Content-Transfer-Encoding: Quoted-Printable

Text
--__Next_1024570278_Part4__
Content-Disposition: Attachment; filename="ueb1.doc"
Content-Type: application/msword;
  Name="ueb1.doc"
Content-Transfer-Encoding: Base64
0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAABAAALgAA
AAAAEAAAMAAAAAAEAAAD+///AAAAAC0AAAD+////////////////////////////////

```

Mime überträgt Dateien nicht binär, um eventuelle Probleme mit der unterschiedlichen Darstellung von Binärdaten auf den diversen Gateways zu vermeiden. In dem Beispiel wird eine Base64-Kodierung benutzt. Diese fasst je drei Bytes zu einer 24-Bit-Einheit zusammen. Daraus erzeugt sie vier ASCII-Zeichen mit je sechs Bit.

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4								
1. Byte									2. Byte									3. Byte													
1. Zeichen								2. Zeichen								3. Zeichen								4. Zeichen							

© tecCHANNEL

Bild 11.1: Je drei Bytes werden bei der Base64-Kodierung in vier Zeichen des einfachen ASCII-Zeichensatzes umgewandelt.

Die Darstellung beschränkt sich damit auf nur 64 Zeichen: 26 Buchstaben (jeweils klein und groß), zehn Ziffern und die beiden Sonderzeichen + und /. Dies bläht zwar das zu übertragende Datenvolumen um ein Drittel auf, verhindert aber zuverlässig Kompatibilitätsprobleme zwischen unterschiedlichen Gateways, E-Mail-Servern und Clients. Der E-Mail-Client des Empfängers findet bei Erhalt der Nachricht alle Informationen in den Kopfzeilen der E-Mail, um die ursprüngliche Datei wiederherzustellen.

11.2.3 E-Mail-Adressierung

Aus der E-Mail-Adresse extrahiert der Mail-Daemon die Zieladresse. Mittlerweile haben viele Adressen ein klares und leicht verständliches Format in der Art `Vorname.Name@Firma`. Der Teil nach dem `@` enthält den Namen des zuständigen Mail-Servers. Die zugehörige IP-Adresse kann der Mail-Daemon über eine DNS-Anfrage ermitteln.

Häufig werden E-Mails jedoch nicht direkt, sondern über spezielle Zwischenstationen (Gateways) geleitet. Diese Gateways übernehmen auch, falls erforderlich, eine Zwischenspeicherung. Denn anders als IP-Pakete sollen E-Mails nicht ohne Weiteres verworfen werden, nur weil durch einen Leitungsengpass die Weiterleitung momentan nicht möglich ist. Eine weitere wichtige Aufgabe von Gateways kann die Überprüfung von eingehenden E-Mails auf verdächtigen Inhalt wie Viren oder Spam sein.

Der eigentliche Zielknoten, also der Rechner, auf dem der Adressat seine E-Mails anschauen wird, ist in der Regel nicht bekannt. Zudem kann er sich von Fall zu Fall ändern. Daher wird die E-Mail zunächst nur bis zum Mail-Server geschickt.

Ist der Benutzer dort angemeldet und läuft auf seinem Client ein aktiver Mail-Daemon, so kann der Server die Nachricht direkt zustellen. Ansonsten informiert bei der nächsten Anmeldung des Benutzers der dann gestartete Mail-Daemon den

Server und holt damit die Post ab. Alternativ kann der Benutzer selbst aktiv die E-Mails abfragen. Unter Umständen muss der Mail-Server die Nachrichten über eine längere Zeit vorhalten, etwa wenn der Adressat in Urlaub ist.

11.3 Usenet

Neben der direkten Kommunikation über E-Mail besteht auch die Möglichkeit, Informationen in Diskussionsforen auszutauschen:

- Die Teilnehmer können Beiträge (Artikel; Englisch: news) schreiben und in dem Forum veröffentlichen.
- Andere Teilnehmer können auf diese Artikel antworten. Diese Antworten werden als neue Beiträge veröffentlicht, auf die die Forumsteilnehmer wiederum reagieren können. Dadurch kann sich die Diskussion auch in mehrere Äste aufspalten. Zur besseren Übersicht stellt die Anwendung alle Antworten als gemeinsamen Baum dar. Man spricht dann von Threads – dem englischen Wort für Faden oder Zwirn. Dementsprechend markiert der Newsreader von Mozilla solche Diskussionen mit dem Symbol einer Zwirnrolle.

Diese Diskussionsforen (Newsgroups) sind auf speziellen News-Servern abgelegt. Einen Verbund vieler News-Server bildet das USENET (ursprünglich Unix User Network). Die daran beteiligten Server tauschen untereinander ständig die neuen Artikel aus (Replikation), so dass ein einheitlicher Stand gewährleistet ist.

Allerdings kann ein Server schon allein aus Speicherplatzgründen nicht alle Newsgroups bereitstellen, sondern immer nur einen Teilbereich abdecken. Außerdem ist es auch möglich, mit der gleichen Technologie private oder interne News-Server zu betreiben.

Die Vielzahl von Diskussionsgruppen ist hierarchisch unterteilt. Die einzelnen Namensbestandteile sind durch Punkte getrennt, und es gilt die Regel „Vom Allgemeinen zum Speziellen“. Die ersten Kürzel spezifiziert das Hauptthema oder ein Land. Einige Beispiele sind:

- comp.speech.research – ein Forum zur Erforschung der Sprachverarbeitung mit Computern.
- de.sci.informatik.ki – eine deutsche Gruppe zum Thema „Künstliche Intelligenz“.
- rec.games.majong – alles über das Spiel Majong (das Kürzel rec steht für recreation; Erholung).

Newsgroups findet man über spezielle Suchdienste wie etwa findolin (<http://www.findolin.com>). Die allermeisten Newsgroups sind unmoderiert. Allerdings wird an die Teilnehmer appelliert, bestimmte Regeln einzuhalten. Man sollte sich klar machen, dass ein eigener Beitrag von Millionen von Menschen gelesen werden könnte. Daher sind Klarheit und Sorgfalt notwendig, um eventuellen Missverständnissen vorzubeugen. Allzu schnell wird ein Beitrag falsch verstanden und

führt zu einem erbitterten Austausch von News. In der RFC 1855 „Netiquette Guidelines“ sind allgemeine Regeln für Newsgroups und andere Kommunikationsdienste zusammengestellt.

Um an Newsgroups teilzunehmen, benötigt man einen entsprechenden Newsreader. Dazu stehen spezielle Programme zur Verfügung. Alternativ bieten die meisten E-Mail-Programme die entsprechende Funktionalität. Man muss nur noch einen passenden News-Server auswählen. Das kann der Server eines Netzanbieters sein (etwa news.t-online.de bei T-Online). Daneben gibt es kommerzielle oder frei verfügbare Server wie den www.individual.de der Freien Universität Berlin. Der komplette Verweis auf eine Gruppe hat dann die Form

```
news://news.t-online.de/de.comm.internet.misc
```

Der Austausch von Artikeln erfolgt über das Network News Transfer Protocol (NNTP). Dieses Protokoll spezifiziert eine Anzahl von Befehlen, um über eine stromorientierte Verbindung Informationen und die Beiträge zu übertragen. Es umfasst sowohl die Kommunikation zwischen News-Server und Newsreader als auch den Abgleich zwischen zwei News-Servern. Für die Nachrichten wird die gleiche Formatierung wie für E-Mails verwendet. Entsprechende Kopfeinträge übermitteln die erforderlichen Informationen. So wird im Eintrag Newsgroups: festgelegt, zu welcher Gruppe oder zu welchen Gruppen ein Beitrag gehört. Der Rumpf der Nachricht wird mittels MIME kodiert.

11.4 Netzwerk-Management

In der Diskussion der verschiedenen Protokollebenen hat sich bereits gezeigt, dass deren Designer Wert auf ein weit gehend selbstständiges, unüberwachtes Funktionieren des Netzes gelegt haben. Die Protokolle enthalten Mechanismen, um mit auftretenden Fehlern umgehen zu können. Die Knoten passen ihr Verhalten teilweise an die aktuelle Situation an. Trotzdem benötigt man Möglichkeiten, das Verhalten eines Netzes im Detail untersuchen zu können, um Fehler oder Schwachstellen aufzuspüren.

Ein dazu häufig verwendetes Protokoll ist das Simple Network Management Protocol (SNMP). SNMP erlaubt es, zahlreiche Parameter von Knoten im Netz abzufragen. Jeder Knoten, der an diesem System teilnimmt, stellt dazu einen SNMP-Server bereit. Von anderen Systemen aus kann man dann mit einem Client die Statusinformationen abfragen. SNMP realisiert dazu ein Abfrage-Antwort-Protokoll auf der Basis von UDP. Die beiden wesentlichen Operationen sind GET, um Werte zu lesen und SET, um Werte zu setzen.

Der Netzwerkknoten sammelt im laufenden Betrieb ständig Informationen und legt sie in einer internen Datenbank, der Management Information Base (MIB), ab. Über ein spezielles Identifizierungssystem kann man per SMTP jede dieser Variablen abfragen. SNMP verwendet dazu eine standardisierte Darstellung zur Übermittlung von Datentypen wie Integer-Zahlen oder Gleitkommazahlen.

11.5 Multimedia-Kommunikation

Mit seiner ständig wachsenden Ausdehnung und Verfügbarkeit ist das Internet auch als Plattform für Multimedia-Anwendungen interessant. Unter Multimedia im eigentlichen Sinne versteht man die Integration verschiedener Medien in einer Anwendung oder in einem Dokument. Dabei können die Medien zeitunabhängig (Text, Grafik, Bild) oder zeitabhängig (Audio, Video) sein.

Weiterhin lassen sich die Anwendungen in zwei Klassen aufteilen:

- Anwendungen zwischen zwei oder mehr Benutzern. Diese bezeichnet man auch als Konferenzanwendungen.
- Anwendungen zwischen einem Server und einem oder mehreren Clients (Streaming-Anwendungen)

Zeitabhängige Multimedia-Anwendungen stellen hohe Anforderungen an die Übertragung. So benötigt etwa eine Video-Konferenz eine große Bandbreite bei gleichzeitig geringer Latenz. Die Synchronisation verschiedener Datenströme (etwa bei getrennten Audio- und Video-Kanälen) erfordert zusätzliche Maßnahmen zur Zusammenführung und Ablaufsteuerung. Hierbei kann es beispielsweise erforderlich sein, bei Leitungsengpässen das Video-Bild einzufrieren und nur noch den Ton zu übertragen, da dieser die wichtigere Informationsquelle für den Kommunikationspartner ist.

11.5.1 Real-Time Transport Protocol

Ein universelles Transportprotokoll für Multimedia-Daten ist das Real-Time Transport Protocol (RTP). Es soll die notwendigen Eigenschaften für eine Echtzeitkommunikation bereitstellen, dabei aber so wenige Festlegungen wie möglich enthalten. So bleibt es etwa der Anwendung selbst überlassen, wie sie mit Paketverlusten umgeht.

RTP setzt auf UDP auf. UDP bietet die notwendige Grundfunktionalität – Zustellung von Paketen ohne großen Overhead. RTP ergänzt diesen Paketdienst um einige für die Echtzeit-Kommunikation notwendige Elemente.

Bild 11.2: Der Protokoll-Stack von RTP setzt auf UDP auf.

Anwendung
RTP
UDP
IP

Der Header für RTP-Pakete ist mindestens 12 Byte lang. Er enthält eine 16 Bit große Sequenznummer für das Paket. Diese Nummer wird bei jedem Paket um Eins erhöht. Anhand dieser Nummern kann der Empfänger erkennen, ob er alle

Pakete in der richtigen Reihenfolge erhält. Die zeitliche Beziehung stellt ein Feld mit einem Zeitstempel her. Dabei ist das exakte Format des Zeitstempels abhängig von der Anwendung.

Die Quelle des Datenstroms wird über eine 32-Bit-Zahl im Feld Synchronization Source (SSRC) mitgeteilt. Über ein 7 Bit großes Feld wird der Typ der Nutzdaten angegeben. Damit könnte beispielsweise ein Wechsel der Kodierungsmethode angezeigt werden. Schließlich kann ein RTP-Strom Beiträge von mehreren Quellen enthalten (zum Beispiel mehrere Mikrofonkanäle).

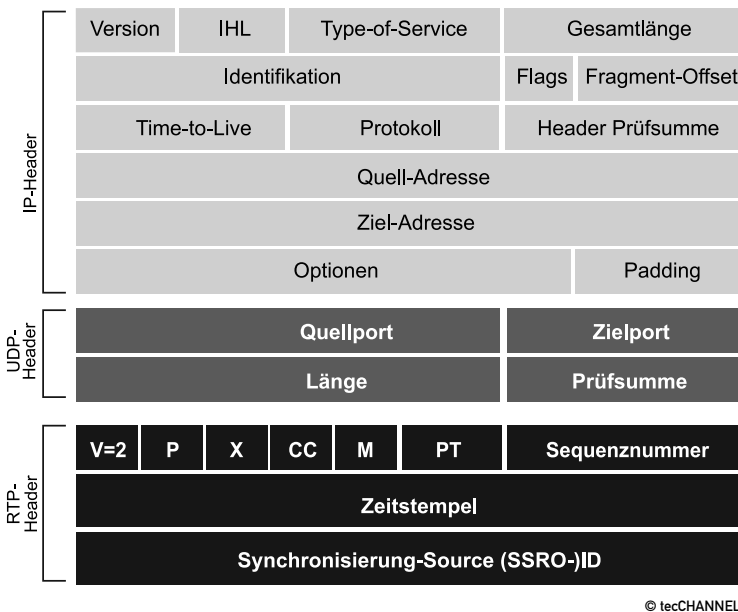


Bild 11.3: RTP regelt den Transport der Echtzeitdaten über einen eigenen Header.

Die Bedeutung der einzelnen Datenfelder im RTP-Header erläutert die Tabelle auf der folgenden Seite.

RTP ist optimiert für die schnelle Übertragung. Das Protokoll enthält keinerlei Elemente zum Umgang mit verlorenen Paketen oder etwa zur Überlastkontrolle. Allerdings ist es wünschenswert, dass der Empfänger über den aktuellen Zustand der Verbindung informiert wird. Beispielsweise könnte ein Sender bei einem Bandbreitenengpass zu einer Kodierung mit niedriger Datenrate wechseln oder unwichtigere Elemente nicht mehr übertragen.

Tabelle 11.3: RTP – Header-Daten

	Bits	Bezeichnung	Bedeutung
V	2	version	RTP-Versionsnummer
P	1	padding	Gibt an, ob der Frame mit Padding-Bytes aufgefüllt werden soll
X	1	extension	Gibt an, ob eine Header-Erweiterung vorliegt
CC	4	CSRC count	Anzahl der CSRC-Identifizier
M	1	marker	Markiert ein bestimmtes Profil
PT	7	payload type	Legt das Format des RTP-Pakets fest und ordnet es einer Anwendung zu
SN	16	sequence number	Dient der Sortierung der empfangenen Pakete
-	32	timestamp	Absenzeit (in Relation zum RTP-Strom)
SSRC	32	sync source	Zufallszahl zur Unterscheidung von verschiedenen RTP-Quellen
CSRC	32	contributing source	Identifikation der beitragenden Quelle (nur bei gemischten RTP-Strömen)

Zur Überwachung und Steuerung einer RTP-Sitzung dient das Protokoll RTP Control Protocol (RTCP). Eine RTP-Sitzung besteht aus einem RTP- und einem RTCP-Kanal. Für eine Sitzung wird für RTP eine gerade Portnummer und für RTCP die darauf folgende ungerade Portnummer vergeben.

Über RTCP tauschen die Kommunikationspartner parallel zu RTP Pakete aus. Dabei kommen in der Regel ebenfalls UDP-Pakete zum Einsatz. Diese Pakete beinhalten Sende- oder Empfangsberichte mit Übertragungsstatistiken. Die Statusberichte tauschen die Partner in periodischen Abständen aus.

Informationen über den Sender werden mit Quellenbeschreibungspaketen übermittelt. Als Kennung dient der so genannte kanonische Name (CNAME). Das übliche Format dafür ist „user@host“. Die Quellenbeschreibungspakete enthalten die Zuordnung zwischen SSRC und CNAME. Mit dieser Information kann der Empfänger die Daten dem Absender zuordnen. Damit kann er auch mehrere Quellen (etwa Audio, Video, Grafik) von einem gemeinsamen Sender unterscheiden.

Schließlich bietet RTCP die Möglichkeit, ergänzende Informationen parallel zu dem Datenstrom über RTP zu senden. Ein Beispiel hierzu sind Untertitel für einen Video-Strom. Trotz all dieser Funktionen streben beide Kommunikationspartner an, den Verkehr über RTCP auf etwa fünf Prozent des Datenaufkommens für den RTP-Verkehr zu beschränken.

11.5.2 RTP-Verbindungsaufbau

Vor einer Kommunikation über RTP muss eine Sitzung eingeleitet werden. Je nach Anwendung reicht diese Aufgabe von der einfachen Ankündigung einer Übertragung bis hin zum aufwendigen Aufbau einer Video-Konferenz. Dabei muss der Initiator zunächst die Gesprächspartner suchen und dann ein gemeinsames Kodierungsverfahren aushandeln.

Eine Arbeitsgruppe der Internet Engineering Task Force (IETF) hat dafür eine Reihe von Protokollen entwickelt. Um beispielsweise eine Telefonverbindung über Internet herzustellen, sind die Protokolle Session Initiation Protocol (SIP) und Session Description Protocol (SDP) gebräuchlich.

Alternativ entwickelte ITU (International Telecommunication Union) die Empfehlung H.323 für den Gesprächsaufbau. Die Eigenschaften des Gesprächs werden dabei über das Call-Control-Protokoll H.245 bestimmt. Geräte, die über H.323 Verbindungen aufbauen, werden auch als H.323-Terminals bezeichnet.

Während der ITU-Standard H.323 vor allem im Umfeld professioneller TK-Anlagen zum Einsatz kommt, gewinnt das von der IETF festgelegte SIP in öffentlichen Netzen und bei Privathaushalten zunehmend an Bedeutung.

11.5.3 VoIP – Sprachübertragung über IP

Eine Anwendung von großer kommerzieller Bedeutung ist die Sprachübertragung für Telefongespräche. Daher begannen verschiedene Firmen frühzeitig, Lösungen zur Sprachübertragung über IP (Voice over IP, VoIP) zu entwickeln. Die paketorientierte Technik verspricht einen deutlichen Kostenvorteil gegenüber dem leitungsorientierten herkömmlichen Telefonsystem.

Zum einen entfällt die Installation eines eigenen Netzwerkes nur für den Telefonverkehr. Zum anderen ist die Übertragung innerhalb des Internets zumindest bei Fernverbindungen wesentlich preisgünstiger möglich. Allerdings erwartet der Kunde einen vergleichbaren Stand bezüglich

- einfacher Bedienung
- guter Sprachqualität
- hoher Zuverlässigkeit

Im Folgenden werden die Grundprinzipien als Beispiel für eine Echtzeitkommunikation über das Internet beschrieben. Eine ausführliche Darstellung findet man beispielsweise in [18] und [19].

11.5.4 Sprachkodierung

Das analoge Telefonnetz überträgt die Sprachsignale in dem Frequenzbereich von 300 bis 3300 Hz (Telefonbandbreite). Ausgehend von diesem Frequenzband wurde bei der Digitalisierung eine Abtastrate von 8000 Hz gewählt. Gemäß dem Nyquist-Kriterium (Maximalfrequenz = halbe Abtastrate) sind so Frequenzen bis

4000 Hz enthalten. Die Pulsmodulation (PCM) kodiert dabei jeden Abtastwert als Zahlen. Der Wertebereich dieser Zahlen bestimmt den Signal-Rauschabstand der Übertragung und damit deren Qualität.

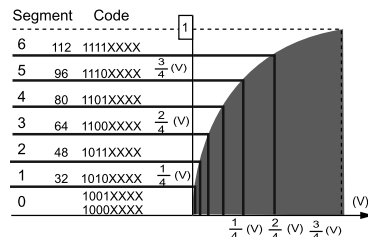
Für Gespräche im Telefonnetz nutzt man physiologische Schwächen des menschlichen Gehörs aus. Daher ist nicht über den gesamten Amplitudenbereich eine gleichmäßige Quantisierung notwendig. Für große Amplituden kann man eine gröbere Rasterung verwenden, ohne dass dadurch eine hörbare Verschlechterung der Sprachqualität auftritt.

Auf diese Weise ist es möglich, mit nur 8 Bit Auflösung eine gute Qualität zu erreichen. Die Kombination aus 8 kHz Abtastrate und 8 Bit Auflösung ergibt eine nötige Bandbreite von 64 Kbit/s – den Standard in ISDN.

A-Law-Kurve
(Europäisches E1-System):

$$F_A(x) = \text{sgn}(x) \cdot \left[\frac{1 + \ln(A \cdot |x|)}{1 + \ln(A)} \right] \quad \text{for } \frac{1}{A} \leq |x| \leq 1$$

$$F_A(x) = \text{sgn}(x) \cdot \left[\frac{1 + A \cdot |x|}{1 + \ln(A)} \right] \quad \text{for } 0 \leq |x| \leq \frac{1}{A}$$



μ-Law-Kurve
(Nordamerikanisches DS1 (T1)-System):

$$F_\mu(x) = \text{sgn}(x) \cdot \left[\frac{\ln(1 + \mu \cdot |x|)}{\ln(1 + \mu)} \right]$$

© tecCHANNEL

Bild 11.4: Die nicht lineare Quantisierungskurve verringert das so genannte Quantisierungsrauschen, da sie den relativen Fehler bei kleinen Abtastwerten reduziert.

Die Datenrate lässt sich weiter verringern, wenn man spezielle Eigenschaften der Sprachsignale ausnutzt. Ein Ansatz dazu beruht auf der Annahme, dass die Amplituden aufeinander folgender Abtastwerte in der Regel nicht beliebig weit auseinander liegen. Überträgt man anstelle der Abtastwerte nur die Differenzen benachbarter Werte, so kann man den dann auftretenden kleineren Wertebereich mit weniger Bit quantisieren.

Bei adaptiven Verfahren wird der Wertebereich zusätzlich an die aktuelle Lautstärke angepasst. Dadurch erreicht man eine möglichst gute Übereinstimmung zwischen Wertebereich und auftretenden Amplituden. Ein Standard nach diesem Prinzip ist der „Adaptive Differenzielle PCM“ (ADPCM) mit 32 Kbit/s gemäß ITU G.726. Gegenüber dem PCM-Standard wird bei nahezu gleicher Sprachqualität nur die halbe Datenrate benötigt. Ein Beispiel für den Einsatz dieses Standards ist DECT (Digital European Cordless Telephony) für schnurlose Telefone.

Sprachsignale enthalten noch sehr viel mehr Strukturen. Eine Reihe von Verfahren, die diese Strukturen weit gehend ausnutzen, beruhen auf dem Basisprinzip von Code Excited Linear Predictive Coding (CELP). Diese Verfahren benutzen spezielle adaptive Filter – so genannte lineare Prädiktoren – zur Vorhersage der nächsten Abtastwerte. Das Anregungssignal wird durch systematisches Probieren von Codes aus einem Vorrat ermittelt [20]. Vertreter dieser Coder-Familie setzen die GSM-Netze für Handytelefonate ein.

Der so genannte CELP Full Rate Coder arbeitet mit 12,2 Kbit/s. Die neue Nachfolgetechnik Half Rate benötigt bei annähernd gleicher Sprachqualität nur noch 5,6 Kbit/s. Für VoIP weit verbreitet sind die Standards G.729A (CompuServe ACELP) mit 8 Kbit/s und G.723.1 Multi Rate Coder mit 5,3 und 6,3 Kbit/s. Ein neuerer Standard speziell für das Internet ist der Internet Low Bit Rate Codec (iLBC). Dieser Coder arbeitet bei 13,3 oder 15,2 Kbit/s. Er wurde speziell für Situationen, in denen Pakete verloren gehen, ausgelegt. In solchen Fällen wird mittels einer so genannten Packet-Loss-Concealment-Einheit für eine Überbrückung der fehlenden Blöcke ohne Störgeräusche gesorgt.

Dialoge enthalten einen Anteil von bis zu 50 Prozent Pausen. Eine weitere Datenreduktion ist daher möglich, wenn der Coder die Sprachpausen nicht mit übertragen muss. Erkennt er, dass ein Teilnehmer im Moment nicht spricht, überträgt er keine Blöcke, die nur Hintergrundgeräusche enthalten. Um den Eindruck einer „toten Leitung“ zu vermeiden, kann die Gegenseite ein künstliches Rauschen einspielen. Damit ist eine deutliche Reduktion der Datenkommunikation möglich. Allerdings können Fehlentscheidungen in der Pause-Detektion oder ein zu spätes Ansprechen auf eine beginnende Sprachaktivität zu einem sehr unnatürlichen Spracheindruck führen.

Bei sämtlichen Codecs muss man zur Berechnung der Bandbreite zu den Sprachdaten noch die Paket-Header hinzuaddieren. Die unten stehende Tabelle zeigt die entsprechenden Werte bei einem Coder mit 6 Kbit/s und einer Dauer von 30 ms für einen Sprachblock.

Tabelle 11.4 Größe eines IP-Pakets mit Sprachdaten	
Kodiertrate	6 Kbit/s
Blocklänge	30 ms
Datenbits pro Block	180 Bit
RTP-Nutzlast	23 Byte
RTP-Header	12 Byte
UDP-Header	8 Byte
IP-Header	20 Byte
Summe	63 Byte

Bei der Übertragung über Ethernet kommen weitere 29 Byte (inklusive 3 Byte LLC-Header) pro Paket hinzu. Insgesamt ergeben sich somit 92 Byte pro Paket in dem 23 Bytes Sprachinformationen enthalten sind. Mit 33,3 Paketen pro Sekunde resultiert daraus eine Bandbreite im Netzwerk von 24,5 Kbit/s pro Richtung und 49 Kbit/s für die Telefonverbindung.

Die Zusammenstellung zeigt, dass die verschiedenen Header in Summe gegenüber der Nutzlast dominieren. Daher wurden Protokoll-Erweiterungen entwickelt, um die Länge der Header zu reduzieren. Ein Ansatzpunkt ist, einen Teil der Header-Informationen nur zu Beginn der Verbindung zu übertragen. Während der eigentlichen Datenübertragung ändern sich diese Informationen entweder gar nicht mehr oder nur geringfügig. Indem das Protokoll nur noch die Veränderungen überträgt, kann es die Länge der Header deutlich reduzieren.

Ein Problem besteht allerdings darin, dass einzelne Pakete verloren gehen können. Ohne weitere Maßnahmen würde dann unter Umständen ein Teil der Veränderungen fehlen, so dass der Empfänger fehlerhafte Werte rekonstruiert. Speziell für dieses Einsatzgebiet ist das Protokoll RObust Header Compression (ROHC), RFC 3095, definiert. Nach [21] lässt sich mit ROHC eine Reduktion der Header von RTP, UDP und IP mit zusammen 20 Byte auf durchschnittlich 6 Byte erzielen, ohne die Sprachqualität zu beeinträchtigen.

Ein zweites wichtiges Kriterium für die Verbindungsqualität ist die Verzögerungszeit. Ab etwa 100 ms machen sich Verzögerungen im Gespräch störend bemerkbar. Laut ITU-T-Empfehlung gelten Verzögerungen bis 200 ms noch als gut und bis 400 ms gerade noch als akzeptabel. Die Sprachkodierung führt durch die Blockbildung bei den niedrigen Datenraten zu etwa 10 bis 30 ms Verzögerung. Hinzu kommt die Latenz der Verbindung.

Innerhalb eines Firmennetzes ist die resultierende Gesamtverzögerung in der Regel unterhalb der kritischen Grenzen. Bei einer Übertragung im Internet können sich allerdings durch die zahlreichen Zwischenstationen und Paketverluste deutliche Qualitätseinbußen ergeben.

11.5.5 Telefonanwendungen

Mit den beschriebenen Mitteln kann ein Rechner die Funktion eines Telefons übernehmen. Dazu genügen ein PC, eine handelsüblichen Sound-Karte, ein Mikrofon und ein Lautsprecher sowie eine entsprechende Software. Eine einfache Anwendung ist das Telefonieren von Rechner zu Rechner.

Die entsprechenden Programme ähneln Chat-Programmen. Über einen Server kann man Verbindung zu den angemeldeten Personen aufnehmen. In der Regel können sich weitere Personen an dem Gespräch beteiligen, so dass eine Konferenz entsteht. Viele Programme unterstützen dabei die parallele Kommunikation über mehrere Medien. So sind in das Programm NetMeeting von Microsoft neben der Sprachkommunikation auch noch Video und Whiteboard – ein gemeinsam genutztes Zeichentablett – integriert.

Neben der Kommunikation zwischen Rechnern ist auch die Verbindung zum öffentlichen Telefonnetz möglich. So bieten diverse Anbieter Gateways zwischen Internet und Telefonnetz an. Eine kostengünstige Verbindung besteht dann aus einer Internet-Verbindung bis zu einem Gateway in der Nähe des Ziels und einer Telefonverbindung vom Gateway bis zum Endteilnehmer. Das Gateway übernimmt die Umsetzung der Daten sowie der Signalisierungsinformationen. Bei H.323 ist ein so genannter Gatekeeper bei der Suche nach dem günstigsten Gateway behilflich.

Das Konzept lässt sich leicht zum Telefonieren zwischen zwei konventionellen Telefonen erweitern. Ein Teilnehmer ruft bei seinem lokalen Gateway an. Von dort wird eine RTP-Sitzung zu dem passenden Gateway in der Nähe des Ziels aufgebaut. Dieses leitet das Gespräch dann wieder an das konventionelle Telefon des Gesprächspartners weiter.

Prof. Dr. Stephan Euler

tecCHANNEL-Links zum Thema	Webcode	Compact
Netzwerkanwendungen	a1501	S.110
Hypertext Transfer Protocol	a208	–
So funktioniert E-Mail	a819	–
Voice over IP	a990	–

Mit den Webcodes gelangen Sie auf www.tecChannel.de direkt zum gewünschten Artikel. Geben Sie dazu den Code in das Feld „Webcode“ in der Titelleiste von www.tecChannel.de ein. Mit „p“ gekennzeichnete Webcodes beziehen sich auf Online-Artikel aus tecCHANNEL-Compact. Es ist möglich, dass Artikel zunächst exklusiv in tecCHANNEL-Compact veröffentlicht werden und daher erst später über den Webcode erreichbar sind.

12. Java-Einsatz im Netzwerk

In den Bibliotheken von Java findet sich eine ganze Reihe von Klassen zur Unterstützung von Anwendungen über Netzwerke und insbesondere im Internet. Die Klassen bieten ein unterschiedliches Einstiegsniveau. Einerseits besteht die Möglichkeit, mittels Sockets auf einer relativ niedrigen Ebene zu arbeiten. Andererseits stehen auch Klassen wie URL zur Verfügung, die bereits einen Großteil der Netzwerkfunktionalität beinhalten. Im Folgenden werden exemplarisch einige Möglichkeiten der beiden Klassen Socket und URL vorgestellt.

12.1 Java Client-Sockets

In der Klassenbibliothek von Java stehen mehrere nach Funktionalitäten getrennte Klassen für Sockets zur Verfügung. Die Klasse Socket realisiert einen Client-Socket. Verschiedene Konstruktoren ermöglichen die Angabe des Servers über den Namen oder die IP-Adresse. Folgender Code erzeugt auf einfache Weise einen Socket:

```
Socket s = new Socket( "www.fh-friedberg.de", 80 );
```

An den so erzeugten Socket können mit folgenden Methoden Ein- und Ausgangsströme angehängt werden

```
getInputStream()  
getOutputStream()
```

Eine Anwendung kann dann diese Ströme in der gleichen Art und Weise wie Ströme von und zu Dateien benutzen. Der Socket übernimmt dann die Details der Netzwerkkommunikation.

Als Beispiel folgt ein Programm, das über eine Socket-Verbindung eine E-Mail mit einer Klausurnote verschickt. Zunächst öffnet das Programm eine Socket-Verbindung zu einem Mail-Server an dessen Port 25. Anschließend sendet es die Befehle gemäß SMTP. Zur besseren Übersicht verzichtet das Beispiel auf eine Erfolgskontrolle. Dazu müssten über einen InputStream die zurückgeschickten SMTP-Statuscodes gelesen und ausgewertet werden.

Das folgende sowie alle anderen Listing in diesem Compact können Sie online von www.tecChannel.de unter dem Webcode a1541 bequem herunterladen.

```
import java.net.*;  
import java.io.*;  
  
public class TestEmail {  
  
    public static void main( String args[]) {  
        String host    = "mailto.t-online.de";  

```

```
String from    = "stephan.euler@t-online.de";
String to      = "stephan.euler@t-online.de";
String betreff = "Note der Informatikklausur III";
String inhalt  = "Sie haben die Note 3 \n";
try
{
    Socket mailSocket = new Socket(host, 25 );
    PrintWriter out = new PrintWriter(
        mailSocket.getOutputStream(), true );
    out.println( "HELO " + host);
    out.println( "MAIL FROM: <" + from + ">");
    out.println( "RCPT TO: <" + to + ">");
    out.println( "DATA" );
    out.println( "SUBJECT: " + betreff );
    out.println( inhalt );
    out.println( ".");
    out.println( "QUIT" );
    out.close();
    mailSocket.close();
}
catch( IOException e ) { System.err.println( e ); }
}
```

12.2 Java Server-Sockets

Einen Server für verbindungsorientierte Sockets realisiert die Klasse `ServerSocket`. Die Klasse enthält eine Reihe von Konstruktoren und Methoden, um einen `ServerSocket` zu erzeugen und anzumelden. Eine kompakte Form ist:

```
ServerSocket socket = new ServerSocket(port, backlog);
```

Die Parameter bestimmen die Portnummer `port` und die Größe `backlog` der Warteschlange für anstehende Clients. Nach dem Aufruf der Methode `accept()` wartet der `Socket` auf einen Client. Meldet sich ein Client an, so kehrt die Methode mit einem normalen `Socket` als Rückgabewert zurück.

Mit diesen Elementen ist in der folgenden Klasse `SocketServer` ein einfacher Server realisiert. Der Server wartet in einer Endlosschleife auf Anfragen. Ein Client erhält bei der Kontaktaufnahme eine kurze Begrüßung. Anschließend wird die Verbindung wieder abgebaut.

```
import java.net.*;
import java.io.*;

public class SocketServer {
    static int backlog = 10; // Laenge der Warteschlange
    static int port    = 1234;
```

```

public static void main( String args[] ) {

    try
    {
        ServerSocket socket =
            new ServerSocket(port, backlog);
        for( ;; ) {
            System.out.println( "Warte auf Verbindung... " );
            Socket sockConnected = socket.accept();
            System.out.println( "Verbunden mit "
                               + sockConnected);
            PrintStream ps = new PrintStream(
                sockConnected.getOutputStream() );
            ps.println( "Hallo" );
            sockConnected.close();
        }

    }
    catch( IOException e ) { System.err.println( e ); }
}

```

12.3 Java UDP-Sockets

Der beschriebene Client-Socket verwendet eine stromorientierte Verbindung (TCP). Paketorientierte Verbindungen (UDP) realisiert die Klasse `DatagramSocket`. So erzeugt folgende Zeile einen neuen Socket für die Datagramm-Kommunikation über Port 8888.

```
DatagramSocket s = new DatagramSocket(8888);
```

Der Datenaustausch erfolgt dann mit Objekten der Klasse `DatagramPacket`. Zum Versenden werden in diese Pakete die Daten und die Zieladressen eingetragen.

Das Zusammenspiel der Klassen soll das Beispiel eines Chat-Servers darstellen. Der Server hat folgende Funktionalitäten:

- Anmeldung von Clients
- Nachricht eines Clients an alle anderen Clients weiterleiten
- Abmeldung von Clients

An- und Abmeldung erfolgen über Pakete, die eine entsprechende Kennung enthalten. Der Server führt eine Liste mit allen aktuell angemeldeten Clients. Erhält er von einem der Clients eine Nachricht, so leitet er eine Kopie davon an alle anderen Clients weiter.

Der folgende Code realisiert einen solchen Server. Die angemeldeten Clients werden in einem Objekt der Klasse `Vector` gespeichert. Die Fehlerprüfung beschränkt sich auf die Ausgabe von eventuell auftretenden Ausnahmefehlern (Exceptions).

Einige der benutzten Methoden sind erst ab Java 1.4 vorhanden. Diese können aber im Bedarfsfall durch ältere, weniger elegante Methoden ersetzt werden.

```
import java.net.*;
import java.io.*;
import java.util.*;

public class DatagrammServer {
    // Kennungen für An- und Abmeldung
    static final String ANMELDUNG = "ANMELDUNG";
    static final String ENDE      = "ENDE";
    static int port      = 1234;
    static int length    = 256;    // Länge eines Pakets

    public static void main( String args[] ) {

        DatagramPacket paket =
            new DatagramPacket( new byte[length], length);
        Vector clients = new Vector();    // Liste der Clients

        try {
            DatagramSocket socket = new DatagramSocket(port);
            for( ;; ) {

                // Warten auf nächstes Paket
                socket.receive( paket );
                InetSocketAddress add =
                    (InetSocketAddress)paket.getSocketAddress();

                // Text aus Paket extrahieren
                String text =
                    new String(paket.getData(), 0,
➡paket.getLength());
                System.out.println( add + ">" + text);

                // Paket auswerten
                if( text.equals( ANMELDUNG ) ) {
                    clients.add( add );
                    System.out.println( "Anzahl Clients: "
                                         + clients.size() );
                } else if( text.equals( ENDE ) ){
                    clients.remove( add );
                    System.out.println( "Anzahl Clients: "
                                         + clients.size() );
                } else {
➡Clients
                    // Versenden von Kopien an alle anderen

                    for( int i=0; i<clients.size(); i++ ) {
                        InetSocketAddress dest =
                            (InetSocketAddress) clients.get(i);
```

Als passendes Gegenstück dient der folgende Client. Bei seinem Aufruf kann der Name des Servers als Argument übergeben werden. Fehlt diese Angabe, sucht der Client auf dem eigenen Rechner nach dem Server.

Anders als bei dem Server erfolgt der Nachrichtenaustausch asynchron. Der Server wartet auf eine Nachricht und reagiert darauf. Demgegenüber kann beim Client zu jedem Zeitpunkt vom Server eine neue Nachricht eintreffen. Es gibt keine zeitliche Koppelung zwischen Eingaben und Ausgaben.

Die Hauptanwendung mit der Schleife für Benutzereingaben und dem Senden an den Server läuft in einem Thread. Die einkommenden Nachrichten behandelt ein eigener Verarbeitungsstrang in einem zweiten Thread. Beide Threads laufen dabei unabhängig voneinander parallel ab.

```
import java.net.*;
import java.io.*;

public class DatagramClient {
    static final String ANMELDUNG = "ANMELDUNG";
    static final String ENDE      = "ENDE";
    static int port      = 1234;
    static int length    = 256;    // Länge eines Pakets

    public static void main( String args[] ) {
        String servername = "localhost";
        String text = null;
        DatagramPacket packet;
        byte[] ba = ANMELDUNG.getBytes();

        // Namen des Servers von Kommandozeile übernehmen
        if( args.length > 0 ) servername = args[0];

        try {
            DatagramSocket socket = new DatagramSocket();
            InetAddress ia = InetAddress.getByName(
➤servername );
            packet = new DatagramPacket( ba, ba.length, ia,
➤port);
            // sende Anmeldung
            socket.send( packet );

            // Lesen der empfangenen Pakete erfolgt in eigenem
➤Thread
            LeseThread lt = new LeseThread( socket );

            // Eingaben von Tastatur an Server schicken
            BufferedReader br = new BufferedReader(
                new InputStreamReader( System.in ) );
            do {
                text = br.readLine();
                ba = text.getBytes();
                packet.setData( ba, 0, ba.length );
                socket.send( packet );
            } while( ! text.equals("ENDE") );

            // alles beenden
            System.exit(0);

        }
        catch( IOException e ) {
            System.err.println("Ausnahmefehler: " + e );
        }
    }
}
```

Dazu gehört die folgende Klasse LeseThread zum Empfang von Nachrichten:

```
class LeseThread implements Runnable {
    static int length = 256;
    DatagramSocket socket;

    LeseThread(DatagramSocket socket ) {
        this.socket = socket;
        Thread t = new Thread(this, "Lesen");
        t.start();
    }

    public void run() {
        DatagramPacket packet =
            new DatagramPacket( new byte[length], length);
        while( true ) {
            try {
                socket.receive( packet );
                InetSocketAddress add =
                    (InetSocketAddress)packet.getSocketAddress();
                String text =
                    new String(packet.getData(), 0,
➡ packet.getLength());
                System.out.println( add + ">" + text);
                //System.out.println( ">" + text);
            }

            catch( IOException e ) {
                System.err.println("Ausnahmefehler: " + e );
            }
        }
    }
}
```

12.4 Die Klasse URL

Die Klassen rund um Sockets behandeln die Netzwerkkommunikation auf einer relativ niedrigen Ebene. In Java existieren jedoch auch zahlreiche Möglichkeiten für eine Kommunikation auf höherer Ebene. Eine solche Möglichkeit besteht über die Klasse URL. Ein URL-Objekt erzeugt man im einfachsten Fall durch den Konstruktor mit dem Namen des Links als Parameter.

```
URL url = new URL("http://www.fh-friedberg.de");
```

Das URL-Objekt beinhaltet zunächst nur die Information über den Link. In einem weiteren Schritt kann man darüber eine Verbindung aufbauen. Eine Möglichkeit dazu bietet die Methode `openConnection()`. Sie liefert ein Objekt der Klasse `URLConnection`, über das die weitere Kommunikation läuft.

Im folgenden Beispiel wird dieses Schema benutzt, um Informationen über einen Link abzufragen. Insbesondere verwendet die Anwendung die Methode `getContentType()`, um den Typ des Dokuments zu erfragen. Der Rückgabewert ist eine Zeichenkette mit der MIME-Angabe des Typs.

Die Methode `getContent()` liefert ein dazu passendes Objekt, um den Inhalt des Links zu lesen. Bei HTML-Dateien sind dies Varianten von `InputStream`. Zeigt der Link aber beispielsweise auf eine Bilddatei im GIF-Format, erhält man ein Objekt der Klasse `URLImageSource` zurück.

```
import java.net.*;
import java.io.*;

public class ExamineURL {
    static int keyWidth = 20;

    public static void main( String args[] ) {
        // Kopieren des Argumentes beim Aufruf
        String link = args[0];

        try {
            System.out.println("Trying URL " + link );
            URL url = new URL( link );
            URLConnection conn = url.openConnection();
            int n = 1;

            // Ausgabe der Elemente des http-Headers
            String key;
            System.out.println("Header fields:");
            while( (key=conn.getHeaderFieldKey(n) ) != null ) {
                key += ": ";
                while( key.length() < keyWidth ) key += " ";
                System.out.println( n + " " + key
                                   + conn.getHeaderField(n) );
                ++n;
            }
            // Objekt für Zugriff auf Inhalt
            Object o = conn.getContent();
            System.out.println( "Content:      " + o );
            System.out.println( "ContentType: "
                               + conn.getContentType() );
            System.out.println( "Permission:  "
                               + conn.getPermission() );

        } catch( Exception ex ) {
            System.out.println("Cannot create URL "
                               + link + " Exception:" + ex);
        }
    }
}
```

Ruft man diese Anwendung mit der Homepage der FH Friedberg als Parameter auf, erhält man folgende Ausgaben zurück:

```
java ExamineURL http://www.fh-friedberg.de
Trying URL http://www.fh-friedberg.de
Header fields:
1 Date: Mon, 13 Dec 2004 08:56:52 GMT
2 MIME-Version: 1.0
3 Content-Type: text/html
Content: java.io.PushbackInputStream@3169f8
ContentType: text/html
Permission: (java.net.SocketPermission
www.fh-friedberg.de:80 connect,resolve)
```

12.5 Mailto-Links

Ein URL-Objekt kann nicht nur auf Dateien verweisen, sondern auch andere Dienste ansprechen. Ein Beispiel sind Verweise mit dem Protokoll mailto. Damit kann ein URL-Objekt auch E-Mails verschicken. Die Klasse URLEmail im Beispiel verwendet diesen Mechanismus, um eine Nachricht mit einer Klausurnote zu verschicken. Wichtig dabei ist, dass vor dem Versand der Java-VM noch der zu verwendende Mail-Server bekannt gemacht werden muss. Dies erfolgt durch Setzen der entsprechenden Systemeigenschaft:

```
System.setProperty("mail.host", "mailto.t-online.de");
```

In analoger Art und Weise kann man bei Bedarf auch andere Eigenschaften wie etwa den Name eines Proxy-Servers festlegen.

```
import java.io.*;
import java.net.*;

public class URLEmail
{
    public static void main(String args[]) throws Exception {
        String from = "stephan.euler@t-online.de";
        String to = "stephan.euler@t-online.de";
        String subject = "Klausurergebnis IN3";
        String nachricht = "Sie haben die Note 2.";

        System.setProperty("mail.host",
                           "mailto.t-online.de");
        URL u = new URL("mailto:" + to);
        URLConnection c = u.openConnection();
        c.setDoOutput(true);
        System.out.println("Connecting...");
        c.connect();
    }
}
```

```

        PrintWriter out = new PrintWriter(
            new OutputStreamWriter(c.getOutputStream()));
        out.println("Subject: " + subject);
        out.println(nachricht);
        out.close();
        System.out.println("Nachricht versendet.");
    }
}

```

Prof. Dr. Stephan Euler

tecCHANNEL-Links zum Thema	Webcode	Compact
Java-Einsatz im Netzwerk	a1502	S.129
Alle Listings zum Download	a1541	–
So funktionier E-Mail	a819	–
Hyper-Threading: Optimierungen und Fallen	a1108	–

Mit den Webcodes gelangen Sie auf www.tecChannel.de direkt zum gewünschten Artikel. Geben Sie dazu den Code in das Feld „Webcode“ in der Titelleiste von www.tecChannel.de ein. Mit „p“ gekennzeichnete Webcodes beziehen sich auf Online-Artikel aus tecCHANNEL-Compact. Es ist möglich, dass Artikel zunächst exklusiv in tecCHANNEL-Compact veröffentlicht werden und daher erst später über den Webcode erreichbar sind.

13. Datensicherheit und Verschlüsselung

Mit der zunehmenden Bedeutung vernetzter Rechnersysteme wächst der Bedarf an Sicherheit. Insbesondere die Verlagerung von kritischen Abläufen wie

- Einkaufen und Verkaufen
- Abwickeln von Bank- und Börsengeschäften
- Bewerbungen auf Ausschreibungen

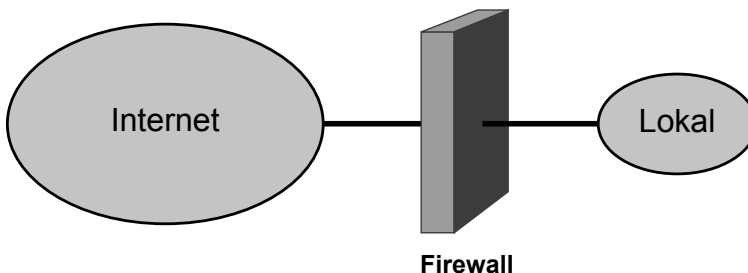
hin zu elektronischen Formen verlangt eine sichere und vertrauliche technische Grundlage. Darüber hinaus gilt es, die direkt oder indirekt an das universelle Netz angeschlossenen Systeme gegen unberechtigte Zugriffe zu sichern.

13.1 Firewall

Wie bei der namensgebenden Anwendung als Schutz gegen die Ausbreitung von Bränden, hat eine Firewall die Aufgabe, ein privates Netz vor unkontrollierten Zugriffen aus anderen Netzen zu schützen.

Unter Firewall versteht man das allgemeine Konzept einer strikten Trennung zwischen internen und externen Netzen. Nach vorgegebenen Regeln entscheidet die Firewall, welche Daten passieren dürfen. Meist stellt eine Firewall ein eigenständiges Gerät (Standardrechner mit entsprechender Software oder spezielle Hardware-Firewall) dar.

Die so genannte Perimeter-Firewall sichert die Grenzen des internen Netzwerks ab und ist zwischen einem lokalen Netz und dem Internet geschaltet. Man kann dies auch als einen Router zwischen den beiden Netzen sehen, der allerdings zusätzlich den Datenverkehr kontrolliert.



© tecCHANNEL

Bild 13.1: Eine Perimeter-Firewall trennt das lokale Netz vom Internet.

Am einfachsten aufgebaut sind filterbasierte Firewalls (Paketfilter). Die Regeln beziehen sich dann auf bestimmte Adressen und Ports. Eine Möglichkeit ist, eine Tabelle mit erlaubten Quell- und Zieladressen zu führen. Als Prinzip gilt dann: „Alles, was nicht ausdrücklich erlaubt ist, ist verboten.“ Dann würde eine Zeile

169.254.107.238, 123, 212.125.100.86, 80

bedeuten, dass alle Pakete vom Rechner 169.254.107.238 und Port 123 zu Rechner 212.125.100.86, Port 80, passieren dürfen. Als Alternative kann man auch das liberale Konzept einsetzen: „Alles, was nicht ausdrücklich verboten ist, ist erlaubt.“ In diesem Fall würde der Eintrag genau diese Kombination verbieten.

Ein Paketfilter arbeitet auf den unteren Protokollebenen. In dem beschriebenen Beispiel analysiert die Firewall die Pakete bis zur Schicht 4, um dann anhand der IP-Adresse und der Portnummer über die Weiterleitung zu entscheiden. Dieser Ansatz wird daher auch als Network Level Firewall bezeichnet.

Die Wirksamkeit einer Firewall lässt sich erhöhen, wenn sie nicht nur auf Basis eines einzelnen Pakets entscheidet, sondern eine komplette Verbindung als eine Einheit betrachtet. Am weitesten gehen in diesem Sinne Application Level Firewalls (Proxy-basierte Firewalls), die auf der höchsten Schicht 7 laufen.

In diesem Fall kommuniziert eine Anwendung im internen Netz, etwa ein Browser, nicht direkt mit dem Internet. Vielmehr wickelt den gesamten Verkehr ein Stellvertreter – der Proxy-Server – ab. Nur der Proxy hat die Berechtigung, Daten mit dem Internet auszutauschen.

Dieses Konzept erlaubt weit gehende Kontroll- und Protokollfunktionen und gilt als sehr sicher. Negativ sind eine Einschränkung der Flexibilität etwa bei exotischen Protokollen und eine verringerte Transfergeschwindigkeit. Als Alternative kommen daher auch Mischformen zwischen den Filtern auf Paketebene und den Proxys auf Anwendungsebene zum Einsatz.

Von außen ist die Firewall der einzige Angriffspunkt auf das lokale Netz. Daher legen Administratoren besondere Sorgfalt auf die Sicherheit dieses Systems. Ist diese Hürde allerdings erst einmal genommen, so befindet sich der Eindringling bereits im relativ ungeschützten internen Netz. Um diese Gefahr zu verringern, kann das interne Netz durch eine zweite Firewall geschützt werden.

Hinter der ersten Firewall sind dann nur Rechner angeordnet, die von außen erreichbar sein müssen. Typischerweise findet man hier Webserver und E-Mail-Server. Den Übergang aus diesem ersten Sicherungsring zu dem internen Netz regelt dann eine zweite Firewall. Die Zone zwischen den beiden Firewalls bildet ein Grenznetz das man auch als De-Militarized Zone (DMZ) bezeichnet.

Zur (zusätzlichen) Absicherung von einzelnen Rechnern im internen Netz bieten sich Personal oder Desktop Firewalls an. Solche Systeme nutzen häufig Privatanwender, um ihren per Modem direkt am Internet angeschlossenen Rechner zu schützen. Die Personal Firewall wirkt dann als Paketfilter, der unerwünschten Datenverkehr zwischen dem Rechner und dem Internet blockiert. Personal Firewalls

bieten sich jedoch auch in Firmen an, um kritische Rechner zusätzlich abzusichern. Denn nicht immer geschieht ein Angriff von außen. So erfolgten etwa 2003 über 63 Prozent aller Datendiebstähle von firmeninternen Mitarbeitern.

Man darf den Gewinn an Sicherheit durch eine Personal Firewall nicht überschätzen. Ein über eine E-Mail eingedrungener Virus könnte intelligent genug sein, zumindest die Firewall des befallenen Rechners von innen zu überwinden. Beispielsweise könnte er eine Regel für eine erlaubte Verbindung eintragen und diese dann für eigene Zwecke nutzen. Eine Personal Firewall ist allerdings gut geeignet, um eine Warnung bei auffälliger Netzwerkaktivität zu erhalten.

13.2 Virtuelle private Netze

Firewalls dienen dazu, das eigene Netz gegenüber dem Internet abzuschotten. Nun gibt es allerdings häufig Situationen, in denen gerade ein externer Zugang erforderlich ist. Ein typischer Fall ist ein Mitarbeiter auf Dienstreise, der Zugriff auf interne Daten benötigt.

Ein recht sicherer Weg besteht in der Verwendung von Wählleitungen. Die Verbindung geht dann über ein Modem und eine Telefonleitung zu einem entsprechenden Firmenanschluss. Durch die leitungsbasierende Verbindungsart ist ein Abhören nicht für jeden und nicht ohne Aufwand möglich. Den Firmenanschluss schützt man durch ein starkes Passwort, zudem sollte man die Daten aber auch noch verschlüsseln.

Wesentlich eleganter als die Wählleitung ist es, auf die Infrastruktur des Internets zurückzugreifen. Der Mitarbeiter nutzt dann beispielsweise den WLAN-Zugang in seinem Hotel, um sich mit dem Internet zu verbinden. Zudem muss die Firma einen Zugang vom Internet zu den internen Daten bereitstellen. Für den Weg dazwischen wird die Infrastruktur des Internet genutzt. Das Problem ist nun, wie diese Verbindung sicher gestaltet werden kann. Der Weg durch das Internet ist nicht geschützt. Daher besteht die Gefahr, dass Unberechtigte den Inhalt der IP-Pakete einsehen oder gar manipulieren.

Betrachten wir ein Szenario, bei dem ein externer Mitarbeiter (Client) auf einen Rechner X im internen Netz zugreifen möchte. Dazu muss er Daten an die Adresse X schicken. Da es sich um vertrauliche Informationen handelt, müssen diese Daten auf jeden Fall verschlüsselt werden. Ein zusätzlicher Schutz soll zudem dafür sorgen, dass der Rechner X öffentlich gar nicht bekannt und nicht ohne weiteres aus dem Internet erreichbar ist.

Eine in diesem Sinne sichere Verbindung realisiert ein so genannter Tunnel durch das Internet. Dazu wird das eigentliche Netzwerkpaket komplett mit Header und Nutzdaten als Einheit verschlüsselt. Das daraus resultierende Packet schickt man dann als verschlüsselte Nutzlast in einem IP-Paket an einen speziellen Server (Remote-Tunnel-Server) im Firmennetz, der von außen erreichbar ist. Aus Sicht des Internets handelt es sich dabei also um eine normale IP-Verbindung zwischen

einem Client und diesem Server. Die Daten in dem IP-Paket sind allerdings durch eine Verschlüsselung geschützt. Der Server nimmt das Paket entgegen und entschlüsselt die Daten. Damit rekonstruiert er das eigentliche Paket, das er dann in das interne Netz einspeist.

Durch diese Technik kann eine Endanwendung sich so verhalten, als wäre der Rechner direkt an das Firmennetz angeschlossen (Virtuelles privates Netz, VPN). Die für das interne Netzwerk bestimmten Pakete durchqueren das dazwischen liegende Internet eingebettet in ein IP-Paket. Auf diese Weise kann man eine sichere Kommunikation über das Internet erreichen. Dies ist nicht der einzige Vorteil. Der Inhalt des Paketes spielt für den Transport durch den IP-Tunnel keine Rolle. Es ist daher durchaus möglich, dass das firmeninterne Netz ein anderes Protokoll oder private IP-Adressen verwendet.

Eine komplette Lösung für den Aufbau eines VPN-Tunnels besteht aus mehreren Komponenten wie einer Benutzer-Authentifikation oder einer Datenverschlüsselung. Einige verbreitete Tunneling-Protokolle sind:

- Point to Point Tunneling Protocol (PPTP)
- Layer 2 Tunneling Protocol (L2TP), RFC 2661
- IP-Security-Protokoll (IPSec)

Sie wurden für verschiedene Anwendungsmöglichkeiten entwickelt und unterscheiden sich dementsprechend in ihren Möglichkeiten.

13.3 Verschlüsselung

Der Inhalt der Pakete im Internet ist relativ leicht einsehbar. Mit wenig Aufwand kann man auf einem Rechner beispielsweise alle ankommenden und abgehenden IP-Pakete lesen. Ohne weitere Maßnahmen sind daher kritische Daten wie Kennwörter oder die Nummern von Kreditkarten nicht vor Unberechtigten geschützt. Im Internet hat man als Anwender keinen Einfluss auf den Übertragungsweg. Man muss daher davon ausgehen, dass die Übertragung nicht abgesichert ist und der Inhalt der Pakete auch Fremden zugänglich ist.

Um trotzdem sensible Daten übermitteln zu können, muss man diese Daten durch eine Verschlüsselung schützen. Im Prinzip handelt es sich dabei um eine Art von Kodierung, ähnlich wie der Einbau von Paritätsinformationen. Der Verschlüsselungsalgorithmus bringt die Nachricht – den Klartext – in eine andere Form. Diese kodierte Nachricht wird im Netz übertragen. Der Empfänger kann den Verschlüsselungsalgorithmus umkehren und den Klartext rekonstruieren.

Die Besonderheit bei diesem Vorgehen ist allerdings, dass der Algorithmus oder dabei verwendete Parameter geheim gehalten werden. Nur Sender und Empfänger kennen den Algorithmus oder die benötigten Schlüssel. Dritte können zwar während der Übertragung die Nachricht abfangen, aber sie können diese nicht verstehen oder gar manipulieren.

Anschaulich kann man sich die Nachricht wie ein Paket mit einem Schloss vorstellen. Der Sender verschließt das Paket und nur mit dem richtigen Schlüssel kann es wieder geöffnet werden. Die Methoden der Verschlüsselung sind Gegenstand der Kryptografie. Demgegenüber bezeichnet Kryptoanalyse die Wissenschaft von der Entschlüsselung ohne Kenntnis des Schlüssels.

Der folgende Abschnitt führt zunächst anhand eines einfachen Beispiels in die Grundregeln ein. Anschließend folgt der Übergang zu digitalen Systemen mit einer Beschreibung einiger wesentliche Verfahren. Darauf aufbauend werden einige konkrete Anwendungen vorgestellt.

In der Literatur zur Kryptografie erhalten die beteiligten Personen oft Namen in der Reihenfolge des Alphabets (Alice, Bob, Carol...). Im Rahmen dieses Textes werden aber weiterhin die Bezeichnungen Sender und Empfänger verwendet. In symmetrischen Fällen, bei denen diese Unterscheidung nicht sinnvoll ist, werden diese durch die Abkürzungen A und B ersetzt.

13.3.1 Monoalphabetische Verschlüsselung

Betrachten wir ein einfaches Beispiel für einen Algorithmus zur Verschlüsselung von Texten. Gegenüber einer starren Umsetzung arbeitet diese Verschlüsselung bereits mit einem variablen Schlüssel, so dass die Kenntnis des Algorithmus noch nicht zur Entschlüsselung genügt:

- Die Buchstaben eines Schlüsselwortes werden in umgekehrter Reihenfolge aufgeschrieben. Eventuelle doppelt vorkommende Buchstaben werden dabei nur einmal berücksichtigt.
- Alle noch fehlenden Buchstaben des Alphabets werden in umgekehrter Reihenfolge angehängt.
- Dem ersten Buchstaben der so entstandenen Zeichenkette ordnet man dann den Buchstaben „A“ zu, dem folgenden „B“ und so weiter.

Nach dieser Vorschrift wird aus dem Schlüsselwort HAMLET folgende Tabelle zur Verschlüsselung erzeugt:

A	B	C	D	E	F	G	H	I	J	K	L	M
T	E	L	M	A	H	Z	Y	X	W	V	U	S
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
R	Q	P	O	N	K	J	I	G	F	D	C	B

Bild 13.2: Der monoalphabetische Algorithmus ordnet jedem Buchstaben einen anderen zu.

Anhand dieser Tabelle wird der zu übermittelnde Text umgesetzt. Jeder einzelne Buchstabe wird gemäß dieser Zuordnung durch einen anderen ersetzt. Als Beispiel ergibt sich folgende Verschlüsselung:

S	E	I	N		O	D	E	R		N	I	C	H	T	S	E	I	N
K	A	X	R		Q	M	A	N		R	X	L	Y	J	K	A	X	R

Bild 13.3: Die Frage aller Fragen in transformierter Form.

Die Leerzeichen verraten noch relativ viel über die Satzstruktur und bieten Ansatzpunkte zur Entschlüsselung. Es ist daher eine gute Idee, entweder das Leerzeichen als weiteres Zeichen in die Übersetzungstabelle aufzunehmen oder einfach alle Leerzeichen zu entfernen. In aller Regel wird der Empfänger nach der Entschlüsselung aller Buchstaben die Wortgrenzen leicht rekonstruieren können. Ohne Leerzeichen erhält man in dem Beispiel die verschlüsselte Nachricht

KAXRQMANRXLYJKAXR

Für den Uneingeweihten handelt es sich hier um eine sinnlose Buchstabenfolge. Der Empfänger kann demgegenüber – sofern er das Schlüsselwort kennt – den Prozess der Verschlüsselung umkehren. Dazu konstruiert er zuerst nach der obigen Vorschrift die Zuordnungstabelle. Dann ersetzt er die empfangenen Zeichen wieder durch die ursprünglichen Zeichen. Aus der resultierenden Buchstabenfolge rekonstruiert er dann durch Einfügen der Leerzeichen an den passenden Stellen den ursprünglichen Text.

Da nur ein Alphabet zur Verschlüsselung benutzt wird, bezeichnet man derartige Verfahren als monoalphabetisch. Weiterhin handelt es sich um ein symmetrisches Verfahren: Es gibt genau einen Schlüssel, der sowohl zum Verschlüsseln als auch zum Entschlüsseln benötigt wird. Sender und Empfänger benötigen die gleichen Informationen. Kennt ein Unbefugter den Schlüssel, so kann er abgefangene Nachrichten entschlüsseln und gefälschte selbst versenden.

Solche Verfahren waren in unterschiedlichen Varianten jahrhundertlang in Gebrauch. Allerdings bieten sie nur einen geringen Schutz. Die Verschlüsselung kann relativ leicht durch den Einsatz der Häufigkeitsanalyse gebrochen werden. Dabei nutzt man die unterschiedliche Häufigkeit für das Auftreten der einzelnen Buchstaben aus. So treten in deutschen Texten die Buchstaben „E“ mit 17 Prozent und „N“ mit 10 Prozent am häufigsten auf.

Daher kann man davon ausgehen, dass der häufigste Buchstabe in einem derart verschlüsselten Text mit recht hoher Wahrscheinlichkeit eigentlich das „E“ darstellt. Weiterhin kann man nach häufig vorkommenden Buchstabenpaaren suchen. So sind die Kombinationen LL, MM, NN, EE oder TT gute Kandidaten für häufig doppelt vorkommende Zeichen.

Auch immer wiederkehrende längere Zeichenfolgen wie „SCH“ bieten Ansatzpunkte für die Entschlüsselung. Auf diese Art und Weise kann durch systematisches Raten unter Berücksichtigung der bekannten Häufigkeiten von Buchstaben und Buchstabenfolgen eine monoalphabetische Verschlüsselung relativ leicht entschlüsselt werden.

Eine besondere Hilfe ist dabei, wenn man einige der Wörter im Klartext kennt. In militärischen Telegrammen taucht zum Beispiel oft ein Hinweis auf das Wetter auf. Darin kann ein Angreifer also gezielt nach der Folge ETTE beziehungsweise dem Muster XYYX suchen. Findet er ein Muster XYYX, so kann er die Zuordnung $X=E$ und $Y=T$ ausprobieren.

Derartige einfache Verschlüsselungsverfahren sind zwar nicht vollkommen nutzlos, aber sie bieten nur einen geringen Schutz. Ob dieser Schutz als ausreichend erachtet werden kann, hängt von der Anwendung ab. Soll nur eine sehr kurze Nachricht geschützt werden oder ist der Schutz nur für eine kurze Zeit erforderlich, so kann eine solche einfache Verschlüsselung durchaus ausreichen. Die verschlüsselte Nachricht ist zunächst gegenüber Unbefugten geschützt. Jemand, der zufällig oder absichtlich die Nachricht liest, erkennt nicht direkt den Inhalt.

13.3.2 Digitale Verschlüsselung

Mit dem Aufkommen von Rechenautomaten und Computern änderte sich zunächst nichts am Prinzip der Verschlüsselung. Allerdings beschleunigen sie die Verschlüsselung erheblich und ermöglichen deutlich aufwendigere Kodierverfahren. Andererseits erhöhen sie auch die Anzahl der möglichen Versuche zum Brechen des Codes um Größenordnungen gegenüber dem manuellen Ansatz. Mit zunehmender Rechenleistung verkürzt sich die Zeitspanne, für die ein Code Sicherheit bietet. So wurden die ersten Rechner während des zweiten Weltkrieges in den britischen Geheimlaboren zur Entschlüsselung des deutschen Funkverkehrs eingesetzt [22]. Diese Rechner hatte man speziell für diesen Zweck entwickelt.

Die beginnende Digitalisierung erforderte neuartige Verschlüsselungsverfahren. Diese arbeiten nicht mehr mit einzelnen Zeichen, sondern auf Bitebene. Die Nachricht wird dabei unabhängig von ihrem Inhalt als Bitstrom behandelt. Bei den im Folgenden vorgestellten Verfahren handelt es sich um so genannte Blockcodes. Sie unterteilen den Bitstrom in Blöcke mit einer vorgegebenen Anzahl von Bits. Jeden Block verschlüsseln sie dann durch eine Bitmanipulation.

Ein solches Verfahren ist der Data Encryption Standard (DES). Der Algorithmus wurde Mitte der 70er Jahre entwickelt und später vom amerikanischen National Institute of Standards and Technology (NIST) als Standard eingeführt [23]. DES beruht auf der Verwürfelung von Gruppen mit jeweils 64 Bit. In einem mehrstufigen Verfahren verknüpft er die einzelnen Bits miteinander und überschreibt damit den Block. Dabei durchläuft DES insgesamt 16 Iterationen – auch als Runden bezeichnet. Die Details der Verknüpfungsfunktion legt ein Schlüssel mit einer Länge von 56 Bit fest. In jedem Durchlauf wird aus diesem Hauptschlüssel ein spezieller Rundenschlüssel berechnet.

Im Grunde ist DES in dieser Basisversion ein monoalphabetisches Verfahren. Es bildet Wörter der Länge 64 Bit auf andere Wörter der gleichen Länge ab. Kennt man die Struktur der ursprünglichen Daten, so eröffnet diese Eigenschaft die Möglichkeit zur Manipulation. Handelt es sich beispielsweise um Datensätze für

Mitarbeiter, so kann man einen Block von 64 Bit, beispielsweise eine achtstellige Mitarbeiterkennung, einfach durch einen anderen Block ersetzen. Um derartige Manipulationen auszuschließen, besteht die Möglichkeit, in die Verschlüsselung eine Abhängigkeit von vergangenen Blöcken einzubauen. Damit ergibt ein bestimmter Datenblock in Abhängigkeit vom Kontext unterschiedliche Codes und die Daten sind gegen einfachen Austausch geschützt. Generell führt diese Verkettung zu einem besseren Schutz gegen die Kryptoanalyse.

Durch den Schlüssel der Länge 56 sind bei DES insgesamt 2^{56} verschiedene Abbildungen möglich. Kennt man ein kurzes Stück Klartext, so kann man durch erschöpfende Suche alle 2^{56} möglichen Schlüssel ausprobieren. Geht man davon aus, dass ein handelsüblicher PC einige zehn Millionen Schlüssel pro Sekunde ausprobieren kann, so benötigt ein Test aller Schlüssel rund 100 Jahre. Im Mittel wird der richtige Schlüssel nach 50 Jahren gefunden.

Allerdings lässt sich die Aufgabe in einfachster Art und Weise parallelisieren und auf viele Rechner aufteilen. Anfang 1999 ist es gelungen, durch die Nutzung der Leerlaufzeit vieler per Internet verbundener Computer eine DES-verschlüsselte Nachricht innerhalb von 23 Stunden zu dechiffrieren. Weiterhin kann durch spezielle Hardware-Bausteine die Suche wesentlich beschleunigt werden. Daher kann man das einfache DES zumindest für Daten, die längere Zeit sicher bleiben müssen, nicht mehr empfehlen.

Eine ausreichende Sicherheit kann man durch einen längeren Schlüssel oder die dreifache Anwendung von DES erhalten. Das verbreitete Tripple-DES erreicht damit eine effektive Schlüssellänge von 168 Bit und scheint damit auch auf längere Zeit als sicher. Daneben haben verschiedene Entwicklungsteams eine Reihe anderer Verfahren vorgeschlagen. Das wohl wichtigste darunter ist IDEA (International Data Encryption Algorithm), entwickelt von Xuejia Lai und James L. Massey von der ETH Zürich [24]. IDEA basiert auf einem Schlüssel der Länge 128 Bit. Die erschöpfende Suche auf einem einzelnen PC würde über 10^{25} Jahre benötigen. Damit besteht ausreichend Sicherheit auch gegen Angriffe mit optimierten Hardware-Bausteinen und parallelem Einsatz vieler Rechner.

13.3.3 Gemeinsame Schlüsselvereinbarung

Grundsätzlich bieten die beschriebenen Verfahren eine ausreichende Sicherheit. Sind die verwendeten Schlüssel groß genug, so können sie nach derzeitigem Stand nicht gebrochen werden. Das Problem dieser Verfahren liegt jedoch in der Verwaltung der Schlüssel. Will ein Sender eine verschlüsselte Nachricht verschicken, muss er dem Empfänger seinen Schlüssel mitteilen oder einen vom Empfänger zuvor erhaltenen Schlüssel verwenden. Neben der eigentlichen Nachricht muss man also auch den Schlüssel übertragen. Diese Übertragung ist besonders sensibel. Denn falls ein Dritter in Besitz des Schlüssels gelangt, kann er die verschlüsselte Nachricht ebenfalls lesen. Daher muss der Schlüssel auf einem besonders sicheren Weg verschickt werden. Beispielsweise kann eine Bank die Schlüssel mit einem zuverlässigen Boten an ihre Kunden ausliefern.

Mit dem zunehmenden digitalen Nachrichtenverkehr stellte die Schlüsselverteilung ein immer größeres logistisches Problem dar. Umso wichtiger war die Entdeckung, dass eine Schlüsselvereinbarung auch über einen unsicheren Kanal möglich ist. Sender und Empfänger vereinbaren dabei einen Schlüssel, den sie dann wie beschrieben für eine symmetrische Verschlüsselung verwenden. Die Vereinbarung des Schlüssels kann jedoch über einen offenen Kanal erfolgen. Ein Dritter kann den Schlüssel aus den übertragenen Informationen nicht rekonstruieren.

Diese zunächst verblüffende Möglichkeit publizierten zuerst Whitfield Diffie und Martin Hellmann [25]. Der Kerngedanke ist, eine mathematische Funktion zu verwenden, die praktisch nicht umkehrbar ist. Eine solche „Einwegfunktion“ liefert für einen Eingangsparameter ohne viel Rechenaufwand ein Ergebnis. Aber die Umkehrung – die Berechnung des Ausgangswertes aus dem Funktionswert – ist in vernünftiger Zeit nicht möglich. Die Schlüsselvereinbarung nach Diffie und Hellmann (manchmal auch als Diffie-Hellmann-Merkle-Verfahren bezeichnet) nutzt die Modulo-Funktion, um diesen Einwegcharakter zu erzielen.

Die Modulo-Funktion berechnet den Rest bei einer ganzzahligen Division. Der Ausdruck $17 \bmod 5$ als Beispiel ergibt den Wert 2, da $17 = 3 \cdot 5 + 2$ gilt. Bei arithmetischen Ausdrücken werden die Operanden zunächst wie gewohnt verknüpft, und auf das Ergebnis wird die Modulo-Funktion angewandt. So ergibt beispielsweise $3 \cdot 3 \bmod 5$ das Ergebnis 4. Das Resultat der Exponentialfunktion $3^x \bmod 31$ für verschiedene Werte von x zeigt folgendes Bild. Anders als bei herkömmlichen Funktionen ist hier keine klare Struktur zu erkennen. Aufgrund der Modulo-Funktion schwanken die Ergebnisse selbst für direkt aufeinander folgende Werte von x stark und lassen keinen Rückschluss auf die Nachbarwerte zu.

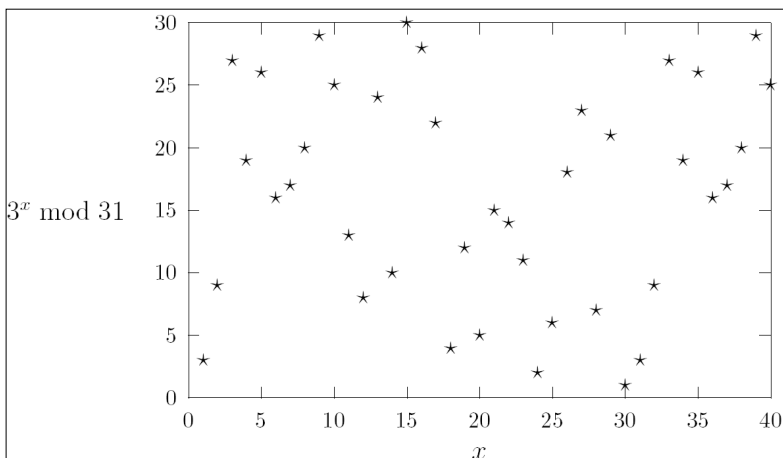


Bild 13.4: Die Modulo-Funktion lässt keinen Rückschluss auf den Ausgangswert zu.

Der Einwegcharakter der Modulo-Funktion resultiert aus der Schwierigkeit, die Funktion zu invertieren. Es ist zwar relativ leicht, für ein gegebenes x die Exponentialfunktion zu berechnen und auf das Ergebnis die Modulo-Operation anzuwenden. Aber für die Umkehrung ist kein schnelles Verfahren bekannt. Die Aufgabe, für einen Wert a ein x zu finden, so dass $3^x \bmod 31 = a$ erfüllt wird, erfordert einen sehr viel höheren Rechenaufwand. Bei großen Zahlen mit mehreren hundert Stellen lässt sich mit geschickten Algorithmen die Exponentialfunktion immer noch in überschaubarer Zeit auswerten. Aber die Umkehrung erfordert dann selbst bei leistungsfähigen Systemen so viel Zeit, dass sie praktisch unmöglich ist.

A	öffentlich	B
	$Y = 7, P = 11$	
$A = 3$		$B = 6$
$Y^A \bmod P$ $7^3 \bmod 11 = 2$		$Y^B \bmod P$ $7^6 \bmod 11 = 4$
	$\alpha = 2, \beta = 4$	
$\beta^A \bmod P$ $4^3 \bmod 11 = 9$		$\alpha^B \bmod P$ $2^6 \bmod 11 = 9$

Bild 13.5: Schlüsselvereinbarung: Der Algorithmus nach Diffie und Hellmann erzeugt einen sicheren Schlüssel über einen unsicheren Kanal.

Betrachten wir die Schlüsselvereinbarung nach Diffie und Hellmann an einem einfachen Beispiel. In der Tabelle sind die einzelnen Schritte dazu dargestellt. Zunächst vereinbaren die beiden Partner A und B zwei Zahlen Y und P . Diese Zahlen müssen bestimmte Bedingungen erfüllen. So müssen Y , $(Y-1)/2$ und P Primzahlen sein. Diese Zahlen können öffentlich bekannt sein. Dann wählt jeder Partner eine geheime Zufallszahl x aus und berechnet damit den Wert der Funktion $Y^x \bmod P$. Das Resultat α beziehungsweise β schickt er an sein Gegenüber. Schließlich berechnet er aus dem vom Gegenüber erhaltenen Wert und seiner Geheimzahl den Schlüssel. Durch die geschickte Konstruktion der Funktion erhalten beide Partner den gleichen Wert – im Beispiel die Zahl 9. Diese Zahl können sie als symmetrischen Schlüssel verwenden. In der Praxis nutzt man sehr viel größere Zahlenwerte, um einen Schlüssel ausreichender Länge zu erhalten.

Die Wirkungsweise dieses Verfahrens lässt sich leicht nachvollziehen. Ausgeschrieben ergeben die beiden Berechnungen $(Y^B \bmod P)^A \bmod P$ und $(Y^A \bmod P)^B \bmod P$. Nach den Regeln der modularen Arithmetik kann der zweite Exponent in den ersten Ausdruck mit hinein gezogen werden. Damit sind beide Ausdrücke gleichwertig mit $Y^{AB} \bmod P$.

Ein Dritter kann über die öffentliche Leitung die Zahlen Y , P , α und β erfahren. Um den Schlüssel zu berechnen, fehlt ihm jedoch die Zahl A (oder gleichwertig B). Dafür müsste er die Funktion

$$Y^A \bmod P = \alpha$$

nach A auflösen, wofür bislang kein effizientes Lösungsverfahren bekannt ist. In dem Beispiel mit den kleinen Zahlen kann aus der Gleichung $7^A \bmod 11 = 2$ der gesuchte Wert $A=3$ noch relativ leicht durch Probieren bestimmt werden. Aber für große Zahlenwerte führt dies zu so langen Zeiten, dass die Umkehrung praktisch nicht möglich ist.

Das Verfahren von Diffie und Hellmann ermöglicht es, über eine Leitung einen gemeinsamen Schlüssel zu vereinbaren, ohne dass die übertragenen Informationen von Dritten verwendet werden können. Allerdings ist es nicht unter allen Umständen sicher. Wenn ein Dritter X die Möglichkeit hat, Nachrichten abzufangen und zu verändern, kann er sich in das Austauschprotokoll einschalten. Er wählt eine eigene Zufallszahl und führt damit das Protokoll mit den Partnern A und B durch. Dann entstehen sichere Verbindungen zwischen A und X sowie B und X. A und B sind im besten Glauben, ihre Nachrichten direkt aneinander zu senden. In Wirklichkeit erreichen die Nachrichten immer zuerst X, der sie dann nach Belieben manipulieren kann, bevor er sie weitergibt. Eine Möglichkeit, sich gegen diese so genannten „Man in the Middle“-Angriffe zu schützen, ist die Verwendung fester Schlüssel, die eine zuverlässige Stelle verwaltet.

13.3.4 Öffentliche Schlüssel

Ein praktischer Nachteil bleibt bei dem beschriebenen Verfahren der Schlüsselvereinbarung: Beide Partner müssen gleichzeitig aktiv sein, um das Protokoll durchführen zu können. Damit ist das Verfahren etwa zum Versand einer E-Mail nur schlecht einsetzbar. Einfacher zu verwenden sind Verfahren, bei denen der Absender seine Nachricht verschlüsseln kann, ohne vorher mit dem Empfänger direkt Informationen austauschen zu müssen. Dies ermöglichen Verfahren mit öffentlichen Schlüsseln (Public Key) [26].

Hierbei stellt der Empfänger einen Schlüssel bereit, der allgemein bekannt gemacht wird. Der Sender verwendet diesen Schlüssel, um seine Nachricht zu verschlüsseln. Der öffentliche Schlüssel genügt jedoch nicht zur Entschlüsselung. Benötigt wird vielmehr ein zweiter Schlüssel. Diesen privaten Schlüssel kennt nur der Empfänger, so dass auch nur er die Verschlüsselung aufheben kann. Solange er ihn geheim hält, kann niemand sonst die Nachricht verstehen.

Das bekannteste derartige Verfahren ist RSA, so benannt nach den Entwicklern Ronald Rivest, Adi Shamir und Leonhard Adleman. RSA basiert ebenfalls auf der Modulo-Arithmetik. Der öffentliche Schlüssel besteht aus zwei Zahlen, die wie folgt gewählt werden:

- Der potenzielle Empfänger A wählt zwei Primzahlen p und q .
 - Aus den beiden Primzahlen berechnet er das Produkt $n=p \cdot q$.
 - Weiterhin wählt er eine Zahl $e < n$.
 - e und n bilden den öffentlichen Schlüssel und werden entsprechend publiziert.
-

Damit kann ein Sender eine Nachricht an den Empfänger A nach der folgenden Vorschrift verschlüsseln

$$C = M^e \bmod n$$

Die Zahl $M < n$ repräsentiert einen Block mit der Länge $\ln_2(n)$ Bit. Eine längere Nachricht muss in entsprechend viele Teile aufgespalten werden. Die resultierende Zahl C ist der geheime Text für diesen Block. Auf Grund der Eigenschaften der Modulo-Funktion ist es sehr schwer, aus C und e wieder auf M zu schließen. Bei hinreichend großen Zahlen n und e kann man die Umkehrung als praktisch unmöglich betrachten.

Damit ist die Verschlüsselung zwar sicher, aber es stellt sich die Frage, wie der Empfänger die Nachricht lesen kann. Hier kommen die beiden geheim gehaltenen Zahlen p und q ins Spiel. Mit diesen beiden Zahlen kann eine weitere Zahl d mit der Eigenschaft

$$e \cdot d = 1 \bmod ((p-1) \cdot (q-1))$$

berechnet werden. Details dazu sind im Kapitel 14 beschrieben. An dieser Stelle genügt es festzustellen, dass die Bestimmung von d relativ einfach ist. Bei dieser Wahl des Produkts ed gilt weiterhin die Beziehung

$$C^d = M^{e \cdot d} \bmod n = M$$

Auch hier sei für Details auf Kapitel 14 verwiesen. Mit anderen Worten: Der Empfänger braucht lediglich einen Exponenten zu berechnen und auf das Ergebnis die Modulo-Funktion anzuwenden, um gemäß

$$M = C^d \bmod n$$

den ursprünglichen Wert M zu erhalten. Die Sicherheit von RSA beruht auf den beiden Zahlen p und q . Kennt man sie oder zumindest das Produkt $(p-1) \cdot (q-1)$, so ist es leicht, daraus d zu bestimmen. Öffentlich bekannt ist aber nur der Wert $n=p \cdot q$. Der Rückschluss von n auf die beiden Faktoren – die so genannte Faktorerlegung – ist ein enorm zeitaufwendiges Verfahren.

Solange niemand eine schnelle Methode zur Suche nach den Primfaktoren findet – ein Problem, an dem die Mathematiker seit mehr als 2000 Jahren arbeiten – ist RSA ein sicheres Verfahren. Sicherheitskritische Anwendungen basieren auf Zahlen größer als 10^{300} . Die weiteren Fortschritte in der Rechengeschwindigkeit können durch Erhöhung von n leicht kompensiert werden.

Die Berechnung von C^d weist eine interessante Symmetrie bezüglich e und d auf. In dem Produkt spielt die Reihenfolge keine Rolle. Man kann also durchaus auch mit dem geheimen Schlüssel d verschlüsseln und dann mit dem öffentlichen Schlüssel e den Vorgang umkehren.

Damit ist keine Vertraulichkeit möglich, da jeder Zugang zu dem öffentlichen Schlüssel hat. Aber diese Eigenschaft kann dafür zum Signieren von Nachrichten genutzt werden. Wenn eine Nachricht zu dem öffentlichen Schlüssel passt, muss

sie mit dem zugehörigen privaten Schlüssel erzeugt worden sein. Damit kann nur der Besitzer des privaten Schlüssels der Absender sein.

13.4 Anwendungen

Ausgehend von den Algorithmen lassen sich sichere Anwendungen realisieren. Für die Praxis spielen neben den theoretischen Eigenschaften auch Fragen wie der benötigte Rechenaufwand oder die sichere Verwaltung der Schlüssel eine entscheidende Rolle. So hat RSA bei allen Vorzügen den Nachteil eines relativ hohen Rechenbedarfs. Damit ist es nicht praktikabel, alle Nachrichten mit RSA zu schützen. Weiterhin sind beim Einsatz der Algorithmen rechtliche Aspekte wie etwa die Patentsituation oder Ausfuhrgesetze zu beachten.

13.4.1 PGP

Eine freie, weit verbreitete Software für die Verschlüsselung von E-Mails ist PGP (Pretty Good Privacy) [27]. Phil Zimmermann entwickelte mit PGP ein einfach zu bedienendes Programm, das mehrere Verschlüsselungsverfahren kombiniert. Die eigentliche Nachricht verschlüsselt ein rechenzeitunkritisches, symmetrisches Verfahren wie DES. Dazu nutzt es einen einmaligen Schlüssel, der asymmetrisch mit dem öffentlichen Schlüssel des Empfängers geschützt ist. Nur der Empfänger kann mit seinem privaten Schlüssel den Einmalschlüssel aus der E-Mail dekodieren und damit die gesamte Nachricht entschlüsseln.

Der Sender fügt zudem noch eine mit seinem eigenen privaten Schlüssel kodierte Signatur an, die eine Prüfsumme über die Nachricht enthält. Damit kann der Empfänger sicherstellen, dass die Nachricht tatsächlich vom angegebenen Sender stammt und kein Angreifer den Text während des Transports verändert hat. In die Anwendung PGP integriert ist die Verwaltung der öffentlichen Schlüssel in einem so genannten Key-Ring (Schlüsselbund). In diesem sind die Schlüssel nach ihrer Vertrauenswürdigkeit bewertet.

13.4.2 Sichere Transportschicht

Eine flexible Möglichkeit für verschiedenste sicherheitskritische Anwendungen wird durch eine sichere Transportschicht bereitgestellt. Am Beispiel von HTTP wird ein zusätzliches Protokoll zwischen TCP und Anwendungsprotokoll eingeschoben. Das Protokoll verwendet die Funktionalitäten von TCP und ergänzt sie um eine entsprechende Sicherung.

Aus Sicht der Anwendung ändert sich nichts, außer dass jetzt die Transportschicht sicher ist. Zur Kennzeichnung hängt man an die Abkürzung des Anwendungsprotokoll den Buchstaben S für Secure an (zum Beispiel HTTPS für HTTP Secure).

Tabelle 13.1: Sichere Transportschicht im Protokollstapel

HTTPS	Anwendung
Sichere Transportschicht	Transport
TCP	
IP	Netzwerk
...	

Der IETF-Standard dazu ist TLS (Transport Layer Security), eine Weiterentwicklung des ursprünglich von der Firma Netscape Communications entwickelten Protokolls SSL (Secure Sockets Layer). Der Ablauf entspricht den bereits besprochenen Grundprinzipien. Der Client benutzt den öffentlichen Schlüssel des Servers, um einen so genannten Sessionkey zu übermitteln. Dieser Sessionkey ist ein einmaliger Schlüssel, gültig nur für die aktuelle Verbindung, zur symmetrischen Verschlüsselung der Daten.

Prof. Dr. Stephan Euler

tecCHANNEL-Links zum Thema	Webcode	Compact
Datensicherheit und Verschlüsselung	a1503	S.139
RSA im Detail	a1504	S.153
Kryptographie-Grundlagen	a416	–
Security im Überblick	p1095	–
Praxis der digitalen Signatur	a909	–

Mit den Webcodes gelangen Sie auf www.tecChannel.de direkt zum gewünschten Artikel. Geben Sie dazu den Code in das Feld „Webcode“ in der Titelleiste von www.tecChannel.de ein. Mit „p“ gekennzeichnete Webcodes beziehen sich auf Online-Artikel aus tecCHANNEL-Compact. Es ist möglich, dass Artikel zunächst exklusiv in tecCHANNEL-Compact veröffentlicht werden und daher erst später über den Webcode erreichbar sind.

14. RSA im Detail

Ausgangspunkt ist die Eulersche φ -Funktion (engl. totient function). Für eine natürliche Zahl n bezeichnet $\varphi(n)$ die Anzahl von Zahlen kleiner n , die teilerfremd zu n sind. Teilerfremd bedeutet, dass zwei Zahlen keinen gemeinsamen ganzzahligen Faktor enthalten. Als Beispiel betrachten wir die Zahl 10 mit den beiden Faktoren 2 und 5. Dann ergibt $\varphi(10)=4$, da nur die Zahlen 1, 3, 7 und 9 teilerfremd zu 10 sind. Alle anderen Zahlen enthalten ebenfalls einen der beiden Teiler.

Für zwei natürliche Zahlen a und m , die ebenfalls teilerfremd sind, gilt der Satz von Euler (auch als Satz von Euler-Fermat bezeichnet)

$$a^{\varphi(m)} \bmod m = 1 \quad (14.1)$$

wobei die Modulo-Funktion $\bmod$ den Rest bei ganzzahliger Division angibt. Gleichung (14.1) besagt demnach, dass bei der Division

$$a^{\varphi(m)} / m$$

der Rest 1 übrig bleibt. Anders formuliert, es gibt eine natürliche Zahl L , die die Bedingung

$$a^{\varphi(m)} \bmod m = L \cdot m + 1 \quad (14.2)$$

erfüllt. Der Exponent kann mit einer weiteren natürlichen Zahl k in der Form

$$a^{k \cdot \varphi(m) + 1} \bmod m = a \bmod m \quad (14.3)$$

erweitert werden. Die Gleichheit lässt sich mit der Rechenregel

$$(a \cdot b) \bmod n = [(a \bmod n) \cdot (b \bmod n)] \bmod n \quad (14.4)$$

leicht zeigen. Ausführlich geschrieben erhält man

$$a^{\varphi(m)} \cdot \dots \cdot a^{\varphi(m)} \cdot a \bmod m \quad (14.5)$$

und nach 14.4

$$[(a^{k \cdot \varphi(m)} \bmod m) \cdot \dots \cdot (a^{k \cdot \varphi(m)} \bmod m) \cdot (a \bmod m)] \bmod m \quad (14.6)$$

Nach dem Satz von Euler sind die Exponenten alle gleich 1, so dass sich in der Tat 14.3 auf $a \bmod m$ reduziert.

Der Ausdruck $k \cdot \varphi(m)$ kann als $1 \bmod \varphi(m)$ gelesen werden. Bei der ganzzahligen Division durch $\varphi(m)$ bleibt gerade der Rest 1 übrig. Im RSA-Algorithmus werden daher die Zahlen e und d genau so gewählt, dass sie die Bedingung $e \cdot d = 1 \bmod \varphi(m)$ erfüllen. Damit stellen sich zwei Probleme:

- Welchen Wert hat $\varphi(m)$?
- Wie kann bei gegebenen e der Wert von d berechnet werden?

Für beliebige Werte von n ist die Berechnung von $\varphi(n)$ zeitaufwändig. Allerdings gibt es einige Sonderfälle. Insbesondere gilt für jede Primzahl p die Regel $\varphi(p) = p - 1$, denn eine Primzahl enthält per Definition keine ganzzahligen Faktoren. Weiterhin lässt sich zeigen, dass für das Produkt zweier Primzahlen p und q die einfache Beziehung

$$\varphi(p \cdot q) = \varphi(p) \cdot \varphi(q) = (p - 1) \cdot (q - 1) \quad (14.7)$$

gilt. Wenn man also m als Produkt zweier Primzahlen wählt, so reduziert sich die Berechnung von $\varphi(m)$ auf die Multiplikation zweier Zahlen. Bleibt die Bestimmung von d . Es gilt

$$e \cdot d = 1 \bmod \varphi(m) = k \cdot \varphi(m) + 1 \quad (14.8)$$

und damit

$$e \cdot d - k \cdot \varphi(m) = 1 \quad (14.9)$$

Es gilt dann, zwei ganze Zahlen d und k zu finden, die diese Beziehung erfüllen. Die Zahl k hat für RSA keine Bedeutung, aber d ist der gesuchte private Schlüssel. Die Lösung von (14.9) basiert auf dem Euklidischen Algorithmus zur Bestimmung des größten gemeinsamen Teilers (ggT) zweier Zahlen.

Berechnet wird die größte ganze Zahl, die beide Zahlen ohne Rest teilt. Das Verfahren beruht darauf, dass ein gemeinsamer Teiler d zweier Zahlen x und y auch deren Differenz teilt:

$$(x - y) / d = x/d - y/d = k_x - k_y \quad (14.10)$$

Da sowohl k_x als auch k_y ganze Zahlen sind, ist auch die Differenz wieder eine ganze Zahl. Um den ggT zu ermitteln, werden im Euklidischen Algorithmus iterativ Differenzen gebildet, wobei immer der kleinere Wert vom größeren abgezogen wird. Das neue Zahlenpaar $x - y$, y hat auf Grund von (14.10) wieder den gleichen ggT. Irgendwann wird ein Wert 0 erreicht und der verbleibende Wert ist dann der ggT von x und y . Man kann das Verfahren beschleunigen, in dem man den kleineren Wert in einem Schritt so oft wie möglich abzieht und nur den verbleibenden Rest weiter verwendet. Dies entspricht wiederum einer Modulo-Operation. Eine beispielhafte Implementierung als Methode in Java hat folgende Form:

```
public static int ggt(int x, int y) {
    if( y > x ) return ggt( y, x);    // x > y sichern
    while( y > 0 ) {
        System.out.println( "x,y = " + x + " " + y);
        int r = x % y;
```

```

        x = y;
        y = r;
    }
    return x;
}

```

Mit den Werten 696 und 532 als Beispiel ergibt sich

```

x, y = 696 532
x, y = 532 164
x, y = 164 40
x, y = 40 4
ggT = 4

```

Die Methode berechnet in 5 Iterationen den ggT 4. In jedem Schritt wird der alte Wert von y nach x geschoben. Als neuer Wert für y wird der Rest der Division x/y eingesetzt. Dieser Rest kann damit in der Form $x - k \cdot y$ geschrieben werden, wobei k das Ergebnis der ganzzahligen Division von x durch y ist. Betrachten wir die Folge der entstehenden Restwerte. Der erste Wert ist

$$y_1 = x - k_1 \cdot y$$

Dann berechnet sich der zweite Wert als

$$\begin{aligned}
 Y_2 &= x_1 - k_2 \cdot y_1 \\
 &= y - k_2 \cdot (x - k_1 \cdot y) \\
 &= -k_2 \cdot x + (1 + k_1 k_2) \cdot y
 \end{aligned}$$

Der Wert y_2 kann als eine Summe der Form $a \cdot x + b \cdot y$ geschrieben werden. Die Koeffizienten a und b entstehen durch Addition und Multiplikation von ganzen Zahlen und sind damit selbst auch wiederum ganze Zahlen.

Diese Argumentation lässt sich auf die weiteren Restwerte verallgemeinern, so dass insbesondere auch für den letzten Restwert – den ggT – die Darstellung

$$\text{ggT} = a \cdot x + b \cdot y \quad a, b \in \mathbb{Z}$$

resultiert. Damit haben wir den Ansatz zur Lösung der Beziehung (14.9). Wir haben e so gewählt, dass der ggT von e und $\varphi(m)$ gerade den Wert 1 hat. Damit ist sicher gestellt, dass ganzzahlige Lösungen für d und k existieren. Die tatsächlichen Werte lassen sich mit einer Erweiterung des Euklidischen Algorithmus bestimmen. Im Prinzip wird dabei die oben skizzierte Berechnung der Koeffizienten aus den Vorgängerwerten vorgenommen.

Insgesamt ist damit ein einfacher Weg zur Berechnung des privaten Schlüssels eröffnet. Die Funktion $\varphi(m)$ wird als einfaches Produkt zweier Zahlen berechnet und dann kann mit dem erweiterten Euklidischen Algorithmus aus $\varphi(m)$ und dem öffentlichen Schlüssel e der private Schlüssel d berechnet werden.

Implementierung

Nach der Darstellung der mathematischen Grundlagen sollen noch einige Aspekte der Implementierung betrachtet werden. Dazu dient die in Java realisierte Klasse RSADemo. Beim Aufruf kann man die Werte für p , q und e vorgeben. Das Programm berechnet dann das zugehörige d und bestimmt für alle Zeichen den verschlüsselten Wert. Die Implementierung des erweiterten Euklidischen Algorithmus verwendet eine rekursive Methode erweiterter Euklid. Diese Methode ruft sich immer wieder auf, bis der ggT erreicht ist. Dann werden davon ausgehend die Werte von u und v ermittelt.

Eine praktische Schwierigkeit besteht in der Berechnung der Potenzen. Zur Verschlüsselung muss die Potenz

$$C = M^e \bmod n \quad (14.11)$$

berechnet werden. Die direkte Berechnung führt schnell zu einem Überlauf durch zu große Zahlen. Glücklicherweise gibt es eine einfache Methode, den Überlauf zu vermeiden. Betrachten wir dazu die Darstellung

$$M = k \cdot n + r$$

bei der M in ein ganzzahliges Vielfaches von n und den Rest r aufgespalten wird. Dann gilt

$$\begin{aligned} M^e &= M^{e-1} \cdot M \\ &= M^{e-1} \cdot (k \cdot n + r) \\ &= M^{e-1} \cdot kn + M^{e-1} \cdot r \end{aligned}$$

Der erste Summand ist wiederum ein ganzzahliges Vielfaches von n und spielt damit bei der Modulo-Bildung keine Rolle. Nutzt man weiterhin die Beziehung $r=M \bmod n$, so kann man schreiben

$$M^e \bmod n = M^{e-1} \cdot M \bmod n \quad (14.12)$$

Die Berechnung von M^e lässt sich damit auf die Bestimmung von $r = M \bmod n$ und M^{e-1} zurückführen. Die gleiche Argumentation kann auf die Auswertung von M^{e-1} angewandt werden. Insgesamt ergibt sich dann eine Folge von Multiplikationen und Restwertbildungen, bei denen die Zahlenwerte im darstellbaren Bereich bleiben. Die Methode `aHochbModn` stellt eine entsprechende Implementierung dar. Es sei angemerkt, dass die Berechnung wesentlich beschleunigt werden kann, wenn man anstelle der fortgesetzten Multiplikation mit Quadrieren arbeitet. Die Anzahl der benötigten Rechenoperationen wächst dann nur noch linear mit der Anzahl der Bits des Exponenten.

```

public class RSADemo
{
    public static int u, v;

    private static int erweiterterEuklid(int a, int b) {
        int g;
        if (b==0) {
            g=a;
            u=1;
            v=0;
        } else {
            g = erweiterterEuklid(b, a%b);
            int t=u-a/b*v;
            u=v;
            v=t;
        }
        return g;
    }

    public static int aHochbModn( int a, int b, int n ) {
        int res = a % n;
        for( int i=0; i < b-1; i++ ) {
            res = (res * a) % n;
        }
        return res;
    }

    public static void main( String[] args) {
        int p = 17;
        int q = 11;
        int e = 9;

        for( int i=0; i < args.length; i++ ) {
            if( args[i].equals("-p") ) {
                p = Integer.parseInt( args[++i] );
            } else if( args[i].equals("-q") ) {
                q = Integer.parseInt( args[++i] );
            } else if( args[i].equals("-e") ) {
                e = Integer.parseInt( args[++i] );
            } else {
                System.out.println("unknown Argument: " +
➡args[i]);
                System.exit(1);
            }
        }

        int n = p*q;
        int ggt = erweiterterEuklid( (p-1)*(q-1), e);
        System.out.println( "ggt = " + ggt);
    }
}

```

```

    if( ggt != 1 ) {
        System.out.print( "e="+e+" und (p-1)*(q-1)="+
        ➡(p-1)*(q-1) );
        System.out.println( " sind nicht teilerfremd!");
        System.exit(1);
    }
    System.out.println( "u,v: " + u + " " + v);
    int d = v;
    if( d < 0 ) d += (p-1)*(q-1); // negative Werte
    ➡korrigieren

    // Schleife über alle Zeichen
    for( int m = 0; m < n; m++ ) {
        System.out.print( "Zeichen = " + m);
        int c = aHochbModn( m, e, n );
        System.out.print( " G = " + c);
        System.out.println( " D = " + aHochbModn(c,d,n) );
    }
}

```

Prof. Dr. Stephan Euler

tecCHANNEL-Links zum Thema	Webcode	Compact
RSA im Detail	a1504	S.153
Listings zum Download	a1541	–
Datensicherheit und Verschlüsselung	a1503	S.139
Kryptographie-Grundlagen	a416	–

Mit den Webcodes gelangen Sie auf www.tecChannel.de direkt zum gewünschten Artikel. Geben Sie dazu den Code in das Feld „Webcode“ in der Titelleiste von www.tecChannel.de ein. Mit „p“ gekennzeichnete Webcodes beziehen sich auf Online-Artikel aus tecCHANNEL-Compact. Es ist möglich, dass Artikel zunächst exklusiv in tecCHANNEL-Compact veröffentlicht werden und daher erst später über den Webcode erreichbar sind.

15. Netzwerk-Sniffer

Mit Netzwerk-Sniffen kann man Fehler im LAN aufspüren, aber auch Programme, die Daten über den ahnungslosen Benutzer versenden. tecCHANNEL stellt im folgenden die günstigsten Lösungen und deren Anwendung vor.

So genannte „Phone Home“-Software macht immer mehr von sich reden. Programme also, die Daten über den Benutzer ausspähen und unbemerkt per Internet an den Hersteller weiterleiten. Weil das Ganze ohne Wissen der Anwender erfolgt, fühlen sich diese zu Recht ausspioniert. Dazu kommen noch böswillige Programme, die auf dem Rechner des Opfers Hintertürchen, Proxies, Spambots oder DDoS-Tools einrichten.

Mit Personal Firewalls (**webcode: a1483**) wie Zone Alarm lässt sich zwar feststellen, ob eine Applikation heimlich Daten überträgt oder ein Angreifer von außen Kontakt herstellen will. Allerdings bleibt damit noch im Dunkeln, welche Daten tatsächlich fließen. Außerdem hilft eine Personal Firewall nicht bei Tools, die ohnehin dazu dienen, Daten über das Internet zu senden oder zu empfangen. Instant Messenger (**webcode: a763**) oder Webbrowser (**webcode: a479**) könnten ungestört Daten übermitteln, ohne dass die Personal Firewall Alarm schlägt.

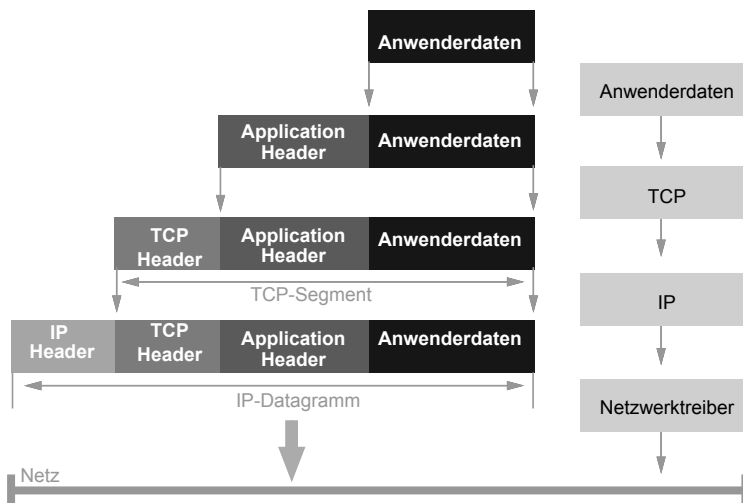
Um bei solchen Tools Unregelmäßigkeiten in der Datenübertragung auszumachen, muss man schwerere Geschütze auffahren und analysieren, welche Daten an welche Rechner geschickt werden. So genannte Netzwerk-Sniffer (Schnüffler) klinken sich in den Datenverkehr ein und protokollieren alle gesendeten und empfangenen Pakete akribisch mit. Stück für Stück kann man dann den Datenstrom mit einem Analyse-Tool auseinander nehmen und so herausfinden, ob ein Programm unerlaubt Daten sendet.

15.1 Sniffing-Grundlagen

Ein Netzwerk-Sniffer besteht aus einer Reihe von Komponenten. Der so genannte Capture Driver klinkt sich in den Treiber der Netzwerkkarte oder des DFÜ-Adapters ein und sorgt dafür, dass alle gesendeten und empfangenen Pakete in einem Buffer zwischengespeichert werden.

Über eine Reihe von Filtern kann der Sniffer sich auf Pakete beschränken, die bestimmten Kriterien wie etwa Netzwerkprotokoll, Zieladresse oder TCP-Port genügen. Da beispielsweise bei einem LAN mit 100 Mbit/s bis zu 144.000 Pakete pro Sekunde ankommen können, ist in solchen Netzen ein schneller Rechner zum Sniffen erforderlich.

Eine Analyse-Komponente untersucht je nach Anwendungszweck die Pakete auf Fehler bei der Datenübertragung oder auf das Vorhandensein von Hackern. Mögliche Angriffe werden anhand bestimmter Muster wie beispielsweise Portscans entdeckt. Bei kommerziellen Sniffen erfolgt die Analyse in Echtzeit. Das setzt natürlich ein entsprechend leistungsfähiges System voraus.



© tecCHANNEL

Bild 15.1: Beim Versenden von Netzwerkpaketen fügt jede Schicht ihre Kontrolldaten hinzu. Der Sniffer muss diese Daten wieder aufschlüsseln.

Der wichtigste Schritt ist die Dekodierung der Netzwerkpakete. Dabei bereitet der Sniffer die Pakete so auf, dass der Administrator die einzelnen Bestandteile unterscheiden kann, ohne genau zu wissen, welche Bytes beispielsweise bei einem TCP/IP-Paket für den Zielport zuständig sind.

Komfortablere Systeme suchen auch per DNS-Abfrage den Hostnamen zu IP-Adressen (**webcode: 209p17**) und bauen nötigenfalls mehrere Pakete wieder zu einem logischen Datenstrom zusammen. Das erspart dem Administrator auch das mühsame händische Zusammensuchen der jeweiligen Sequenznummern in diesem Datenstrom sowie die Zuordnung der einzelnen Pakete beim Three-way-handshake (**webcode: 209p27**).

15.2 Sniffing-Tools

Normalerweise finden Network-Sniffer oder Protocol Analyzer in großen Netzen Verwendung. Sie sollen dem Administrator dabei helfen, Schwachstellen aufzudecken oder Fehler zu finden. Dementsprechend sind diese Applikationen auch mit einer Vielzahl von Features und Verwaltungsmöglichkeiten ausgestattet, wie beispielsweise die Aufschlüsselung des Datenverkehrs nach Protokollen und Sender/Empfänger. Trendanalysen und Tools zur automatischen Fehlererkennung gehören ebenfalls oft zum Standardumfang.

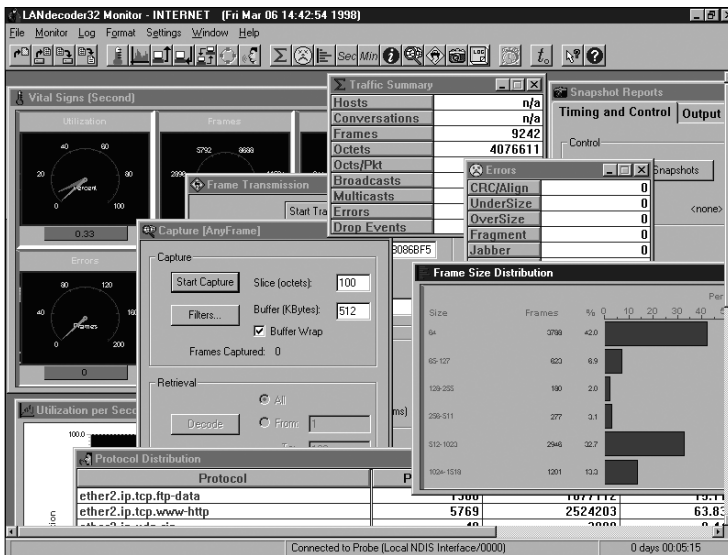


Bild 15.2: Professionelle Sniffer wie etwa der LANDecoder32 bieten zusätzliche Features wie Trendanalyse, kosten aber auch entsprechend.

Diese Funktionsvielfalt hat ihren Preis. Ein leistungsfähiges Programm zur Protokollüberwachung und -analyse mit verteilten Agenten kostet etwa 5000 Euro. Das geht natürlich weit über das hinaus, was man für ein Homeoffice oder ein kleines Büronetzwerk benötigt.

Um zu untersuchen, ob ein Programm unerlaubt Daten sendet und welche, sind umfangreiche Analyse-Tools jedoch nicht notwendig. Auch die Dekodierung muss nicht unbedingt jedes verfügbare exotische Protokoll wie etwa DecNet unterstützen. Da die großen Hersteller aber keine abgespeckte Version ihrer teuren Management-Suites im Angebot haben, kommen immer mehr Free- und Shareware-Tools auf den Markt, die genau dieses Segment abdecken.

15.3 Analyzer 3.0alpha

Von der Politechnischen Universität Turin (www.polito.it/index.en.html) stammen der kostenlose Analyzer und der zugehörige Capture-Treiber WinPCap (<http://winpcap.polito.it/>). Die aktuelle als „stable“ bezeichnete Version ist zwar die 2.2, allerdings empfehlen die Entwickler selbst den Einsatz der Version 3.0alpha. Die neueste Version von WinPCap unterstützt jetzt auch das so genannte Remote-Capture. Damit kann man die Daten von mehreren Stationen einsammeln und somit ein Bild des gesamten Netzwerk-Traffics erhalten. Das ist vor allem in geschwächten Umgebungen notwendig.

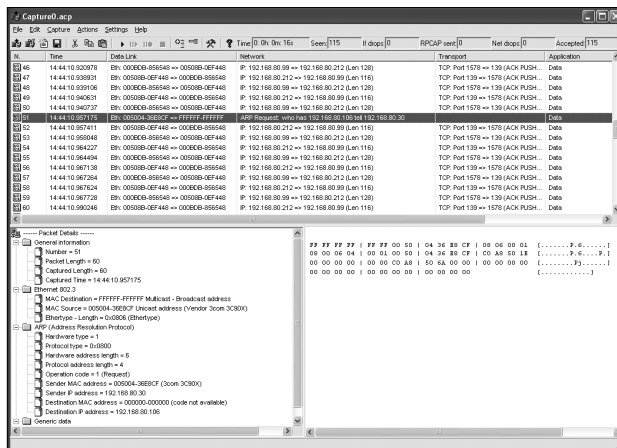


Bild 15.3: Der Freeware-Analyzer dekodiert eingehende Pakete und stellt sie übersichtlich nach Protokoll-Ebenen dar.

Bei den Grundfunktionen Capturing und Dekodierung bietet das Tool immerhin ein Multiple-Document-Interface (MDI), so dass sich mehrere Datenströme parallel betrachten und vergleichen lassen. Außerdem können erfahrene Anwender über die Konfigurationsdateien eigene Protokoll- und Paketformate einstellen. Der Analyzer verwendet dabei einen eigens entwickelten XML-Dialekt: NetPDL (<http://analyzer.polito.it/30alpha/docs/dissectors/NetPDLSpec.htm>).

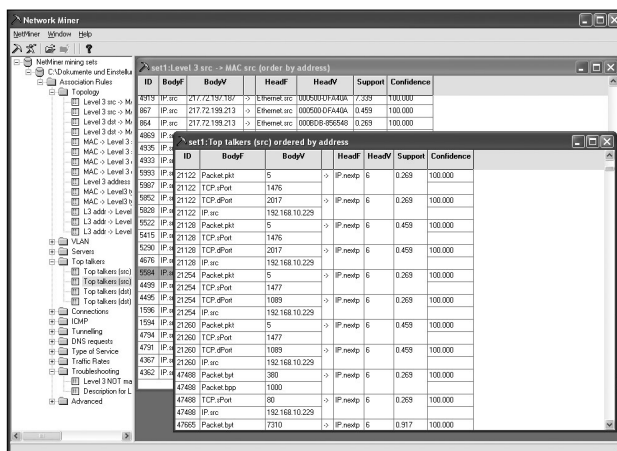


Bild 15.4: Über den Netminer lassen sich Datenströme statistisch analysieren. Allerdings benötigt die Aufbereitung der Daten selbst bei kleinen Datenmengen extrem viel Zeit.

Bei der Definition von Filtern verlässt sich der Analyzer komplett auf die Syntax von tcpdump (http://winpcap.polito.it/docs/docs31beta4/html/group__language.html), die sich dem Benutzer erst auf den zweiten Blick erschließt. Dafür bieten die Filter auch alle Möglichkeiten, so beispielsweise die Abfrage auf bestimmte Bits im TCP-Header.

Neu dazugekommen sind zwei Tools. Das „Network Logger Module“ protokolliert Netzwerk-Sessions mit und speichert die wichtigsten Informationen darüber in einer Datenbank. Hier kann der Benutzer sich eine Reihe von Statistiken anzeigen lassen, etwa die Netzwerkauslastung oder eine Verteilung nach Protokolltyp. Mit dem „Network Miner Module“ lassen sich klassische Datamining-Verfahren auf die vom Netlogger erstellten Datenbanken anwenden, etwa um Anomalien herauszufinden.

Fazit: Für die einfachen Aufgaben reicht der kostenlose Analyzer vollkommen aus. Wer das Tool jedoch ausreizen will, muss sich sehr intensiv mit den einzelnen Protokollen beschäftigen.

Tabelle 15.1: Quickinfo Analyzer 2.1

Hersteller	Politecnico di Torino (http://netgroup-serv.polito.it/analyzer/)
Preis	Freeware
Demoversion	http://analyzer.polito.it/30alpha/download.htm , unbegrenzt, 5,2 MByte
Hardware	Pentium III und 128 MByte RAM
Software	Windows 9x/Me/2000/XP

15.4 CommView 4.1

Bei CommView 4.1 von Tamosoft (<http://www.tamos.com/>) handelt sich um einen kostengünstigen Sniffer mit einer ganzen Reihe von hilfreichen Features. So kann das Tool, das für den Privatanutzer 129 Dollar kostet, neben dem Sniffing und der Protokollanalyse auch viele Statistiken über den Netzwerk-Traffic ausgeben und die verfügbare Bandbreite überwachen.

Praktisch ist die Fähigkeit, TCP-Sessions wieder zusammenzusetzen, also einen Datenstrom, der in mehrere IP-Pakete aufgeteilt ist, als Gesamtübersicht darzustellen. Dabei gibt CommView die Daten entweder als Text, als HTML oder als Hex aus. Bilder kann der Sniffer dagegen nicht anzeigen.

Bei der Suche nach Hosts, mit denen Kontakt aufgenommen wird, unterstützt CommView den Anwender durch das IP-Statistik-Fenster. Hier gruppiert das Tool alle Pakete nach Quell- und Zieladresse. Damit erhält man einen ersten Überblick über die geflossenen Datenströme.

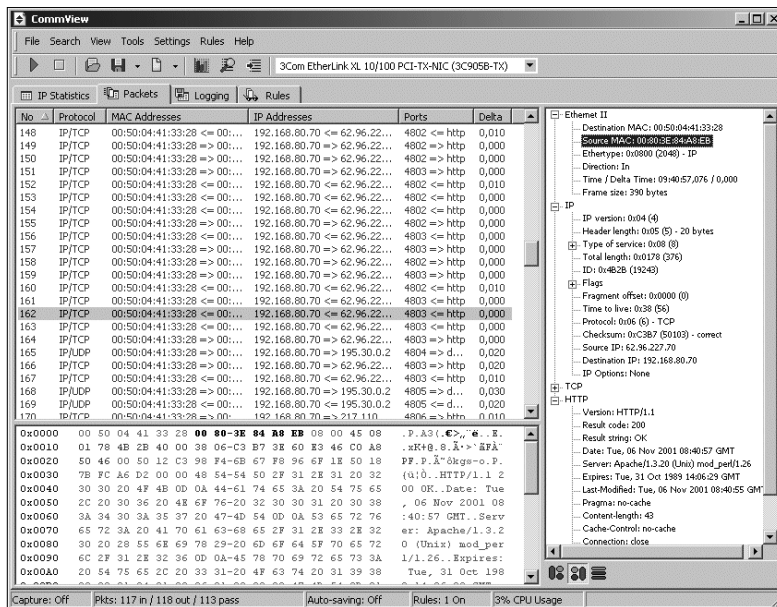


Bild 15.5: CommView bietet eine ausführliche Aufschlüsselung der gesammelten Pakete.

CommView-Filterdefinition

Die Filterdefinition von CommView ist sehr gut gelungen. Man kann nach IP- und MAC-Adresse filtern, nach Ethernet- und IP-Protokollen sowie nach TCP-Flags und -Ports. Dabei lässt sich jeweils festlegen, ob nur ein- oder ausgehende Pakete berücksichtigt werden oder beide Richtungen. Besonders hervorzuheben ist das Filtern nach frei definierbaren Texten. Das belastet zwar die CPU erheblich, ermöglicht aber eine sehr flexible Auswahl der Pakete.

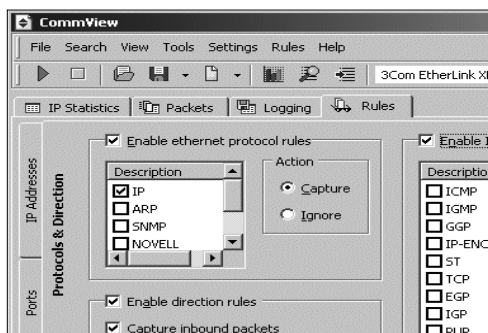


Bild 15.6: CommView bietet eine sehr flexible und dennoch einfach zu handhabende Filterfunktion.

Nützlich ist die Integration von SmartWhois, einem weiteren Programm von TamoSoft. Damit lassen sich detaillierte Informationen über die Hosts einholen. Über den CommView Remote Agent, der noch einmal mit 149 Dollar zu Buche schlägt, lassen sich auch die Daten von mehreren Stationen aggregieren.

Wer detaillierte Auskünfte über sein WLAN will, muss zur WiFi-Version greifen, die allerdings stolze 499 Dollar kostet. An zusätzlichen Informationen bietet sie jedoch lediglich Daten über gefundene WLANs, Auslastung der einzelnen Channels und Ähnliches. Klassisches Packet-Capturing können Sie auch mit der Standardversion erledigen.

Fazit: Insgesamt ist CommView 4.1 seine 129 US-Dollar auf jeden Fall wert, bietet es doch Funktionen, die sonst nur teuren Sniffern vorbehalten sind. Wer sein LAN im Auge behalten möchte oder auch nur Sorge hat, dass „Phone Home“-Software heimlich Daten übermittelt, ist mit CommView gut bedient.

Tabelle 15.2: Quickinfo CommView 4.1

Hersteller	Tamosoft (www.tamos.com)
Preis	Privatlizenz 129 US-Dollar
Demoversion	http://www.tamos.com/download/main/ , 30 Tage Laufzeit, 2,5 MByte
Hardware	Pentium III und 128 MByte RAM
Software	Windows 9x/Me/2000/XP/2003

15.5 Erste Schritte mit Sniffing

Am Beispiel des CommView zeigen wir die ersten Schritte zu einer erfolgreichen Überwachung des lokalen Netzwerkverkehrs, etwa um eine „Phone Home“-Software zu entlarven.

Je nachdem, ob der Rechner in einem lokalen Netz hängt oder per DFÜ-Verbindung mit dem Internet verbunden ist, sollten Sie zunächst einige grundlegende Filter definieren, die nur den Internet-Traffic des eigenen Rechners akzeptieren. Ansonsten müssen Sie sich aus einem Wust von Protokollen und Broadcasts die Pakete herausuchen, die von Ihrem System ins Internet gehen.

Bei CommView ist dafür der Haupttreiber „Rules“ zuständig. Unter Protocols & Direction selektieren Sie alle IP-Pakete, die an unseren Rechner gerichtet sind oder von ihm gesendet werden.

Zusätzlich setzen Sie unter „IP Addresses“ die IP-Adresse der lokalen Maschine als Ziel oder Quelle. Bei bestimmten Verdachtsmomenten hinsichtlich verwendeter Ports kann man die Filter weiter eingrenzen, um das Datenaufkommen in Grenzen zu halten. Hier ist das nicht der Fall.

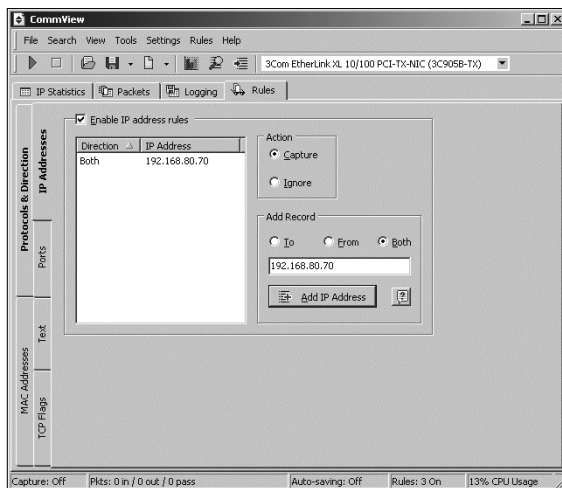


Bild 15.7: CommView berücksichtigt nur IP-Pakete, die von der zu überwachten Station kommen oder dediziert an sie gesendet werden.

Outlook überwacht

Nach der Definition der Filter starten Sie das Capturing und danach die verdächtige Applikation, in diesem Beispiel Outlook 2002.

Local IP	Remote IP	In	Out	Direction	Sessions	Ports	Hostname	Bytes
192.168.80.70	192.168.80.212	73	66	Out	0	netbios-s...	VORWERK	69,157
192.168.80.70	Nameserver1	7	7	Out	0	domain	ns2.Space.Net	1,357
192.168.80.70	192.168.80.255	0	1	Out	0	netbios-dgm		243
192.168.80.70	217.72.192.134	14	15	Out	1	995	pop3.web.de	3,883
192.168.80.70	213.165.64.20	11	10	Out	1	pop3	mail.gmx.net	1,429
192.168.80.70	192.168.10.229	1	1	In	0	4408	renner	114

Bild 15.8: Die IP Statistik zeigt keine auffälligen Verbindungen.

Outlook 2002 produziert im Beispiel keine verdächtigen Verbindungen. Es nimmt Kontakt auf für einen DNS-Query und dann zu den Mailservern. Dabei nutzt es einmal den Standard POP3-Port und einmal den Port 995 (POP3 mit SSL).

Bei der ungesicherten Verbindung über den POP3-Port fällt bei der Paketanalyse gleich auf, dass das Passwort im Klartext übertragen wird: Ein erhebliches Sicherheitsrisiko, da im Netz jeder einen Sniffer benutzen und so die Passwörter sammeln könnte. Sehr sicherheitsbewusste Administratoren setzen deshalb einen Sniffer auf POP3-Verbindungen an, damit sie ihre Benutzer vor diesem Risiko warnen können. Ähnlich sensitive Protokolle sind FTP, Telnet oder HTTP.

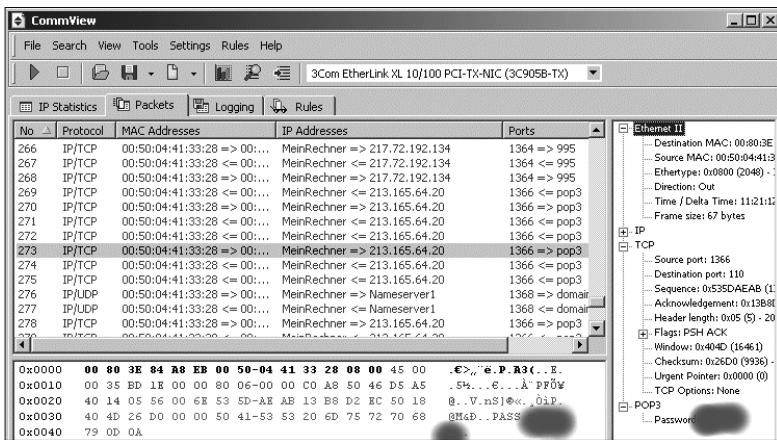


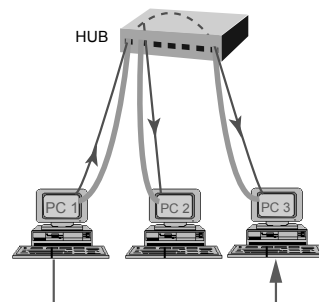
Bild 15.9: Bei POP3 werden Passwörter ungeschützt im Klartext übertragen.

15.6 Sniffing im Netzwerk

Wenn es darum geht, den Datenverkehr des lokalen Rechners zu überwachen, ist das Sniffing noch ziemlich einfach. Der Capture Driver muss sich lediglich in den Treiber der NIC einklinken. Um den Traffic im gesamten Ethernet-Segment zu analysieren, müssen einige zusätzliche Vorkehrungen getroffen werden.

Ethernet ist von der Grundidee her ein Shared Medium. Das heißt unter anderem auch, dass alle Pakete an alle Stationen gesendet werden. Damit nun nicht die einzelnen Rechner ständig damit beschäftigt sind, die für sie bestimmten Pakete herauszusuchen, ist bereits in der Netzwerkkarte ein Filter eingebaut. Dieser leitet nur solche Pakete an den Treiber weiter, die speziell an die MAC-Adresse der NIC gerichtet sind oder bei denen es sich um Broad- oder Multicasts handelt.

Bild 15.10: Ein Hub sendet alle eingehenden Datenpakete an alle Ports weiter und ermöglicht so eine komplette Überwachung per Sniffer.

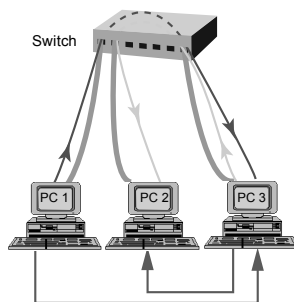


© tecCHANNEL

Laut den von Intel und Microsoft herausgegebenen PC99-Richtlinien für PC Design, sollten Netzwerkkarten jedoch den so genannten „Promiscuous Mode“ unterstützen. Damit kann der Capture Driver die Karte anweisen, alle Netzwerkpakete zu empfangen und an den Treiber weiterzuleiten. Über diesen Weg ist es dann auch möglich, den Datenverkehr in einem Netzwerksegment zu überwachen.

15.7 Sniffing im geschwitchten LAN

Mit der zunehmenden Verbreitung von Switches wird das allerdings immer schwieriger, denn Switches sorgen ja dafür, dass Netzwerkpakete nur noch dediziert an den tatsächlichen Empfänger gesendet werden. Das erreichen diese Geräte, indem sie sich merken, an welchem Port welche MAC-Adresse zu finden ist. Demzufolge „sieht“ auch eine Karte im Promiscuous Mode nur noch Broadcasts und an sie gerichtete Pakete. Das verhindert natürlich eine effektive Fehleranalyse per Netzwerk-Sniffer.



© tecCHANNEL

Bild 15.11: Ein Switch sendet Datenpakete nur an den eigentlichen Empfänger weiter und verhindert so Sniffing von fremden Paketen.

Teure Switches unterstützen deshalb ein Feature namens „Port Mirroring“. Dabei wird der Datenverkehr eines Ports auf einen anderen gespiegelt. An diesen Port hängt man dann den Sniffer und überwacht den Traffic. Meistens wird der Uplink-Port gespiegelt, weil man dann fast den gesamten Datenstrom bekommt.

Bei Switches ohne die Port-Spiegelung muss man zu einem etwas unfeinen Mittel greifen, dem „Switch Jamming“. Dabei schickt man dem Switch eine Vielzahl von Paketen mit verschiedenen MAC-Adressen, bis sein interner Speicher für die Verwaltung überläuft.

Die meisten Switches schalten in diesem Fall um und verhalten sich dann wie ein Hub, senden also alle Pakete an alle Ports. Und genau darin liegt der Nachteil dieses Verfahrens: Während der Analyse degradiert man den Switch zu einem Hub und verliert damit alle Performance-Vorteile.

15.8 Sniffing im WLAN

Da die Datenpakete im WLAN per Funk übertragen werden, kann jeder – so ähnlich wie bei einem Hub – die Pakete mitlesen. Dazu muss nur die WLAN-Karte in den Promiscuous Mode geschaltet werden. Somit sind auch für WLANs keine speziellen Sniffer-Programme notwendig; für eine Analyse des reinen Datenverkehrs reichen die Standardmodelle problemlos aus.

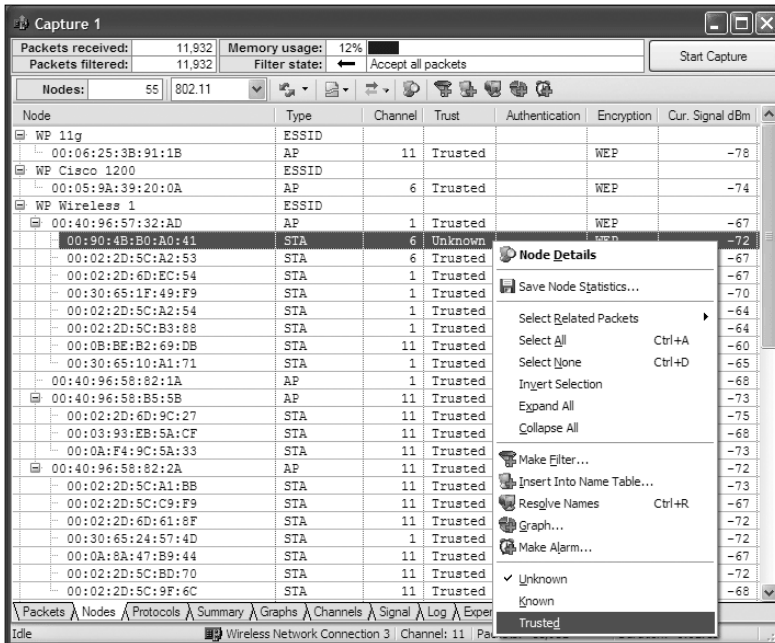


Bild 15.12: Mit Produkten wie Airopeek von WildPackets erhält man eine Reihe WLAN-spezifischer Informationen, die man bei normalen Sniffen nicht bekommt.

Warum also die erheblich teureren Produkte kaufen? Diese können beispielsweise überwachen, ob ungenehmigte Access Points aufgestellt sind, oder irgendwer versucht mitzulauschen. Zudem liefern sie detaillierte Statistiken über Auslastung der Funkkanäle und Fehlermeldungen der Access Points, etwa wenn sie einen Client wegen Überlastung ablehnen. Ein hilfreiches Tool zur Optimierung der WLAN-Konfiguration also.

Mit AirSnare (<http://home.comcast.net/~jay.deboer/airsnare/index.htm>) gibt es ein kostenloses Tool, das den Datenverkehr überwacht und auf unerlaubte MAC-Adressen überprüft. Entdeckt das Tool so eine Station, protokolliert es den verursachten Traffic mit und sendet optional eine Warnmeldung an den Benutzer.

15.9 Lokale Angriffe von Sniffen

Nicht immer haben Sniffer etwas Gutes im Sinn. Oft versuchen Hacker durch das Mitverfolgen des Netzwerkverkehrs, an Passworte oder andere sensitive Informationen zu gelangen. Ziel des Administrators muss es also sein, Sniffing zu verhindern oder zeitnah zu entdecken, um Gegenmaßnahmen einleiten zu können.

Wendet der Angreifer Switch-Jamming an, um an alle Pakete des Netzwerks zu gelangen, erfahren Sie das bei einem verwaltbaren Switch über die Management-Konsole, denn es wird ein entsprechender Eintrag im Log oder im SNMP gemacht. Dann erfahren Sie auch den Port, von dem plötzlich so viele verschiedene MAC-Adressen kommen, dass der interne Puffer überläuft.

Handelt es sich dagegen um einen „dummen“ Switch, können Sie immer noch eine IDS-Station einrichten und den dort eingehenden Netzwerkverkehr analysieren. Kommen plötzlich Netzwerkpakete an dieser Station an, die bei normalem Switch-Betrieb nicht ankommen dürften, muss etwas im Netzwerk faul sein.

Einen etwas eleganteren Weg kann ein Angreifer mittels ARP-Spoofing wählen. Dabei klinkt er sich in die Ethernet-Kommunikation zwischen zwei Stationen ein und agiert dabei parallel als Gateway, damit die beiden Stationen nichts von dem Lauscher mitbekommen.

15.10 Entdecken von Sniffen

Da ein Angreifer sich beim Mitprotokollieren von Netzwerkverkehr meist passiv verhält, kann die Entdeckung nur über Umwege erfolgen. Das gilt besonders für Netzwerke, die mit Hubs verbunden sind, denn da muss der Angreifer überhaupt nicht aktiv werden, um die Daten zu Gesicht zu bekommen.

1. Viele Sniffer-Programme lösen IP-Adressen per Reverse DNS-Lookup gleich zu Host-Namen auf, denn dieser gibt möglicherweise auch Aufschluss über die Aufgabe der Station, etwa mail.domain.de. Eine Häufung von DNS-Traffic lässt sich aber auch entdecken. Einen Schritt weiter kann man gehen, indem man von einer Station aus Datenpakete an einen nicht existierenden Host schickt. Tauchen plötzlich Reverse DNS-Lookups für diese IP-Adresse auf, muss wohl irgendwo eine Station sein, die auf nicht für sie bestimmten Datenverkehr lauscht.

2. Normalerweise werden ARP-Requests an die Ethernet-Broadcast-Adresse (FF:FF:FF:FF:FF:FF) geschickt. Schickt man dagegen den Request an die Adresse FF:FF:FF:FF:FF:FE, sollte eine normale Netzwerkkarte dieses Paket ignorieren. Ist sie aber im Promiscuous Mode, reicht sie das Paket an das Betriebssystem weiter. Linux (zumindest im Kernel 2.4) und Windows überprüfen allerdings nicht alle sechs Bytes der Hardware-Adresse, um herauszufinden, ob es sich um einen Broadcast handelt, sondern nur die ersten Bytes. So kann man mit einer Whohas-Abfrage eine IP-Adresse überprüfen: Antwortet die Station mit dieser IP-Adresse auf die Anfrage mit einem ARP-Reply, muss sie im Promiscuous Mode sein.

3. Über die Messung von Latenzzeiten kann man ebenfalls ermitteln, ob sich eine Station im Promiscuous Mode befindet. Da eine solche Station keinen Hardware-Filter mehr hat, bekommt sie alle Pakete zu sehen und muss sie verarbeiten. Dementsprechend dauern Antworten länger als normal. Zunächst ermittelt eine Messstation die durchschnittliche Antwortzeit des zu überprüfenden Rechners über ICMP-Echo-Anfragen. Danach flutet sie das Netzwerk mit Ethernet-Paketen an nicht vergebene MAC-Adressen und misst parallel die Verzögerungszeit bei erneuten ICMP-Echo-Anfragen an den Zielrechner. Ist die Latenzzeit erheblich höher, liegt das daran, dass der Zielrechner mit der Verarbeitung von nicht für ihn bestimmten Paketen beschäftigt ist. Die Zeitmessung muss allerdings eine Genauigkeit im Mikrosekunden-Bereich aufweisen. Sentinel (www.packetfactory.net/Projects/sentinel/) ist beispielsweise ein Linux-Programm, das diese Verfahren anwendet, um Sniffer zu lokalisieren.

15.11 Fazit

Datenschutzexperten werden angesichts der Möglichkeiten des Sniffings die Hände über dem Kopf zusammenschlagen. Dennoch macht Sniffing Sinn und ist auch notwendig. Einerseits, um Fehler im LAN aufzuspüren und andererseits, um Hacker oder „Phone Home“-Software aufzuspüren. Für Letzteres reichen die hier vorgestellten Programme und Verfahren allemal aus. Es muss also nicht immer die Lösung für 5000 Euro und mehr sein.

Ein weiterer Vorteil der Analyse-Tools: Sie helfen ungemein beim Verständnis der verschiedenen Netzwerkprotokolle. So wird beispielsweise unser Grundlagenbeitrag TCP/IP (**webcode: p209**) um einiges einsichtiger, wenn man parallel mit einem Sniffer den Datenfluss verfolgt.

Dennoch sollten alle Möchtegern-Sniffer an dieser Stelle vorgewarnt sein: Sniffing zum Zwecke der Datenausspähung ist strafbar!

Mike Hartmann

tecCHANNEL-Links zum Thema	Webcode	Compact
Netzwerk-Sniffer	a766	S.159
Ethernet-Grundlagen	a717	–
TCP/IP-Grundlagen	p209	–
So funktioniert HTTP	a208	–
So funktioniert FTP	a207	–
So funktioniert DNS	a205	–
Netzwerk-Starterkits	a619	–
Personal Firewalls	a405	–

16. Übungen und ausgewählte Lösungen

16.1. Übungen Kapitel 2

Übung 2.1

Welche Bandbreite ist für folgende Echtzeitübertragung erforderlich:

- Musik von CD-ROM (75 Minuten Musik auf 650 MByte)
- Video mit Auflösung 320 x 240 Pixel, 1 Byte Farbinformation pro Pixel, 10 Bilder pro Sekunde
- Video mit Auflösung 640 x 480 Pixel, 3 Byte Farbinformation pro Pixel, 30 Bilder pro Sekunde

Wie stark müssen die Daten jeweils komprimiert werden, damit sie in Echtzeit über eine ISDN-Leitung (64 kbit/s) übertragen werden können?

Übung 2.2

In den Fernsehnachrichten wird ein Live-Interview mit dem Korrespondenten in New York gezeigt. Die Übertragung von Bild- und Sprachdaten erfolgt über getrennte Wege:

- Sprache: eine Satellitenverbindung über einen geostationären Satelliten, Gesamtstrecke 70000 km.
- Bild: über ein Unterseekabel, Gesamtstrecke 5000 km.

Berechnen Sie die Ausbreitungsverzögerung für beide Strecken. Welche Laufzeitdifferenz ergibt sich?

Übung 2.3

Eine Verbindung soll nicht mehr als 10 ms Einwegverzögerung haben.

- Wie ist die maximale Länge, wenn nur die Ausbreitungsgeschwindigkeit berücksichtigt wird?
- Wie ist der entsprechende Wert, wenn man zusätzlich die Übertragungsverzögerung bei einer Bandbreite von 1 Mbit/s und einer Paketgröße von 1024 Bytes einrechnet?

Übung 2.4

Sie sollen die Kommunikation mit einem Erkundungsfahrzeug auf dem Mond planen. Die Entfernung beträgt 385.000 km, und die Übertragung soll über eine Bandbreite von 1 Mbit/s erfolgen.

- Wie groß ist die Ausbreitungsverzögerung? Wie groß ist damit die minimale Antwortzeit auf eine Anfrage?

- Wie sind die Werte für die Verbindung mit einem Roboter auf dem Mars (kleinster Abstand zur Erde ungefähr 54 Millionen Kilometer)?
- Berechnen Sie auf der Basis der Ausbreitungsverzögerung das Verzögerungs-Bandbreiten-Produkt für die Verbindung zwischen Erde und Mond.
- Eine Webcam auf dem Mond nimmt Bilder in guter Qualität auf. Ein Bild benötigt 2 MByte Speicherplatz. Wie lange dauert es mindestens, bis nach der Anforderung von der Erde aus ein Bild vollständig übertragen wurde?

Übung 2.5

Entwerfen Sie ein Netz für 10 Knoten in Form von zwei verbundenen Sternen.

Übung 2.6

Sie erstellen mit einem Textverarbeitungsprogramm eine Hotelreservierung, die Sie dann ausdrucken und per Fax an ein Hotel in Miami schicken.

1. Verfolgen Sie den Übertragungsweg. Welche Umwandlungen werden vorgenommen, wie sind die Schnittstellen?
2. Vergleichen Sie den Ablauf mit der Alternative, den Text als E-Mail an das Hotel zu schicken. Wo liegen Vor- und Nachteile der beiden Möglichkeiten?

16.2 Übungen Kapitel 3

Übung 3.1

Die Zeichenkette Friedberg soll übertragen werden. Zur Sicherung gegen Übertragungsfehler werden Paritätsinformationen eingebaut. Ergänzen Sie in der Tabelle die Bits für Querparität sowie das zusätzliche Byte für die Längsparität. Verwenden Sie in beiden Fällen gerade Parität.

Querparität	Binär-Darstellung	Zeichen
	100 0110	F
	111 0010	r
	110 1001	i
	110 0101	e
	110 0100	d
	110 0010	b
	110 0101	e
	111 0010	r
	110 0111	g
		Längsparität

Übung 3.2

Wie groß ist der Hamming-Abstand bei zweidimensionaler Parität?

Übung 3.3

Berechnen Sie für das Divisor-Polynom x^3+x^2+1 die Prüfzeichen zu der Nachricht 101011011.

Übung 3.4

Die Prüfziffer der Internationalen Standard Buchnummer ISBN wird nach folgendem Algorithmus berechnet: Zunächst wird aus den 9 Ziffern der eigentlichen Kennung eine gewichtete Summe bestimmt. Dazu wird die erste Ziffer der ISBN mit 10 multipliziert, die zweite mit 9, die dritte mit 8 und so weiter. Die Produkte werden addiert. Für die Summe wird der Rest bei Division durch 11 bestimmt (Modulus 11). Dieser Rest wird als Prüfziffer angehängt. Der Sonderfall Rest 10 wird durch ein X als Prüfziffer markiert. Implementieren Sie diesen Prüfalgorithmus beispielsweise als C-Programm oder in Excel. Testen die Korrektheit an Hand einiger Beispiele.

Übung 3.5

Das nachfolgende C-Programm simuliert einen Kanal mit Bit-Fehlern. Dazu werden aus einer Textdatei test.txt nacheinander alle Zeilen gelesen. Aus jeder Zeile wird ein Rahmen erstellt. In der Grundversion wird dabei lediglich die Zeichenkette kopiert. Der Rahmen wird dann an die Kanalsimulation übergeben. Dort werden nach einer vorgegebenen Bitfehler-Wahrscheinlichkeit Übertragungsfehler in den Rahmen eingefügt. Aus dem Rahmen wird dann wieder der Text extrahiert und in eine Datei mit dem Namen uebertragen.txt geschrieben.

- Untersuchen Sie mit der Grundversion den Einfluss der Bitfehler auf die Rahmenfehler, das heißt die Anzahl der fehlerhaften Rahmen (Zeilen). Erstellen Sie eine grafische Darstellung der Abhängigkeit zwischen der Häufigkeit von Bitfehlern und Rahmenfehlern.
- Fügen Sie in den vorgesehenen Funktionen eine Paritätsprüfung ein. Wie wirken sich Quer- und Längsparität aus? Wie oft müssen Rahmen wiederholt werden?
- Integrieren Sie eine CRC-Prüfung (Hinweis: eine Implementierung ist auf der Seite www.w3.org/TR/PNG-CRCApPendix.html des World Wide Web Consortiums angegeben).

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <string.h>
#include <math.h>
#define MAX_ZEICHEN 200
```

```

/* Kanalsimulation
 * Verarbeite len Daten aus buffer
 * Mit der Wahrscheinlichkeit prob wird ein Bit verfälscht.
 * Rückgabewert: Anzahl der veränderten Bits */
int kanal( char *buffer, int len, double prob ){
    int i, j, limit;
    unsigned int pos;
    int count = 0;

    /* Um auch kleine Wahrscheinlichkeiten behandeln zu können,
     * werden zwei Zufallszahlen erzeugt. Nur wenn beide kleiner
     * als Wurzel(prob) sind wird das jeweilige Bit gedreht. */
    limit = (int) (sqrt(prob) * RAND_MAX);
    for( i=0; i<len; i++ ) { /* über alle Zeichen */
        pos = 1; /* Bit-Maske */
        for( j=0; j<8; j++ ) { /* über alle 8 bits */
            if( rand()< limit && rand() < limit) {
                /* Bit mittels XOR umdrehen */
                buffer[i] ^= pos;
                ++count;
            }
            pos <<= 1; /* Maske für nächstes Bit */
        }
    }
    return count;}

void bildeRahmen( char *rahmen, char *zeile ) {
    /* Hier wird ein Rahmen gebildet */
    strcpy( rahmen, zeile );}

void leseRahmen( char *erhalten, char *rahmen ) {
    /* Hier wird der Text aus dem Rahmen geholt */
    strcpy( erhalten, rahmen );}

int rahmenFalsch( char *rahmen ) {
    /* Bis jetzt sind keine Rahmen falsch */
    return 0;}

void main(int argc, char* argv[]){
    double prob = 0.025; /* Bitfehler-Wahrscheinlichkeit */
    int len;
    int rahmenFehler = 0;
    int c_char = 0, c_fehler = 0;
    char zeile[MAX_ZEICHEN],
          rahmen[MAX_ZEICHEN],
          erhalten[MAX_ZEICHEN];
    FILE *fp_in, *fp_out;

    /* Initialisierung des Zufallszahlengenerators */
    srand( (unsigned)time( NULL ) );

```

```

fp_in  = fopen( "test.txt", "r" );
fp_out = fopen( "uebertragen.txt", "w" );
while( fgets( zeile, sizeof zeile, fp_in ) ) {
    len = strlen( zeile );
    do {
        bildeRahmen( rahmen, zeile );
        c_fehler += kanal( rahmen, len, prob );
        c_char  += len * 8;
    } while( rahmenFalsch( rahmen ) );

    leseRahmen( erhalten, rahmen);
    if( strcmp( erhalten, zeile ) ) ++rahmenFehler;
    fprintf( fp_out, "%s", rahmen );    }
printf( "Anzahl von Bitfehlern:    %d, relativ %f%%\n",
➤ c_fehler, 100.*(double) c_fehler / c_char );
printf( "Anzahl von Rahmenfehlern: %d\n",
➤ rahmenFehler );}

```

Übung 3.6

Sie wollen eine Datei von 3 MByte von Frankfurt nach New York (8000 km inklusive Umwegen) übertragen. Berechnen Sie die Dauer vom Beginn der Übertragung (Senden des ersten Zeichens) bis zum Ende (Empfang des letzten Zeichens) für eine Leitung mit 1 Mbit/s. Auf der Strecke befindet sich in der Mitte (bei 4000 km) ein Knoten, der die korrekte Übertragung prüft. Dazu nimmt er Pakete von jeweils 1024 Bytes an. Dann benötigt er 1 ms zur Paritätsprüfung jedes Pakets. Anschließend verschickt er das Paket. Sobald er das letzte Zeichen verschickt hat, ist er zum Empfang des nächsten Pakets bereit. Dann schickt er eine Bestätigung (8 Bytes) an den Sender, damit dieser das nächste Paket absenden kann. Welche Zeit benötigt nun die Übertragung?

Hinweis: Zeichnen Sie ein Zeitstrahldiagramm mit Sender, Knoten und Empfänger. Verwenden sie die Lichtgeschwindigkeit $c=300000 \text{ km/s}$ als Ausbreitungsgeschwindigkeit. Rechnen Sie bei der Dateigröße mit $1 \text{ MByte} = 1024 \cdot 1024 \text{ Byte}$.

16.3 Übungen Kapitel 4

Übung 4.1

Planen Sie das Netzwerk für Ihre Firma. Die Firma befindet sich in einem dreistöckigen Gebäude. In jedem Stockwerk sind die Räume wie folgt verteilt:

Jeder Raum ist 5 m breit und 10 m lang. Die Labore sind 10 m breit und 23 m lang, und die Geschosshöhe beträgt 4 m. Die normalen Räume benötigen je 4 Netzanschlüsse, die Labore je 10. Die Server werden in Raum 5, 1. Stockwerk untergebracht. Von dort sollen Kabel zu allen Anschlüssen im gleichen Stockwerk gelegt werden. Über den Kabelschacht können Verteiler in den beiden anderen Stockwerken erreicht werden.

Labor	1	2	3	4	5	Schacht
	6	7	8	9	10	

- Skizzieren Sie die notwendige Verkabelung.
- Informieren Sie sich im Internet über die entsprechenden Kosten der benötigten Teile (zum Beispiel www.transtec.de). Welche Gesamtkosten ergeben sich inklusive eines großen Servers?
- Wie groß sind die Preisunterschiede bei verschiedenen Alternativen (unterschiedliche Übertragungskapazitäten je nach Anforderung wie etwa Büroräume oder CAD-Arbeitsplätze)?
- Wie sieht eine Alternative mit WLAN aus?

16.4 Übungen Kapitel 5

Übung 5.1

Entwerfen Sie ein Netz für zehn Knoten in Form von zwei verbundenen Sternen. Wie sieht die Weiterleitungstabelle in den beiden Knoten im Zentrum aus?

Übung 5.2

Vergleichen Sie die minimale Verzögerungszeit für einen Ethernet-Rahmen mit 1000 Bytes Nutzdaten in einem Switch in Abhängigkeit davon, ob das Store-and-Forward- oder das Cut-Through-Prinzip verwendet wird.

16.5 Übungen Kapitel 6

Übung 6.1

Füllen Sie für Ihren Rechner die Felder im folgenden Formblatt aus. Benutzen Sie die Programme ipconfig, nslookup, ping, arp und tracert mit entsprechenden Optionen, um die Informationen zu erhalten.

Hostname	
IP-Adresse	
Subnet-Mask	
Adresse ihres Netzwerks	
Netzwerk-Klasse (A, B oder C?)	
Ethernet-Adresse	
DNS-Server	
Anzahl der Hops bis zu www.ibm.com	

16.6 Übungen Kapitel 7

Übung 7.1

Betrachten Sie eine TCP-Verbindung mit 100Mbit/s. Wie lange dauert es bei maximaler Ausnutzung der Bandbreite, bis sich die Segmentnummer wiederholt?

Übung 7.2

Die Werte für die Abbildung zur Überlastkontrolle in Kapitel 7 wurden mit folgendem C-Programm erzeugt.

```
#include "stdio.h"
#include "stdlib.h"
int main(int argc, char* argv[]){
    int t;
    double fenster    = 1;
    double kapazitaet = 20;

    for( t=0; t<60; t++ ) {
        printf("%d %.2f %.2f\n", t, fenster, kapazitaet);
        if( fenster < kapazitaet ) ++fenster;
        else fenster /= 2.;
        /* Kapazitaet um einen Wert aus [-0.6,0.6] ändern */
        kapazitaet += 1.2 * (rand()/RAND_MAX - 0.5) ;}
    return 0;
}
```

Ausgehend von diesem Programm:

- Testen Sie den Einfluss der verschiedenen Parameter auf den Verlauf.
- Integrieren Sie einen Zähler für die Häufigkeit der Kapazitätsüberschreitung.
- Wie lässt sich das Modell für die aktuell zur Verfügung stehende Kapazität verfeinern? Wie könnte beispielsweise ein Verhalten mit zwei Zuständen – frei und stark belastet – modelliert werden?

16.7 Übungen Kapitel 8

Übung 8.1

Wie viele Zellen pro Sekunde benötigt man zur Übertragung von Sprachsignalen in ISDN-Qualität über ATM? Wie ist dabei das Verhältnis zwischen Nutzdaten und Header-Daten (die Datenfelder sind komplett mit Sprachdaten gefüllt)?

16.8 Übungen Kapitel 9

Übung 9.1

Entwickeln Sie eine Anwendung Telefonbuch. Die Anwendung besteht aus einem Server und einem Client. Der Server verwaltet das Telefonbuch und kann beispielsweise die Einträge aus einer Datei lesen. Über eine Socket-Verbindung kann der Client Anfragen an den Server schicken. Eine solche Anfrage enthält einen Namen. Findet der Server zu diesem Namen einen Eintrag, so sendet er als Antwort die Telefonnummer. Andernfalls sendet er eine Fehlermeldung.

- Implementieren Sie den Server mit Schwerpunkt auf der Kommunikation.
- Zum Testen des Servers können Sie die Anwendung telnet verwenden.
- Implementieren Sie den Client.
- Erweitern Sie die Programme um einige Funktionalitäten, so dass etwa über den Client auch neue Einträge eingegeben werden können.

16.9 Übungen Kapitel 10

Übung 10.1

Erweitern Sie die Telefonbuchanwendung um:

- eine Prozedur zur Eingabe neuer Einträge
- eine Prozedur zur Suche aller Einträge mit einem gegebenen Namen

16.10 Übungen Kapitel 11

Übung 11.1

Verwenden Sie telnet, um mit SMTP-Befehlen eine E-Mail zu schreiben und zu versenden. Testen Sie, welche Adressen als Absender akzeptiert werden.

Übung 11.2

Über welchen URI kann man den selbst entwickelten Server ansprechen? Wie lässt sich der Server zu einem Besucherzähler ausbauen?

Übung 11.3

Geben Sie den Verweis auf eine Newsgroup in Ihren Browser ein. Wie zeigt er die entsprechende Gruppe an? Unter Windows können Sie den Verweis auch direkt auf dem Desktop anlegen. Was passiert dann bei Aktivieren?

16.11 Übungen Kapitel 12

Übung 12.1

Ergänzen Sie die Socket-Anwendung zum E-Mail-Versand um eine Kontrolle der zurückgegebenen Statusmeldungen.

Übung 12.2

Erweitern Sie das Beispiel eines Chat-Servers:

- Ein Client wird automatisch mit der ersten Nachricht angemeldet.
- Wenn ein Client nach einer vorgegebenen Anzahl von Nachrichten selbst keine geschickt hat, wird er aus der Liste der aktiven Clients gelöscht. Ergänzen Sie das Protokoll, so dass Clients ansonsten leere Nachrichten als „Lebenszeichen“ schicken können.
- Was passiert, wenn ein eingegebener Text länger als die Paketlänge 256 ist? Erweitern Sie die Programme, um auch solche Texte behandeln zu können.
- Wenn sich der Client hinter einer Firewall befindet, kann der Sendeport von Paket zu Paket variieren. Verändern Sie das Protokoll dahingehend, dass ein Client bei der Anmeldung eine eindeutige Kennung erhält. Mit dieser Meldung markiert er dann alle Nachrichten, so dass sie vom Client wieder eindeutig zugeordnet werden können.

16.12 Ausgewählte Lösungen

Lösung 1.2

In dem folgenden C-Programm wird eine Liste von Rechnern getestet. Dazu wird jeweils über die Funktion `system` der Befehl `ping` aufgerufen. Die Option `-n 1` besagt, dass nur jeweils ein Paket geschickt wird. Die Ausgabe des Aufrufs wird in eine Datei mit dem Namen `tmp.log` umgeleitet. Von dort wird sie wieder von dem Testprogramm gelesen. Dann wird untersucht, ob die siebte Zeile die Zeichenfolge „(0% Verlust“ enthält.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
void main(int argc, char* argv[]){
    FILE *fp;
```

```
char *befehl="ping -n 1 ";
char *rechner[] = {
    "www.t-online.de",
    "www.gibtsnicht.nix",
    "128.30.52.25",
    "www.vw.com"    };
char *nach = " > tmp.log";
char aufruf[200];
char zeile[200];
int i;
for(i=0;i<(sizeof rechner)/(sizeof (char*)); i++ ) {
    printf("%-20s", rechner[i] );
    sprintf(aufruf, "%s%s%s", befehl,rechner[i],nach);
    system( aufruf );
    fp = fopen( "tmp.log", "r" );
    for( int i=0; i<7; i++ ) {
        fgets( zeile, sizeof zeile, fp ); }
    if( strstr( zeile, "(0% Verlust" ) ) {
        printf("\tOkay\n");
    } else { printf("\tFehler\n" ); }
    fclose( fp ); } }
```

Als Ausgabe resultiert

www.t-online.de	Okay
www.gibtsnicht.nix	Fehler
128.30.52.25	Okay
www.vw.com	Okay

Lösung 2.2

Sprache: $t = 70000 / 300000 = 0,233$ s; Bild: $t = 5000 / 300000 = 0,0167$ s

Die Laufzeitdifferenz beträgt 0,216 Sekunden.

Lösung 2.31

$$l_{\max} = c \cdot t = 300000 \text{ km/s} \cdot 10\text{ms} = 3000 \text{ km}$$

$$t_u = 1024 \cdot 8 / 1 \text{ Mbit/s} = 8,192 \text{ ms}$$

Damit bleibt für die Ausbreitungsverzögerung $t_a = 10 - 8,192 = 1,808$ ms.

Schließlich ergibt sich $l_{\max} = 300000 \text{ km/s} \cdot 1,808 \text{ ms} = 542,4 \text{ km}$

Lösung 2.4

1. Ausbreitungsverzögerung: $t_a = 1,283$ s; Minimale Antwortzeit: $2 \cdot t_a = 2,566$ s
 2. Ausbreitungsverzögerung: $t_a = 54 \cdot 10^6 \text{ km} / 300000 \text{ km/s} = 180$ s; Minimale Antwortzeit: $2 \cdot t_a = 360$ s
 3. Verzögerungs-Bandbreiten-Produkt: $1,283 \text{ s} \cdot 1 \text{ Mbit/s} = 157 \text{ kByte}$
 4. $t_u = 2 \text{ MByte} / 1 \text{ Mbit/s} = 16,78$ s; Mindestdauer: $2 \cdot t_a + t_u = 19,35$ s
-

Lösung 3.1

Querparität	Binär-Darstellung	Zeichen
1	100 0110	F
0	111 0010	r
0	110 1001	i
0	110 0101	e
1	110 0100	d
1	110 0010	b
0	110 0101	e
0	111 0010	r
1	110 0111	g
0	100 1110	Längsparität

Lösung 3.2

Zwei gültige Zeichen unterscheiden sich in mindestens vier Positionen. Demnach beträgt der Hamming-Abstand 4.

Lösung 3.3

Als Rest erhält man die Bitfolge 110.

Lösung 3.5

Die folgende Funktion zeigt eine mögliche Implementierung der Paritätsprüfung für ein Zeichen.

```
// Funktion parity
// Zählt die Anzahl der Bits mit Wert 1 in einem Zeichen
// Rückgabewert: 1 bei ungerader, 0 bei gerader Anzahl
int parity( char zeichen ) {
    int j;
    unsigned int pos;
    int count = 0;

    pos = 1;
    for( j=0; j<8; j++ ) { /* alle bits */
        if( zeichen & pos ) ++count;
        pos <<= 1;
    }
    return count % 2;}

```

Abhängig vom Resultat der Prüfung muss das Paritätsbit (höchstwertige Bit) gesetzt werden. Eine entsprechende Anweisung ist:

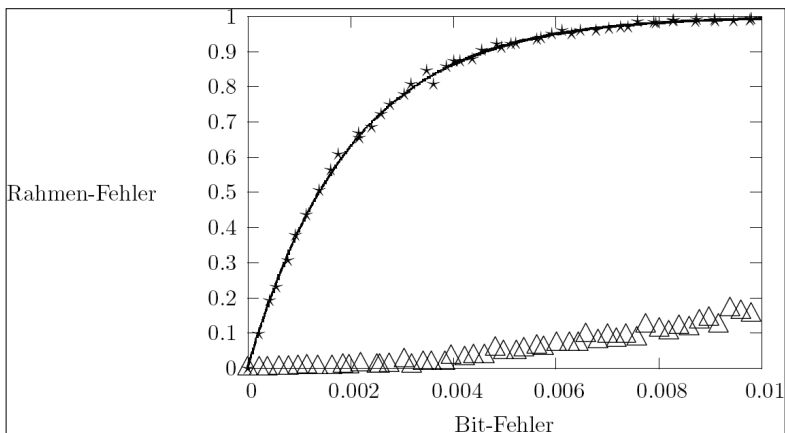
```
rahmen[i] |= 0x80; // Setzen des Paritätsbits
```

Schließlich muss der Empfänger die Paritätsbits wieder löschen, um die ursprünglichen Zeichen zu erhalten. Der Programmcode dazu ist:

```
rahmen[i] &= 0x7f; // Löschen des Paritätsbits
```

Aus der gegebenen Bitfehler-Wahrscheinlichkeit kann die Wahrscheinlichkeit für fehlerhafte Rahmen berechnet werden. Am einfachsten betrachtet man dazu die Wahrscheinlichkeit für fehlerfreie Rahmen. Damit ein Rahmen fehlerfrei ist, muss jedes einzelne Bit korrekt sein. Mit der Bezeichnung P_{bit} für die Wahrscheinlichkeit eines korrekten Bits ist die Wahrscheinlichkeit, dass N Bit korrekt sind, die Potenz $(P_{\text{bit}})^N$. Umgeschrieben auf die Fehlerwahrscheinlichkeiten ergibt sich dann $P_{\text{RF}} = 1 - (1 - P_{\text{BF}})^N$, wobei P_{BF} die Bitfehler- und P_{RF} die Rahmenfehler-Wahrscheinlichkeit bezeichnen.

Das Bild zeigt die Simulationsergebnisse für Rahmen mit 62 Zeichen. Zusätzlich ist der Verlauf der Funktion $P_{\text{RF}} = 1 - (1 - P_{\text{BF}})^{8 \cdot 62}$ eingetragen.



Lösung 3.6

Ablauf für einen Block:

Von Frankfurt bis zum Knoten (tU+tA)	0,02153 s
Paritätsprüfung	0,00100 s
Versand (tU)	0,00819 s
Bestätigung (tU+tA)	0,01334 s
Summe pro Block	0,04406 s

Insgesamt resultiert daraus eine Übertragungsdauer von 135,35 Sekunden.

Lösung 5.2

In einem Store-and-Forward-Switch wird ein Rahmen erst nach erfolgter Prüfung weitergeleitet. Damit muss der ganze Rahmen (1000 Bytes Nutzdaten plus 29 Byte Header) abgewartet werden. Bei beispielsweise 100 Mbit/s dauert dies

$$\frac{1092 \cdot 8 \text{ bit}}{100 \text{ Mbit/s}} = 823,2 \mu\text{s}$$

Demgegenüber benötigt ein Cut-Through-Switch nur die Zieladresse. Er beginnt daher bereits nach 14 Byte mit der Weiterleitung und benötigt dafür

$$\frac{14 \cdot 8 \text{ bit}}{100 \text{ Mbit/s}} = 11,2 \mu\text{s}$$

Lösung 7.1

Die Segmentnummer ist 32 bit groß. Es können damit 2^{32} Bytes geschickt werden, bis sich die Nummer wiederholt. Dann gilt für die Übertragungsverzögerung

$$t_u = 2^{32} \cdot 8 / 100 / 10^6 = 346,6 \text{ sec} = 5,73 \text{ min}$$

Lösung 8.1

Die Datenrate beträgt 64 kbit/s. Bei 48 Byte Nutzdaten pro Zelle folgt damit

$$\frac{64000 \text{ bit/s}}{48 \cdot 8 \text{ bit}} = 167 \text{ Zellen/s}$$

Das Verhältnis zwischen Nutz- und Header-Daten beträgt bei ATM stets 48 : 5.

Lösung 9.2

Mit der Funktion getsockopt können diverse Optionen abgefragt werden. Die maximale Länge erhält man über den Namen SO_MAX_MSG_SIZE.

Prof. Dr. Stephan Euler

tecCHANNEL-Links zum Thema	Webcode	Compact
Übungen und ausgewählte Lösungen	a1505	S.172
Listings zum Download	a1541	–

Mit den Webcodes gelangen Sie auf www.tecChannel.de direkt zum gewünschten Artikel. Geben Sie dazu den Code in das Feld „Webcode“ in der Titelleiste von www.tecChannel.de ein. Mit „p“ gekennzeichnete Webcodes beziehen sich auf Online-Artikel aus tecCHANNEL-Compact. Es ist möglich, dass Artikel zunächst exklusiv in tecCHANNEL-Compact veröffentlicht werden und daher erst später über den Webcode erreichbar sind.

Akronyme

3GPP	3rd Generation Partnership Project
AAL	ATM Adaptation Layer
ACF	Application Configuration File
ACK	Acknowledge. ASCII-Zeichen 6
ADPCM	Adaptive Differentielle Pulscodemodulation
ADSL	Asymmetric Digital Subscriber Line
AP	Access Point
APNIC	Asia Pacific Network Information Centre
ARIN	American Registry for Internet Numbers
ARP	Address Resolution Protocol
AS	Autonomes System. In sich abgeschlossenes System mit einheitlichem Routing
ASCII	American Standard Code for Information Interchange
ATM	Asynchronous Transfer Mode
BACnet	Building Automation Control Network
BGP	Border Gateway Protocol. Protokoll für das Routing zwischen autonomen Systemen
CAN	Controller Area Network
CCITT	Comité Consultatif International Téléphonique et Télégraphique
CELP	Code Excited Linear Predictive Coding
CEPT	Conférence Européenne des Administrations des Postes et des Télécommunications
CIDR	Classless InterDomain Routing
CRC	Cyclic Redundancy Check
CTS	Clear To Send
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
DECT	Digital European Cordless Telephony. Standard für digitales schnurloses Telefon
DES	Data Encryption Standard
DHCP	Dynamic Host Configuration Protocol
DLE	Data Link Escape
DMZ	De-Militarized Zone
DNS	Domain Name System
DSL	Digital Subscriber Line
EIB	Europäischer Installationsbus
EIGRP	Enhanced Interior Gateway Routing Protocol
ETSI	European Telecommunications Standards Institute
ETX	End of TeXt. ASCII-Zeichen 3
FDDI	Fiber Distributed Data Interface
FTP	File Transport (Transfer) Protocol
GAN	Global Area Network

GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications. Standard für digitales Funktelefon
HDLC	High Level Data Link Control
HTML	HyperText Markup Language
HTTP	HyperText Transport Protocol
IAB	Internet Architecture Board
IANA	Internet Assigned Numbers Authority
ICANN	Internet Corporation for Assigned Names and Numbers
ICMP	Internet Control Message Protocol
IDEA	International Data Encryption Algorithm
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IESG	Internet Engineering Steering Group
IGRP	Interior Gateway Routing Protocol
IHL	Internet Header Length
IKE	Internet Key Exchange
ILBC	Internet Low Bit Rate Codec
IM	Instant Messenger
IMAP	Internet Message Access Protocol
IP	Internet Protocol
IPsec	IP Security Protocol
IRC	Internet Relay Chat
IRTF	Internet Research Task Force
ISDN	Integrated Services Digital Network
ISO	International Standards Organization
ISO-IEC	Technical Committee of the International Organization for Standardization and International Electrotechnical Commission
ISOC	Internet SOCIety
ISP	Internet Service Provider
ITU	International Telecommunications Union
JPEG	Joint Photographic Experts Group
L2F	Layer 2 Forwarding
L2TP	Layer 2 Tunneling Protocol
LACNIC	Latin American and Caribbean Internet Addresses Registry
LAN	Local Area Network
LCP	Link Control Protocol. Protokoll zur Verhandlung der Parameter einer Verbindung mit PPP
LDAP	Lightweight Directory Access Protocol
LLC	Logical Link Control
LON	Local Operating Network
MACA	Multiple Access with Collision Avoidance
MAN	Metropolitan Area Network
MAP	Manufacturing Automation Protocol

MIB	Management Information Base
MIME	Multi-Purpose Internet Mail Extension
MPEG	Motion Picture Experts Group
MSS	Maximum Segment Size. Maximale Größe eines TCP-Segments
NAT	Network Address Translation
NAK	Negative Acknowledge. ASCII-Zeichen 21
NIC	Network Information Center
NIST	National Institute of Standards and Technology
NNI	Network Node Interface. Netzwerk-Interface bei ATM
NNTP	Network News Transfer Protocol
NRZ	Non-Return to Zero
NRZI	Non-Return to Zero Inverted
NTP	Network Time Protocol
OSI	Open Systems Interconnection
PAN	Personal Area Network
PARC	Palo Alto Research Center, Forschungslabor der Firma XEROX
PAT	Port and Address Translation
PCM	Pulscodemodulation
PGP	Pretty Good Privacy
POP3	Post Office Protocol Version 3
PPP	Point-to-Point Protocol
PPTP	Point-to-Point Tunneling Protocol
PTI	Payload Type Identifier. Kennung für Zellentyp bei ATM
PVC	Permanent Virtual Circuit. Permanente virtuelle Leitung über Paketvermittlung
QoS	Quality of Service
RARP	Reverse Address Resolution Protocol
RFC	Request For Comment
RIP	Routing Information Protocol
RIPE	Réseaux IP Européens
RIPE	NCC RIPE Network Coordination Centre
RMI	Remote Method Invocation
ROHC	RObust Header Compression
RPC	Remote Procedure Call
RTP	Real-Time Transport Protocol
RTCP	RTP Control Protocol
RTS	Request To Send
RTT	Round-Trip Time
SAN	Storage / System Area Network
SAP	Session Announcement Protocol
SDP	Session Description Protocol
SEAL	Simple Efficient Adaptation Layer. Bezeichnung für AAL-5
SIP	Session Initiation Protocol
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol

SSL	Secure Sockets Layer
SSRC	Synchronization Source. Quelle eines RTP-Stroms
STP	Spanning Tree Protocol
STX	Start of TeXt. ASCII-Zeichen 2
SVC	Switched Virtual Circuit. Vermittelte virtuelle Leitung über Paketvermittlung
SWS	Send Window Size. Die maximale Anzahl von ausstehenden Rahmen beim Sliding Window Algorithmus
TCP	Transmission Control Protocol
TFTP	Trivial File Transport Protocol
THT	Token Hold Time. Die Zeit, für die ein Knoten beim Token-Ring das Token behalten darf.
TLS	Transport Layer Security
TOS	Type of Service. Spezifikation im IP-Header
TRT	Token Rotation Time. Die Umlaufzeit für das Token bei Token-Ring-Netzen
TTL	Time To Live. Zähler für die maximale Lebensdauer eines IP-Datagramms
TTP	Time Triggered Protocol. Echtzeitfähiger Bus zur Vernetzung im Kfz-Bereich
UBE	Unsolicited Bulk E-Mail. Unverlangte Massen-E-Mails
UCE	Unsolicited Commercial E-Mail. Unverlangte kommerzielle E-Mails
UDP	User Datagram Protocol
UNI	User Node Interface. Endpunkt-Interface bei ATM
URI	Universal Resource Identifier
URL	Universal Resource Locator
URN	Universal Resource Name
USB	Universal Serial Bus
UUID	Universally Unique Identifier
VC	Virtual Circuit. Virtuelle Leitung über Paketvermittlung
VCI	Virtual Circuit Identifier
VCI	Virtual Channel Identifier. Kennung für virtuellen ATM-Kanal
VPI	Virtual Path Identifier. Kennung für virtuellen ATM-Pfad
VPN	Virtual Private Network
RRRP	Virtual Router Redundancy Protocol
W3C	World Wide Web Consortium
WAN	Wide Area Network
WEP	Wireless Equivalent Privacy. Ursprünglicher Sicherheitsmechanismus für WLAN
Wi-Fi	Wireless Fidelity
WMAN	Wireless Metropolitan Area Networks
WPA	Wireless Protected Access. Verbesserter Sicherheitsmechanismus für WLAN
WPAN	Wireless Personal Area Network

Literaturverzeichnis

- [1] Larry L. Peterson and Bruce S. Davie. Computernetze. dpunkt.verlag, Heidelberg, 1. edition, 2000
- [2] Douglas E. Comer. Computernetzwerke und Internets. Prentice Hall
- [3] C. Lindemann, C. Immler, and F. Harms. Internet Intern. Data Becker, 2000
- [4] Erich Stein. Taschenbuch Rechnernetze und Internet. Fachbuchverlag Leipzig im Carl Hanser Verlag, 2001
- [5] Andrew S. Tanenbaum. Computernetzwerke. Prentice Hall, 2000
- [6] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. Numerical recipes in C. Cambridge University Press, 1992
- [7] M. Schwartz. Telecommunication Networks: Protocols, Modeling and Analysis. Addison-Wesley, 1988
- [8] Echelon. Introduction to the LONWORKS System 1.0, 1999
- [9] Y. Yeh, M. G. Hluchyj, and A. S. Acampora. The knockout switch: A simple, modular architecture for high-performance packet switching. IEEE J. Sel. Areas Commun., 5(8):1274 -1283, October 1987
- [10] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jonathan B. Postel, Lawrence G. Roberts, and Stephen S. Wolff. The past and future history of the internet. Communications of the ACM, 40(2):102-108, 1997
- [11] Daniel Karrenberg, Gerard Ross, Paul Wilson, and Leslie Nobile. Development of the regional internet registry system. The Internet Protocol Journal, www.cisco.com/warp/public/759/ipj_4-4/ipj_4-4_regional.html, 2001
- [12] H. Häckelmann, H.J. Petzold, and S. Strahinger. Kommunikationssysteme: Technik und Anwendung. Springer, Berlin, 2000
- [13] Emmanuel Chimi. High-Speed Networking: Konzepte, Technologien, Standards. Hanser, München Wien, 1998
- [14] Sonia Fahmy, Raj Jain, Shivkumar Kalyanaraman, Rohit Goyal, Bobby Vandalore, and Xiangrong Cai. A survey of protocols and open issues in ATM multipoint communication. OSU Technical Report, Ohio State University, August 21 1997
- [15] W. R. Stevens. UNIX network programming. Prentice Hall, Englewood Cliffs, 1990
- [16] L. Wall and R.L. Schwartz. Programmieren mit Perl. O'Reilly Verlag, 2001

-
- [17] Microsoft. RPC Programmer's Guide and Reference
 - [18] Rolf-Dieter Köhler. Voice over IP. mitp-Verlag, Bonn, 2002
 - [19] B. Goode. Voice over Internet Protocol (VoIP). Proc. IEEE, Vol. 90:1495-1517, 2002
 - [20] P. Noll. Speech and audio coding for multimedia communications. In Proceedings International Cost 254 Workshop on Intelligent Communication Technologies and Applications, pages 253-263, Neuchatel, 2000
 - [21] Stephan Rein, Frank Fitzek, and Martin Reisslein. Voice quality evaluation for wireless transmission with ROHC. In Proceedings of the 7th IASTED International Conference on Internet and Multimedia Systems and Applications (IMSA), pages 461-466, Honolulu, August 2003
 - [22] S. Singh. Geheime Botschaften. Carl Hanser Verlag, München, 2000
 - [23] National Institute of Standards and Technology (NIST). FIPS Publication 81: DES Modes of Operation, December 2, 1980
 - [24] X. Lai and J. Massey. A proposal for a new block encryption standard. In I. B. Damgard, editor, Proc. EUROCRYPT 90, pages 389-404. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 473
 - [25] W. Diffie and M. E. Hellman. New directions in cryptography. IEEE Trans. Inform. Theory, IT-22:644-654, November 1976
 - [26] W. Diffie. The first ten years of public-key cryptography. Proc. IEEE, 76(5):560-577, May 1988
 - [27] Philip Zimmermann, Christopher Creutzig, and Andreas Buhl. PGP - Pretty Good Privacy - Der Briefumschlag für Ihre elektronische Post. Art d'Ameublement, 1999
 - [28] Dieter Weißhaar, editor. IT-Kompendium - IT-Know-how von A-Z. Eigenverlag transtec, 2003
 - [29] Tom Sheldon. The Encyclopedia of Networking and Telecommunications. Osborne/McGraw-Hill, 2001

Glossar

8B/10B

Kodiervorgang zum Umwandeln von 8-Bit-Datenpaketen zur seriellen Übertragung in 10-Bit-Datenfolgen. Die Umkodierung muss sicherstellen, dass der 10-Bit-Daten-String genügend viele Taktinformationen enthält, mit denen sich Sender und Empfänger gegenseitig synchronisieren. Das erfordert mindestens 4 Pegelwechsel pro Paket.

API

Application Programming Interface. Spezifizierte Schnittstelle zu Funktionen in einer Anwendung oder einem OS. Damit lassen sich häufig benutzte Funktionen, wie beispielsweise das Formatieren von Texten oder Darstellen von Fenstern, in einer zentralen DLL unterbringen.

ARP

Address Resolution Protocol. TCP/IP-Protokoll, über das die physikalische Netzwerkadresse eines Clients ermittelt werden kann.

ATM

Asynchronous Transfer Mode. Netzwerktechnologie für LAN und WAN, die neben Daten auch Echtzeitsprache und Video unterstützt.

CELP

Code Excited Linear Prediction. Ein Sprachkodierungsverfahren auf LPC-

Basis, das die Zustände stimmhaft und stimmlos und dazwischen angeordnete Arten der Anregung verwendet. Algebrische CELP-Varianten (ACELP) erweitern CELP um Kompressionsmethoden zur Bandbreitenreduzierung.

CIFS

Common Internet File System. Verbesserte Version des Server Message Block (SMB) Protokolls zum Datenaustausch über Inter- oder Intranet. CIFS unterstützt kein Printer-Sharing.

CoS

Class of Service. Dienstklasse bei der Priorisierung von Datenverkehr.

DECT

Digital Enhanced Cordless Telecommunications. Digitaler, abhörsicherer Protokollstandard für die lokale Anbindung von tragbaren Sprachtelefonen. DECT ist für die Punkt-zu-Punkt-Verbindung zwischen einer Basisstation (Fixed Part – FP) und einem Mobilteil (Portable Part – PP) ausgelegt und arbeitet in einem reservierten Frequenzbereich (in Europa zwischen 1880 und 1900 MHz).

DHCP

Dynamic Host Configuration Protocol. Bei DHCP bezieht ein Arbeitsrechner seine Konfiguration des IP-Netzwerks von einem Server.

DWDM

Dense Wavelength Division Multiplexing. Bei dieser Übertragungstechnik per Lichtwellenleiter werden verschiedene Laser mit unterschiedlichen Wellenlängen eingesetzt. Ein solches Übertragungssystem kann eine Datenrate im Terabit-Bereich erzielen.

ETSI

Europäisches Standardisierungsinstitut für Telekommunikation.

IANA

Internet Assigned Numbers Authority. Zeichnet für die Administration des Domain Name System (DNS) verantwortlich. Regelt über regionale Registrare wie APNIC, ARIN oder RIPE die Vergabe von IP-Adressen und Top Level Domains (TLDs).

ICMP

Internet Control Message Protocol. TCP/IP-Protokoll zum Austausch von Fehler- und Statusmeldungen.

IEEE

Institute of Electrical and Electronic Engineers. Eine in den USA ansässige Ingenieurvereinigung zur Erstellung von Standards und Normen.

IETF

Internet Engineering Task Force. Entwickelt und verwaltet die als RFC (Request for Comments) bekannten Internet-Standards.

ISM

Industrial, Scientific, Medical. Bezeichnung für das weltweit zur freien Nutzung durch Funkanwendungen bestimmte 2,4-GHz-Frequenzband.

Kerberos

Sicherheitsstandard des MIT zur zentralen Authentifizierung von Nutzern (single-logon). Kerberos ist nicht für die Freigabe von Ressourcen zuständig, sondern es weist einem Benutzer eine ID zu, die während der Sitzung erhalten bleibt. Über diese ID können dann die Ressourcen entscheiden, welche Rechte der Benutzer erhält.

LPC

Linear Predictive Coding. Allgemeine Bezeichnung für Kodierungsverfahren zur Nachbildung von Sprachsignalen mittels mathematischer Modelle. Das bekannteste LPC-Verfahren ist CELP (Code Excited Linear Prediction).

MAC

Media Access Control. Unterste Ebene der Sicherungsschicht (Data Link Layer) im OSI-Referenzmodell. Die MAC-Ebene steuert Vorgänge, die für bestimmte LAN-Typen spezifisch sind. Darunter fallen beispielsweise die Algorithmen zur Kanalverwaltung und Frame-Aufbau oder Kollisions- und Fehlererkennung.

MAC-Adresse

Jede Ethernet- und Token-Ring-Netzwerkkarte kann über diese Seriennummer eindeutig identifiziert werden.

NetBIOS

Network Basic Input Output System: Protokoll in DOS- und Windows-Netzwerken. NetBIOS stellt eine Programmierschnittstelle für Applikationen zur Verfügung, die auf Schicht 5 des OSI-Modells arbeiten. NetBIOS kann auf dem nicht Routing-fähigen Transportprotokoll NetBEUI sowie den Routing-fähigen Protokollen TCP/IP und IPX/SPX aufsetzen.

OSI

Open Systems Interconnect. Ein ISO-Standard für weltweite Kommunikation, der ein Rahmenmodell für die Implementierung von Protokollen in sieben Schichten definiert.

OSPF

Open Shortest Path First. Routing-Protokoll. OSPF kommuniziert durch das Senden von Link-State Advertisements (LSAs) mit allen Routern derselben Hierarchiestufe. Dabei werden Informationen über die Interfaces, die verwendeten Metriken und andere Variablen ausgetauscht.

QoS

Quality of Service. Technische Merkmale eines Netzwerkdienstes, die sich durch Managementmaßnahmen garantieren lassen. Der Begriff wird oft als Umschreibung für garantierte Bandbreiten verwendet.

RARP

Reverse Address Resolution Protocol (RFC903). Client-Protokoll zur Auflö-

sung von MAC- in IP-Adressen über eine Anfrage bei einem RARP-Server.

RC4

Ein Verschlüsselungsverfahren von RSA Data Security. Es arbeitet mit einem geheimen Schlüssel und variabler Schlüssellänge.

RIP

Routing Information Protocol (RFC 2453). Zählt zu den Interior Gateway Protocols für den Austausch von Routing-Informationen zwischen Gateways in autonomen Netzen.

RMON

Steht für Remote Network Monitoring (RFC1757, RFC2021). Eine erweiterte SNMP MIB, die das Sammeln und Abfragen von Ethernet-Leistungsdaten aus der Ferne erlaubt. RMON gliedert die Daten in funktionsorientierte Gruppen wie Statistics, History, Host, Host-TopN et cetera.

RPC

Remote Procedure Call. Programmierschnittstelle zum Ausführen von Funktionen auf entfernten Rechnern. Sun hat RPC für die „Open Network Computing“-Architektur entwickelt. Eine weitere RPC-Architektur ist DCOM.

RSVP

Resource Reservation Protocol. End-to-End-Signalisierungsverfahren, über das sich die Quality of Service (QoS) für einen Datenstrom festlegen lässt.

RTP

Real Time Transport Protocol. Dient der Übertragung von Echtzeitverkehr wie Sprache oder Video über IP-Netze. RTP setzt dazu weit gehend auf das verbindungslose UDP-Protokoll auf.

SAN

Ein SAN (Storage Area Network) erfasst Unternehmensdaten zentral, speichert sie und liefert sie auf Anforderung wieder aus. Es wird parallel zu einem vorhandenen LAN/WAN aufgebaut und nur für den Datentransfer genutzt.

SMB

Server Message Block. Netzwerkprotokoll auf Schicht 6 und 7 des OSI-Modells. Es verfügt über Mechanismen zur Freigabe von Dateien, Druckern sowie Kommunikationsschnittstellen wie seriellen Ports. Es definiert auch abstrakte Kommunikation über named pipes und mail slots. Zur Datenübertragung ist ein Transportprotokoll wie TCP/IP oder NetBEUI notwendig.

SMP

Symmetrisches Multiprocessing. System, bei dem mehr als ein Prozessor zur Ausführung von Programmen verwendet wird. Eine Komponente des Betriebssystems (Scheduler) ist für die Verteilung der Aufgaben auf die einzelnen Prozessoren zuständig.

SNMP

Simple Network Management Protocol. Netzwerkprotokoll zur Verwaltung von Netzwerkgeräten.

SOI

Silicon on Insulator: SOI-Technologien nutzen eine vergrabene Oxidschicht für die vollständige dielektrische Isolation jedes Bauelements in einer integrierten Schaltung. So verringern sich die Leckströme, und die parasitären Kapazitäten sinken deutlich.

SSP

Service Switching Point. Knoten zur diensteneutralen Verbindungssteuerung in Intelligenzen Netzen (IN).

STP

Signal Transfer Point. Routing-Element in Intelligenzen Netzen.

Switch

Netzwerkvermittler. Schließt wie ein Hub mehrere Endgeräte zu einem lokalen Netz zusammen. Dabei schaltet ein Switch dezidierte Verbindungen zwischen den Kommunikationspartnern. Im Gegensatz zum Hub steht jedem Endgerät die volle Netzbandbreite zur Verfügung.

TCP/IP

Transport Control Protocol/Internet Protocol. Das meistverbreitete Netzwerkprotokoll zur Übermittlung von Daten. IP dient zur reinen Datenübermittlung, TCP stellt sicher, dass die Daten auch fehlerfrei ankommen.

TDP

Thermal Design Power. Maximale Leistungsaufnahme eines Prozessors

unter Volllast. Der TDP-Zustand der CPU wird bei der Entwicklung von Lüftern und Kühlmaßnahmen genutzt. Im normalen Betrieb erreichen Prozessoren den TDP-Wert kaum.

TE

Terminal Equipment. Allgemeine Bezeichnung für ein Endgerät.

TOE

TCP/IP Offload Engine. Je nach Ausführung Chip oder Steckkarte, die sich um die Umsetzung des TCP/IP Protocol Stack kümmert, ohne die CPU des Rechners zu belasten.

ToS

Type of Service. Kennzeichnung für den Dienstetyp, lässt sich zur Priorisierung von Datenverkehr nutzen.

TTL

Time to live: Angabe im Internet Protocol (IP), die bestimmt, nach wie vielen Hops ein Paket als unzustellbar verworfen wird.

UDP

Mit dem User Datagram Protocol versendet man Datagramme zwischen zwei Systemen. Es garantiert weder Zustellungen noch korrekte Reihenfolge der zugestellten Datenpakete.

VLANs

Virtual LAN. Logische Segmentierung geswitchter LANs unabhängig von der

physikalischen Netzstruktur. Wird am Switch durch Definition logisch und funktionell zusammengehöriger Ports definiert. Dadurch lässt es sich im Gegensatz zu einem physikalischen LAN jederzeit per Software modifizieren. Ein VLAN bildet wie ein physikalisches Netzsegment eine abgeschlossene Broadcast- und Kollisionsdomäne, der Verkehr zwischen VLANs muss also explizit geroutet werden.

VoIP

Voice over IP. Telefonie über IP-basierte Datenverbindungen.

WEP

WEP, Wireless Equivalent Privacy. Bezeichnung für Verschlüsselungsverfahren auf RC4-Basis, die von auf dem Standard IEEE 802.11 basierenden Funknetzen genutzt werden.

WINS

Windows Internet Naming Service: Dienst, der bei Microsofts Server-Betriebssystemen für eine Umsetzung von NetBIOS-Namen auf IP-Adressen sorgt.

WPAN

Wireless Personal Area Network. Bezeichnung für ein Funknetz kurzer Reichweite. Wird auch als Bezeichnung für die IEEE-Arbeitsgruppe 802.15 verwendet, die Standards für WPANs erarbeitet.

Index

Symbole

10Base-T 41
10Base2 41
10Base5 41
4B5B-Code 26
7-Bit-Zeichensatz 30

A

Access Point 46
ACF 102
Acknowledge Number 77
Acknowledgement 34
Ad-hoc-Netz 46
Adaptive Differenzielle PCM 125
Address Resolution Protocol (ARP) 63
ADPCM 125
Advertised Window 78
Aktives Scanning 46
Anwendungs- oder Verarbeitungsschicht
(application layer) 23
API 88
Application Configuration File 102
ARP-Cache 63
ARP-Tabelle 63
ASCII 30
ATM 54
Attachment 117
Ausbreitungsverzögerung 13
Autonomes System (AS) 66, 69

B

Backbone 47
Bandbreite 13, 42
Banyan-Netzwerk 56
Batcher-Netzwerk 56
Baudot-Code 26
Baudrate 26
Baumtopologie 18
Best-Effort-IP 73
BGP 69
bitorientiert 27
Bitübertragungsschicht
physical layer) 22
Bluetooth 47
Body 116

Border-Gateway 69
Border Gateway Protocol (BGP) 69
Bridge 50
Burst 30
Bustopologie 19, 41
ByteFlight 48

C

CAN (Controller Area Network) 48
CELP 126
Classless InterDomain Routing (CIDR)
60
Client-Sockets 129
CNAME 123
CRC-Polynome 34
CSMA/CD (Carrier Sense Multiple Access
with Collision Detection) 39, 42
CTS, Clear To Send 46
Cut-Through-Prinzip 55
Cyclic Redundancy Check CRC 32

D

Datagramm 51, 62
DatagramPacket 131
Data Encryption Standard 145
Data Link Escape 28
De-Militarized Zone 140
DECT 125
DES 145
DHCP 61
Diffie-Hellmann-Merkle 147
Digital European Cordless Telephony 125
Distanzvektor-Routing 66, 67
DLE 28
DMZ 140
DNS 70
Domain Name System (DNS) 70
Drahtlose LAN 44
Dynamic Host Configuration Protocol 61

E

E-Mail 115
Echtzeitverhalten 47
EIB 48
Einwegfunktion 147

Ende-zu-Ende Schicht 22

Entfernungstabelle 67

Ethernet 38

Ethernet-Adresse 39

ETX 28

F

Fast Ethernet 41

FDDI (Fiber Distributed Data Interface)
42, 47

Fehlererkennung 29

Feldbus 47

Fileservice 42

Firewall 139

Firewire IEEE 1394 47

FlexRay 48

Flusskontrolle 78

Forum 119

Frage-und-Antwort 100

Frame 27

Frame-Relay 54

Full-Duplex-Betrieb 41

Funkzellen 46

G

GAN 17

Gateway 50, 118

Generator-Polynome 34

Gerade Parität (even parity) 30

Gigabit Ethernet 41

H

H.323 124

Hamming-Abstand 31

Header 116

Hop 67

HTML 110

HTTPS 151

Hub 41, 49, 50

I

IAB 71

IANA 72

IBM-Token-Ring 42

ICANN 72

ICMP 64

IDEA 146

IDL 102

IEEE 38

IEEE 1394 47

IEEE 802.11 45

IEEE 802.1D 52

IEEE 802.2 39

IEEE 802.3 38

IEEE 802.4 47

IEEE 802.5 42

IETF 71, 124

IHL 62

Interdomain-Routing 66, 69

Interface Definition Language 102

Internet-Standards 71

Internet Architecture Board (IAB) 71

Internet Assigned Numbers Authority
(IANA) 59, 72

Internet Control Message Protocol
(ICMP) 64

Internet Draft 71

Internet Engineering Steering Group
(IESG) 72

Internet Engineering Task Force
(IETF) 71

Internet Header Length 62

Internet Protokoll 58

Internet Society (ISOC) 71

Intradomain-Routing 66

IP-Adresse 58

IP-Security-Protokoll 142

IPSec 142

IPv4 58

IPv4 Paket-Format 61

IPv6 60

ISO/OSI 21

ISOC 71

ITU 124

J

Java 129

Java Client-Sockets 129

Java Server-Sockets 130

Java UDP-Sockets 131

Jitter 16

K

Kapselung 21

Klassen 59

Klasseneinteilung 59

Knockout-Switch 56

Knoten 18

Kodierung 25

Kollision 45

Kollisionsdomäne (collision domain) 41
Kryptoanalyse 143
Kryptografie 143

L

L2TP 142
LAN, local area network 16
Latenz 13
Layer 2 Tunneling Protocol 142
Lease 61
Lichtleiterkabel 42
Link-State-Routing 69
Local Operating Network (LON) 48
Logical Link Control (LLC) 39
LONWORKS 48
Loop-Back 59

M

MACA, Multiple Access with Collision Avoidance 45
Mailto-Links 137
MAN 38
Manchester-Kodierung 26
Maximum Segment Size 76
Media Access Control (MAC) 39
Medienzugriff 39, 43
Mehrfachzugriffsnetz 39
MIDL 103
MIME 115, 116
Monoalphabetische Verschlüsselung 143
MSS 76
Multiport-Bridge 50

N

Name-Server 70
Names, Universal Resource 113
Network Information Center (NIC) 59
Netzüberwachung 44
Next-Hop-Router 63
Non-Return to Zero (NRZ) 25

O

Öffentliche Schlüssel 149
On-the-fly-Prinzip 55
OSI-Referenzmodell 21

P

Paketfilter 56, 140
PAN 17

Parität 30
Passives Scanning 46
PCM 125
Peer-to-Peer-Mode 46
Permanent Virtual Circuit 53
PGP 151
Ping 64
Point-to-Point Tunneling Protocol 142
Polynom-Division 34
POP3 115
Port 49
Post Office Protocol 115
PPTP 142
Profibus 47
Protokollstapel 20
Proxy, Proxy-Server 114, 140
Public Key 149
Punkt-zu-Punkt Topologie 17
Push-Operation 76
PVC 53

Q

QoS, Quality of Service 53

R

Rahmen 22, 27
Rahmengröße 28
RARP 64
Real-Time Transport Protocol 121
Regional Internet Registries (RIR) 72
Remote-Tunnel-Server 141
Remote Procedure Call 100
Repeater 41, 50
Request for Comments (RFC) 71
Reverse Address Resolution Protocol 64
RFC 71
Ringtopologie 19
RIPE Network Coordination Centre 72
RIR 72
RMI 100
ROHC 127
Round-Trip Time, RTT 13
Router 41, 50
Routing 65
Routing Information Protocol (RIP) 69
RPC 100
RSA 149
RTCP 123
RTP 121
RTT 78

S

SAN 17
SDP 124
Secure Sockets Layer 152
Segmentgröße 76
Segmentierung 75
Send Window Size, SWS 36
Server-Sockets 130
Session Initiation Protocol 124
Sicherungsschicht (data link layer) 22
Signieren 150
SIP 124
Sitzungsschicht (session layer) 23
Sliding-Window-Algorithmus 36, 77
SMTP 115, 116
SNMP 120
Socket-Verbindung 129
Sockets 88
Spanning Tree 52
Sprachkodierung 124
SSL 152
Sterntopologie 18
Sternkoppler 41
Stop-and-Wait-Algorithmus 35
Store-and-Forward-Prinzip 55
Stubs 101
Subnetzadresse 60
Subnetzmaske 60
SVC 53
Switch 41, 49, 50
Switched Virtual Circuit 53
Switching Fabric 54

T

TCP 73, 75
TCP-Header 78
TCP Segment-Format 76
Threads 119, 133
THT, Token Hold Time 44
TLS 152
Token 42
Token Bus IEEE 802.4 47
Token Ring 42
Top Level Domain 70
TOS 62
Transceiver 41
Transportschicht (transport layer) 22
Transport Layer Security 152
Tripple-DES 146
TRT, Token Rotation Time 44

TTL 62
TTP (Time Triggered Protocol) 48
Tunnel 141
Type of Service 62

U

Übertragungsverzögerung 40
UDP 73
UDP-Sockets 131
UDP Paket-Format 74
Ungerade Parität (odd parity) 30
Universally Unique Identifier 102
URI 113
URL 113
URN 113
USB 47
Usenet 119
UUID 102

V

VC 52
VCI 53
Verbindungslose Kommunikation 95
Verbindungsorientierte Kommunikation 88
Verbindungsorientierte Sockets 130
Vermittlung 49
Vermittlungsschicht (network layer) 22
Verschlüsselung 142
Verzögerung-Bandbreite-Produkt 15
Virtual Circuit 52
Virtual Private Network 54
Virtuelle private Netze 141
Virtuelle Verbindungen 52
VoIP 124
VPN 54, 142

W

WAN, wide area network 16
Wireless LAN 44
Wireless Metropolitan Area Networks (WMAN) 45
WLAN 44
WPAN 47
WWW, World Wide Web 110

Z

Zeichenzähler 28
Zone 70
Zyklische Redundanzprüfung 32