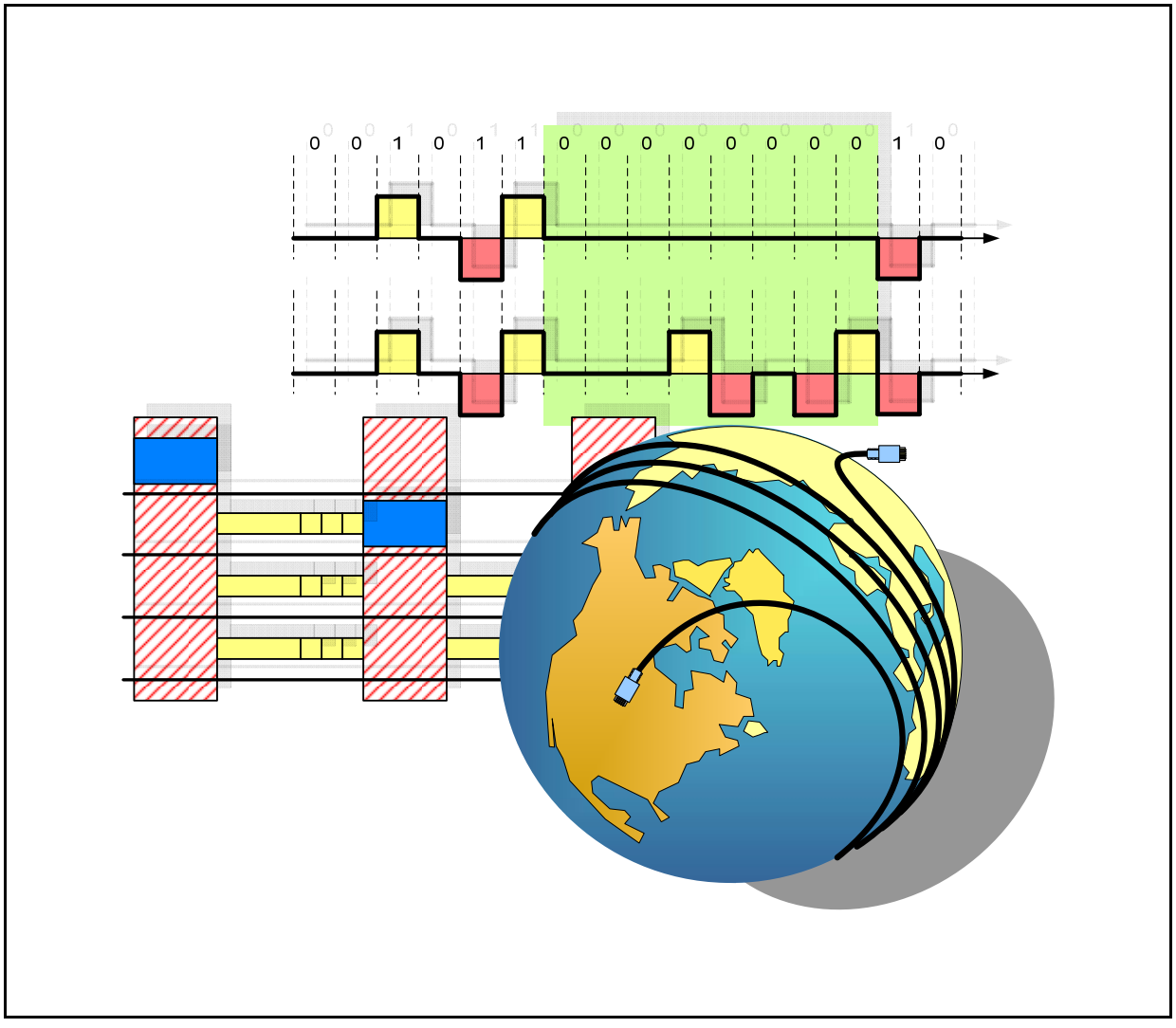
Rechnernetze

Wintersemester 2012/2013

Jörg Roth



Rechnernetze

Prof. Dr. habil. Jörg Roth
Georg-Simon-Ohm Hochschule Nürnberg
Fakultät Informatik
Keßlerplatz 12
90489 Nürnberg

Inhalt

Rechnernetze

WS 2012/13

Kapitel 1

Einleitung

Übersicht über die Vorlesung

Zwei Verlesungen befassen sich mit Rechnernetzen und Kommunikation:

I — Rechnernetze

Bitübertragung bis Transport
- Bitübertragung
- Fehlerbehandlung
- Medienzugriff
- Packet und Circuit Switching
- Routing
- Vermittlungsprotokolle des Internets
- Transportprotokolle des Internets

II — Rechnerkommunikation

Anwendungsnahe Themen
- Anwendungsentwicklung auf der Transportschicht
- Webservices
- Peer-to-Peer
- Namen
- Sicherheit
- Darstellung von Daten
- Konsistenz verteilter Daten

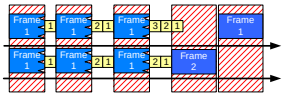
Übersicht über die Vorlesung

Vorlesung *Rechnernetze*

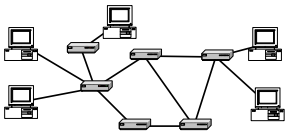
Kapitel 1: Einleitung

Kapitel 2: Protokolle und Stapel

Direktverbindungsnetzwerke



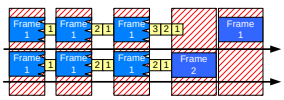
- Kapitel 3: Bitübertragung
- Kapitel 4: Sicherung der Übertragung
- Kapitel 5: Medienzugriff



Übersicht über die Vorlesung

Kommunikation zwischen nicht-direkt verbundenen Rechnern

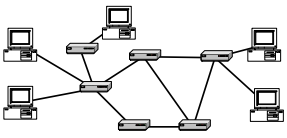
- Kapitel 6: Grundlagen
- Kapitel 7: Routing



Vermittlung im Internet

- Kapitel 8: IP
- Kapitel 9: Weitere Protokolle der Schicht 3

Kapitel 10: Transportprotokolle des Internets



Literatur

[1] Peterson, L., L.; Davie, B., S: *Computernetze*, dpunkt, 2007, ISBN 978-3-89864-491-4

[2] Tanenbaum, A., S.: *Computernetzwerke*, Pearson, 2003, ISBN 978-3-82737-046-4

[3] Tanenbaum A. S. , van Steen M.: *Verteilte Systeme*, Pearson, 2007, 978-3-82737-293-2

[4] Coulouris, G.; Dillimore, J.; Kindberg, T.: *Verteilte Systeme*, Addison Wesley, 2002, ISBN 3-8273-7022-1

[5] Roth, J.: *Mobile Computing*, dpunkt, 2005, ISBN 3-89864-366-2

[6] Roth, J.: *Prüfungstrainer Rechnernetze*, Vieweg+Teubner, 2010, ISBN 978-3-8348-0925-4

Kapitel 1: Einleitung

Ziel: Einleitung und Motivation zu dem Gebiet der Rechnernetze

- Historisches
- Motivation
- Das Internet

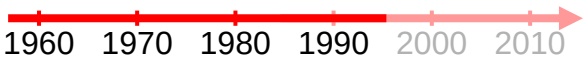
Motivation

Rechnernetze historisch:

- 60er Jahre: Netzwerke ermöglichten entfernten Zugriff auf Großrechner
  - Rechner waren teuer und groß
  - ein Rechner für viele Benutzer (z.B. Filialen)
  - Terminals waren in der Regel textbasiert
- 80er Jahre:
  - Lokale Netze
  - Benutzung von gemeinsamen Ressourcen, z.B. Drucker, Plattenkapazität
  - Beginn des Internets

Motivation

- 90er Jahre:
  - Rechner wurden individueller ("Personal" Computer)
  - Boom des Internets
  - Vernetzung zum "Vergnügen", z.B. Netzwerkspiele, LAN-Partys, Browsen im Internet
  - Computer-unterstützte Gruppenarbeit, Video-Konferenzen
- Mitte der 90er Jahre:
  - Rechner werden immer mobiler, z.B. Notebooks, PDAs
  - Boom des Mobilfunks



Motivation

3. Jahrtausend:

- Beginn der Ära der "allgegenwärtigen Computer"?
- Smart-Phone-Ära (Beginn 2007)
- Klassische PCs auf dem Rückzug

Motivation

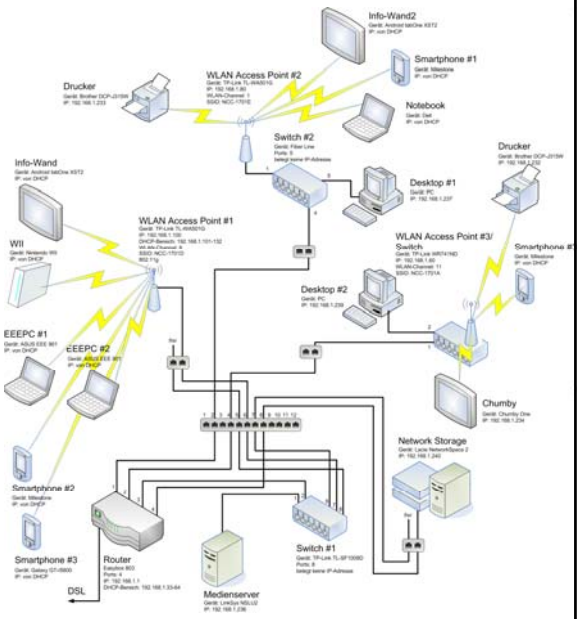
Begünstigende Faktoren für vernetzte Systeme

- Fortschritte im Bereich der Netzwerktechnologien
  - Leistungsfähige Kommunikationsverfahren
  - Leistungsfähige Vermittlungsknoten
  - Leistungsfähige Endgeräte
- Leistungsfähige Clients und Server
  - Web-Server, Dateiserver, Datenbank-server
  - Browser laufen selbst auf Handys
- Standards
  - TCP/IP als Lingua Franca der Vernetzung
  - WWW: HTTP/HTML
  - XML

Ein Mini-Dateiserver

Motivation

- Heimvernetzung und Konsummarkt

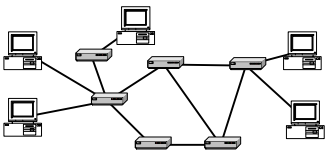


Heimnetz des Dozenten
(Stand Feb. 2012), u.a.
1 Router
2 Switches
3 WLAN Access Points
1 Network Storage
4 DHCP-Server
18 IP-Knoten

Motivation

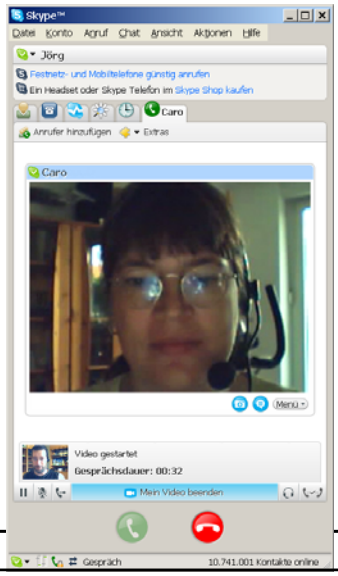
Vorteile von vernetzten, verteilten Systemen

- Leistungsfähigkeit
  - sie lässt sich (zumindest theoretisch) beliebig steigern, indem Rechner gekoppelt werden
- Verfügbarkeit
  - redundante Kopplung
  - Lastenausgleich
- Zugriff auf *gemeinsame* Daten
  - Warenhäuser, Banken etc.
  - Gruppenarbeit (Groupware)



Motivation

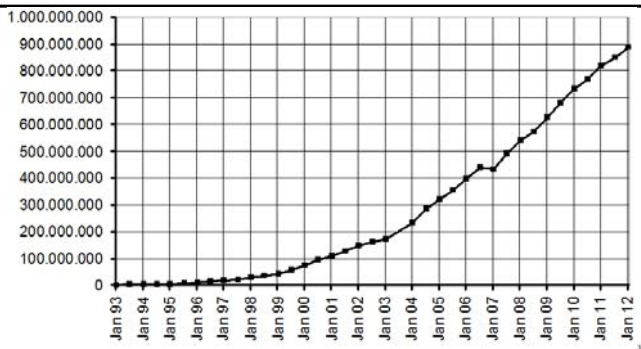
- Zugriff auf teuere Ressourcen
  - A0-Farb-Laserdrucker
  - Hochauflösende Video-Hardware
- Der Rechner als Kommunikations-medium
  - E-Mail
  - Multimedia-Kommunikation
  - IP-Telefonie, Videotelefonie
  - Mobiltelefonie: Sprache, Instant Messaging



Weltweites Internet

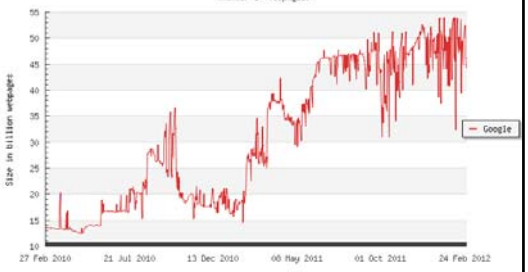
- Weltweiter Zusammenschluss aller auf TCP/IP-basierender Rechner
- Geht auf das ARPANET des US Dep. of Defense (DoD) zurück (1969); ursprüngliches Ziel: Redundanz
- 1989 ("eine der Geburtsstunden"): Übergabe der Verantwortung an die National Science Foundation (NSF); Anzahl der Rechnerknoten: 130 000
- Boom in den 90er Jahren:
  - Zugriff auf das Internet durch alle gängigen Betriebssysteme
  - Zugriff durch Modem über Internet-Service Provider von zu Hause aus
  - Einfache Bedienung durch das World Wide Web

Weltweites Internet



Internet Hosts
(http://www.isc.org/solutions/survey)

Google Index Größe
(*http://www.worldwidewebsize.com/*,
Stand Feb. 2012)



Rechnernetze

WS 2012/13

Kapitel 2

Protokolle und Stapel

Kapitel 2: Protokolle und Stapel

Ziel: Darstellung der modularen Zerlegung von Kommunikationsaufgaben

- Protokolle, Stapel
- Referenzmodelle
- Adressierung

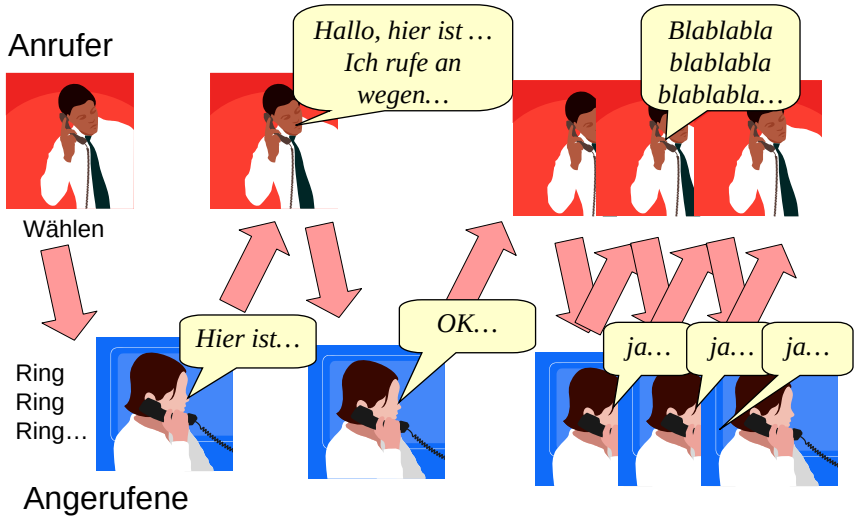
Protokolle

Was ist ein Protokoll?

- Protokolle *regeln den Ablauf* bei der Kommunikation zwischen Kommunikationspartnern.
- Grundelemente sind *Nachrichten*, die zwischen den Partnern ausgetauscht werden.
- Eine zusammengehörige Folge von Nachrichten wird *Sitzung* genannt.
- Festgelegt wird
  - das Format von Nachrichten
  - die Reihenfolge der Nachrichten in einer Sitzung
  - das Verhalten, das Nachrichten beim Kommunikationspartner hervorrufen

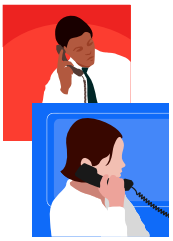
Protokolle

Ein Beispiel aus dem Alltag – ein Telefonanruf:



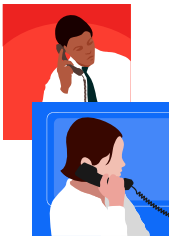
Protokolle und Schichten

Protokolle können auf verschiedenen Ebenen (genannt *Schichten*) angesiedelt werden:



- Wie wird Information physisch übertragen? (gesprochene Sprache über Telefon)
- Behebung von Fehlern ("habe ich nicht verstanden", oder "ja…")
- Regeln des Zugriffs auf das Medium (warten, bis der Gegenüber einen Satz beendet hat, bevor man selbst spricht; wenn ins Wort gefallen, wird der letzte Satz wiederholt)

Protokolle und Schichten



- Verbindungen über Zwischenpunkte (Vermitteln eines Gesprächs über die Telefonzentrale)
- Anpassen der Übertragungsgeschwindigkeit ("das ging mir zu schnell", "bitte warten, ich hole was zu schreiben")
- Wie wird eine Sitzung gestartet, wie beendet? (wählen, "Hallo hier ist…"…; "Auf Wiederhören")
- Interpretation von Inhalten (Sprechen in deutsch oder in englischer Sprache)

Ein Menge von Protokollen auf verschiedenen Schichten wird *Protokollstapel* genannt

Protokolle und Schichten

Referenzmodelle und Protokollstapel

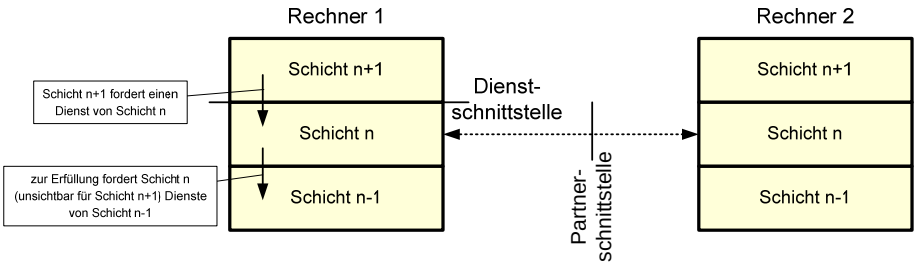
- OSI-Referenzmodell
  - 7 Schichten
  - Mehrere tausend Seiten Spezifikation
- IEEE 802
  - Behandelt die untersten 2 (OSI-)Schichten
  - Beispiele: 802.3 (Ethernet), 802.11 (WLAN)
- TCP/IP-Protokollsuite
  - Referenzmodell mit 4 Schichten
  - Grundlage für das weltweite Internet

Kommunikationsschichten

Idee: Einteilung von komplexen Kommunikations-systemen in *Schichten*:

- Jede Schicht übernimmt eine konkrete Aufgabe (in Form von *Diensten*).
- Die Realisierung der Schicht sowie die Schichten unterhalb einer Schicht bleiben verborgen (Prinzip der Abstraktion).
- Je höher die Schicht, desto mächtiger die Dienste.
- Ersetzen von Realisierungen möglich, wenn die Schnittstellen gleich bleiben.
  - Z.B. E-Mail abrufen funktioniert gleich, egal ob über DSL oder Analogmodem verbunden
  - Austausch von Ethernet gegen WLAN und immer noch den Zugriff auf einen File-Server behalten

Kommunikationsschichten

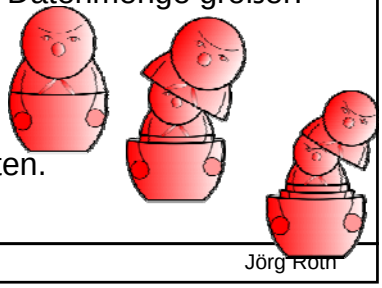


- Auf jeder Schicht werden Daten übertragen (zumindest virtuell).
- Nur auf der untersten Schicht – der physikalischen Schicht – findet eine *reale* Übertragung statt.

Datenübertragung auf den Schichten

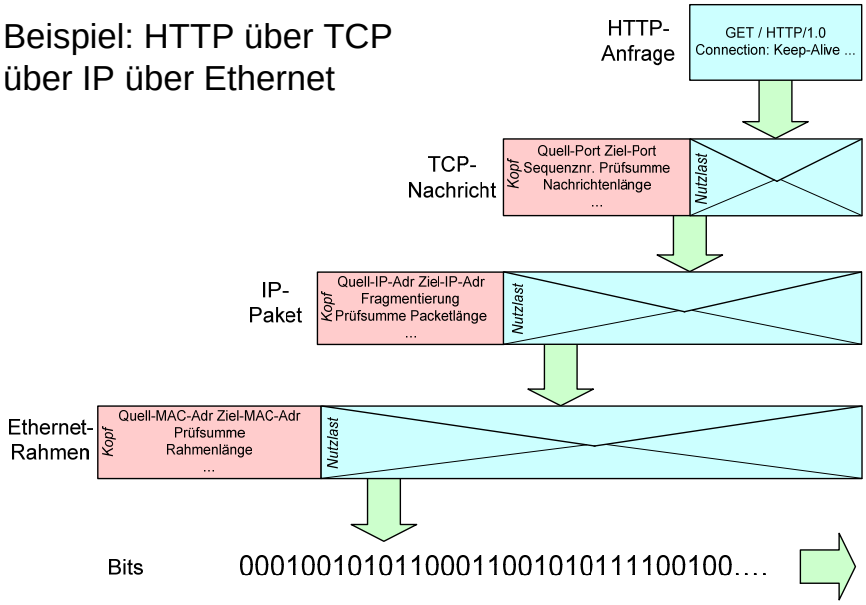
Auf jeder Schicht werden die Daten der Schicht darüber "verpackt":

- Die Daten werden als *Nutzlast* verpackt; für den Inhalt interessiert sich die Schicht nicht.
- Neben der Nutzlast werden *Kopfinformationen* übertragen, zusammen ergibt dies die Nutzlast für die Schicht darunter.
- Auf dem Weg nach unten wird die Datenmenge größer.
- Auf dem Weg nach oben (beim Empfänger) muss jede Schicht ihre Kopfinformationen entfernen, um die jeweilige Nutzlast zu erhalten.



Datenübertragung auf den Schichten

Beispiel: HTTP über TCP über IP über Ethernet



OSI-Referenzmodell

Das OSI-Referenzmodell ist eines der ersten Modelle, das die Kommunikation zwischen Rechnern allgemein beschreibt.

Es sieht 7 Schichten vor:

1. Bitübertragungsschicht
   - Wie werden *Bits* übertragen?
     Festlegung von mechanischen, elektrischen, elektromagnetischen Eigenschaften z.B.
     - Stecker und Kabel
     - Spannungen für Signalpegel
     - Frequenzen

OSI-Referenzmodell

2. Sicherungsschicht

- Einzelne Bits sind für die Übertragung von Daten ungeeignet:
  - einzelne Bits können bei der Übertragung "umkippen"
  - man fasst sie daher zu *Frames* (*Rahmen*) zusammen
- Wie werden Frames *zuverlässig* übertragen? Sicherungen durch redundante Informationen
  - Erkennen von Fehlern, erneutes Anfordern *(Automatic Repeat Request*)
  - Senden mit Redundanz die eine Korrektur erlaubt *(Forward Error Correction)*

OSI-Referenzmodell

3. Vermittlungsschicht

- Schichten 1 und 2 befassen sich nur mit *direkt* (d.h. über ein physikalisches Medium) verbundenen Rechnern
- Schicht 3: Vermittlung von *Paketen*, zwischen Rechnern, die *nicht direkt verbunden* sind $\Rightarrow$ *Routing*

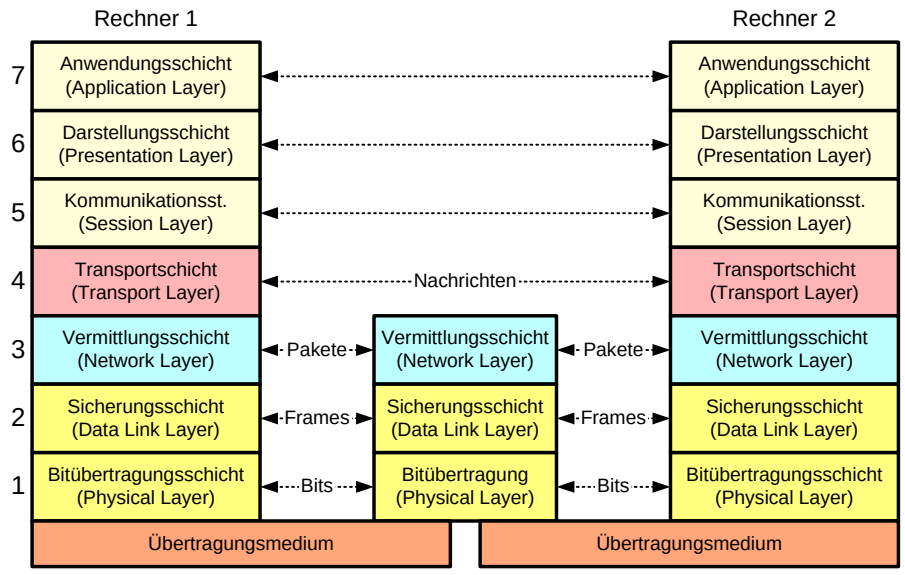
4. Transportschicht

- Bereitstellung zusätzlicher Dienste:
  - Logische Verbindungen
  - Kompensieren von Paketverlusten
  - Vermeidung von Engpässen, Flusskontrolle
  - Dienstgüte

OSI-Referenzmodell

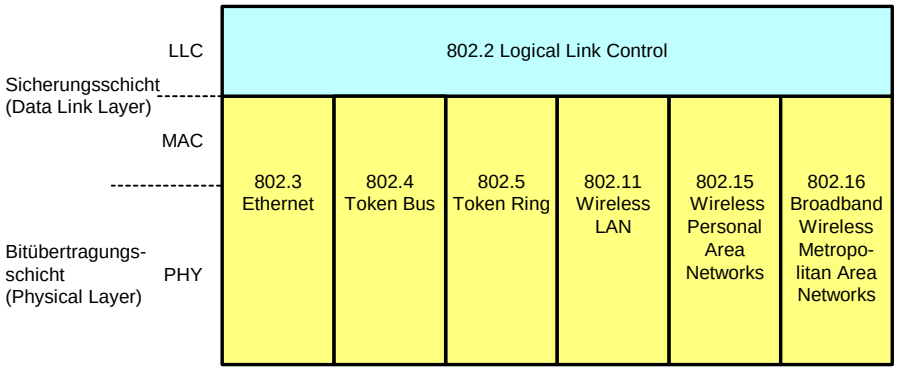
5. Kommunikationssteuerungsschicht
   - Login, Logout
   - Wiederanlauf bei Verbindungsabbrüchen
6. Darstellungsschicht
   - Bedeutung der Binärfolgen
   - Zeichensätze, Zahlendarstellung
7. Anwendungsschicht
   - Höhere Funktionen *für* Anwendungen (hier liegen *nicht* die Anwendungen selbst), z.B.:
     - Benutzerzugangsdienste, Verwalten von Rechten
     - Dateizugriffsdienste

OSI-Referenzmodell



IEEE 802

IEEE begann im Jahr 1980 ("80") im Februar ("2") die Schichten 1 und 2 für verschiedene Netzwerke zu spezifizieren.



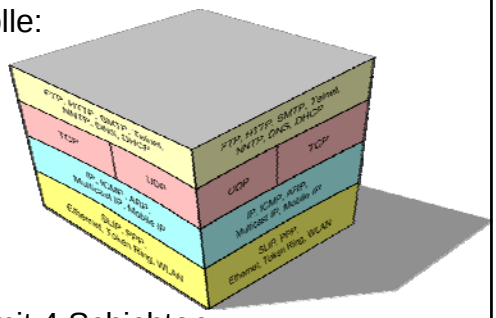
IEEE 802

Die Schichten von IEEE 802:

- *LLC* (*Logical Link Control*): 3 Arten von Logical Links
  - unbestätigt, verbindungslos
  - bestätigt, verbindungslos
  - verbindungsorientiert
- *MAC* (*Media Access Control*): Zugriff auf das gemeinsame Medium, z.B.
  - CSMA/CD (Ethernet)
  - CSMA/CA (WLAN)
- *PHY*: Bitübertragungsschicht

TCP/IP-Protokollsuite

- Wird oft als "Lingua Franca" der Vernetzung angesehen
- Benannt nach den Protokollen
  - *TCP* (*Transmission Control Protocol*)
  - *IP* (*Internet Protocol*)
- Weitere bekannte Protokolle:
  - Telnet
  - FTP
  - SMTP, POP3, IMAP
  - NNTP
  - HTTP

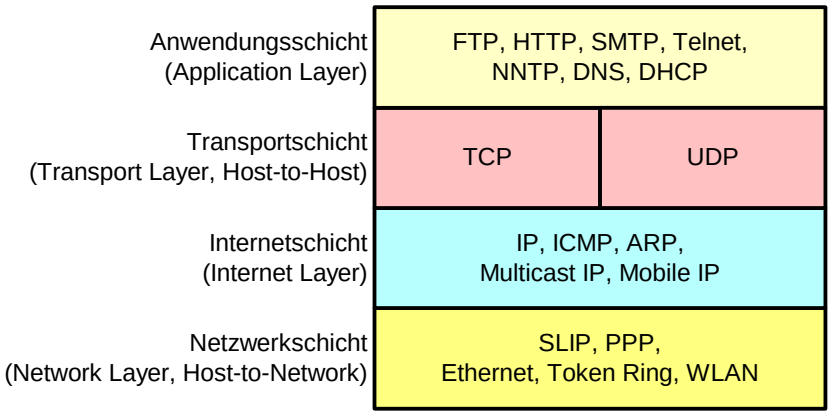


- Eigenes Referenzmodell mit 4 Schichten

TCP/IP-Protokollsuite

OSI-Referenzmodell	TCP/IP-Referenzmodell
Anwendungsschicht (Application Layer)	Anwendungsschicht (Application Layer)
Darstellungsschicht (Presentation Layer)	
Kommunikationssteuerungsschicht (Session Layer)	
Transportschicht (Transport Layer)	Transportschicht (Transport Layer, Host-to-Host)
Vermittlungsschicht (Network Layer)	Internetschicht (Internet Layer)
Sicherungsschicht (Data Link Layer)	Netzwerkschicht (Network Layer, Host-to-Network)
Bitübertragungsschicht (Physical Layer)	

TCP/IP-Protokollsuite

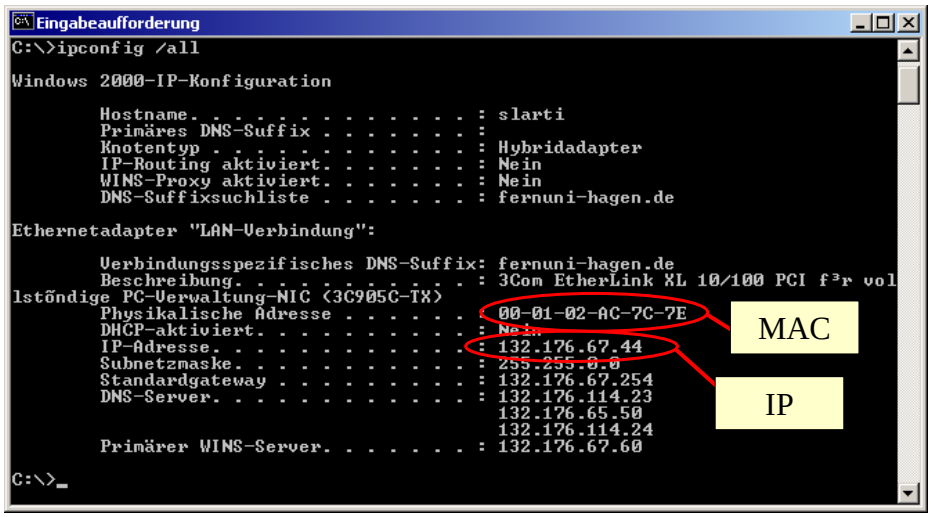


Adressierung

Jede Schicht besitzt ihre eigene Adressierung, Beispiele:

- Schicht 1 und 2: *MAC-Adressen* (48 Bit), z.B. **`00-01-02-AC-7C-7E`**
- Schicht 3: *IP-Adressen* (32 Bit, in Zukunft 128 Bit), z.B. **`132.176.67.44`**
  Alternativ: DNS-Rechnername
- Schicht 4: Zusätzlich zur IP-Adresse: Port (16 Bit), z.B. **`132.176.67.44:5555`**
- Schicht 5-7: *URL*, z.B. **`http://www.ohm-hochschule.de/seitenbaum/home/fachbereiche/page.html`**

Adressierung

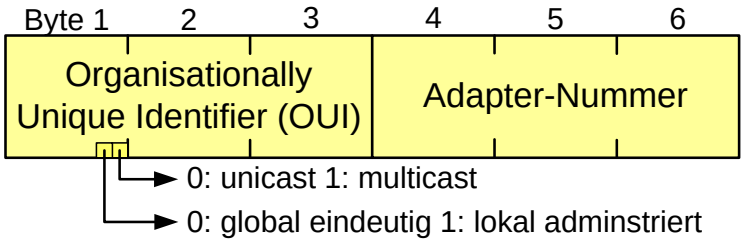


Adressierung

MAC-Adressen (*MAC*: *Media Access Control*)

Genau genommen gibt es zwei Formate:

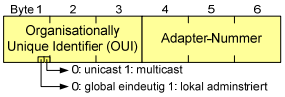
- MAC-48: das "alte", nur für *Netzwerk*-Hardware
- EUI-48 (Extended Unique Identifier): verwendbar für alle Arten kommunizierender Hard- und Software (also auch z.B. Firewire-Adapter)
- MAC-48 und EUI-48 haben denselben Aufbau

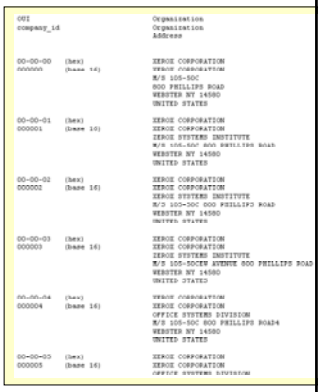


MAC-Adressen

Ersten drei Bytes:

- Organisationally Unique Identifier: eine weltweit zentral administrierte Liste von Herstellern
- das letzte Bit (LSB) des ersten Bytes gibt an: 0: unicast 1: Multicast
- das vorletzte Bit des ersten Bytes:
  - 0: *Universally Administered Address* (fest eingebrannt, der Normallfall)
  - 1: *Locally Administered Address* (eher unüblich)

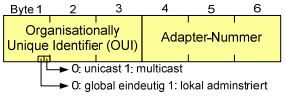




http://standards.ieee.org/develop/regauth/oui/oui.txt

MAC-Adressen

Die weiteren 3 Bytes:



- Die Adapter-Nummer kann vom Hersteller beliebig (aber eindeutig) vergeben werden.
- Es stehen 2^{24}=16.8 mio. Adapter-Nummern pro Hersteller zur Verfügung.
- Man nimmt an, dass der Nummernkreis bis etwa zum Jahr 2100 ausreicht.
- EUI-48 fügt sich nahtlos in das Adressierungsschema EUI-64 für den so genannten *Interface Identifier* von IPv6 ein.

OSI vs. TCP

Disput der Referenzmodelle

- OSI- und TCP-Referenzmodelle waren lange Zeit konkurrierende Ansätze.
- Das OSI-Referenzmodell wurde kein Erfolg, während TCP/IP die Welt der Rechnernetze signifikant geprägt hat.
- Grund: TCP war schnell lauffähig, hauptsächlich durch das Betriebssystem UNIX, OSI-Protokolle waren dagegen schwer zu implementieren.
- Heute ist das OSI-Referenzmodell noch für didaktische Zwecke wertvoll.

Rechnernetze

WS 2012/13

Kapitel 3

Direktverbindungsnetzwerke

– Bitübertragung

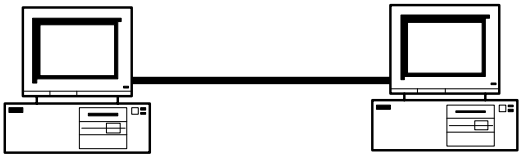
Kapitel 3: Direktverbindungsnetzwerke – Bitübertragung

Ziel: Darstellung von Mechanismen zur Bitübertragung zwischen physikalisch verbundenen Rechnern

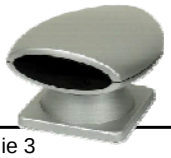
- Übertragungsmedien
- Betriebsweisen
- Kodierung

Direktverbindungsnetzwerke

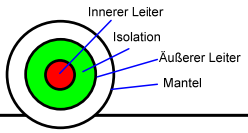
- Das einfachste Netzwerk besteht nur aus zwei Rechnern und einem Übertragungsweg.
- Betroffen sind die OSI-Schichten 1 und 2.
- Physikalische Übertragung:
  - Kabel
  - Lichtwellenleiter
  - Funk
  - Infrarot
  - Laser



Infrarot-Transceiver



Koax-Kabel



Mobilfunk-Station

Beispiel Kabel

Kabel	Maximale Länge	Ethernet Bandbreite
Dünnes Koax-Kabel	200 m	10 MBit/s
Dickes Koax-Kabel	500 m	10 MBit/s
Twisted Pair	100 m	100 MBit -10 GBit/s

Twisted Pair	Grenzfrequenz	Ethernet Bandbreite
Cat 1	100 kHz	
Cat 2	1,5 MHz	
Cat 3	16 MHz	10 MBit/s
Cat 4	20 MHz	
Cat 5	100 MHz	1 GBit/s
Cat 6	250 MHz	
Cat 7	1000 MHz	10 GBit/s

Beispiel Funk

ISM-Bänder (Industrial Scientific Medical) sind teilweise freigegeben

Frequenzband [MHz]	System
13,553-13,567	RFIDs
433,05-434,79	Funkschlüssel
790-860	LTE
865-868	UHF-RFIDs, Zigbee
890-915	GSM (GSM 900)
935-960	GSM (GSM 900)
1227,6	GPS
1575,42	GPS
1710-1785	GSM (DCS 1800)
1805-1880	GSM (DCS 1800)
1880-1900	DECT
1900-1920	UMTS (UTRA-TDD)
1920-1980	UMTS (UTRA-FDD)
2010-2025	UMTS (UTRA-TDD)
2110-2170	UMTS (UTRA-FDD)
2400-2483,5	WLAN 802.11b, Bluetooth, Zigbee
2500-2690	LTE
5150-5250	WLAN 802.11a
5725-5825	WLAN 802.11a

Beispiel Infrarot

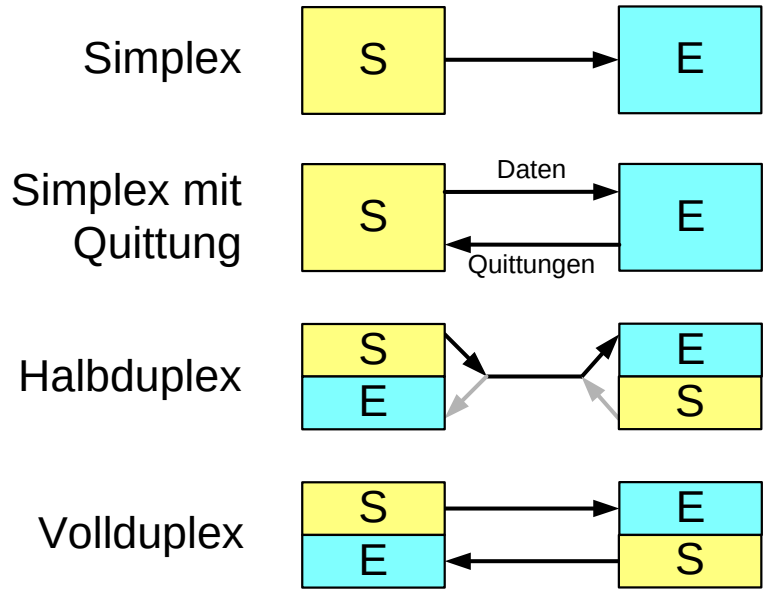
- Die Infrarotkommunikation wurde schon im Jahr 1979 von HP zur Kopplung von Taschenrechner (HP-41C) und Drucker eingesetzt.
- Lange Zeit war sie in fast allen Notebooks, PDAs und vielen Mobiltelefonen verfügbar.
- Mittlerweile abgelöst durch Bluetooth.

Betriebsweisen

Synchron vs. asynchron:

- *Synchron*: Sender und Empfänger synchronisieren die Übertragung, d.h. der Sender darf nicht zu beliebigen Zeitpunkten senden.
  - Zentraler Takt
  - Explizite Sendefreigabe durch den Empfänger
- *Asynchron*: der Sender kann zu beliebigen Zeitpunkten senden.
  - Beispiel Tastatur
  - Start-Stop-Erkennung notwendig
  - In der Regel langsamer als synchron

Betriebsweisen



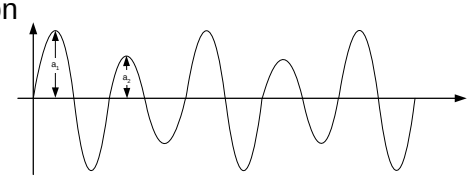
Die Bitübertragungsschicht

Auf der Bitübertragungsschicht wird vor allem festgelegt, wie Bits dargestellt werden, z.B.:

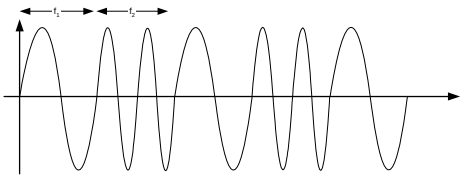
- *Modulation*: Darstellung von Binärzeichen durch Variation von *Amplitude*, *Frequenz* oder *Phase* eines sinusartigen Signals
  - *Amplitudenmodulation*: Variation der Amplitude
  - *Frequenzmodulation*: Variation einer Grundfrequenz
  - *Phasenmodulation*: Kodieren der Binärdaten durch Phasensprünge

Modulation

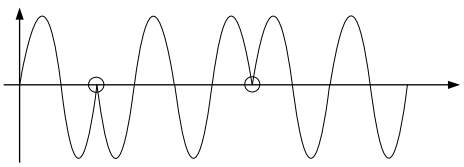
- Amplitudenmodulation



- Frequenzmodulation



- Phasenmodulation



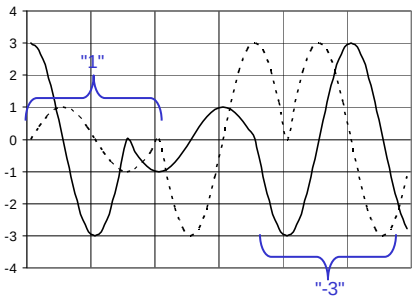
Modulation

Ausgefeiltere Modulationsart *16-QAM* (Quadraturamplitudenmodulation)

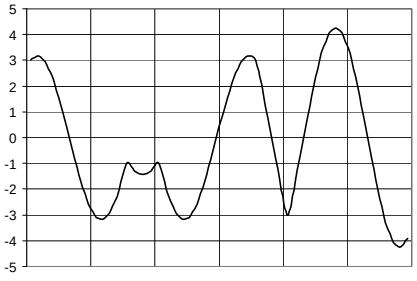
- Wird z.B. bei "Highspeed" UMTS verwendet
- Kombination aus Phasen- und Amplitudenmodulation
- Idee:
  - man verwendet zwei Sinusschwingungen, die um 90° gegeneinander verschoben sind
  - jede einzelne verwendet 4 Amplituden (-3, -1, 1, 3), erlaubt also 2 Bits zu kodieren
  - beide Schwingungen werden addiert, man kann damit 4 Bits pro Symbol kodieren

Modulation

Einzelne Signale

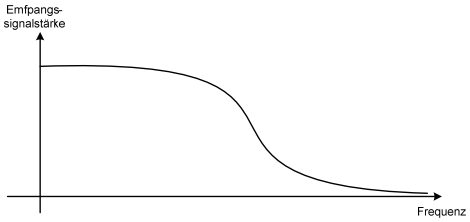


Summe



Theoretische Obergrenzen für Datenraten

Beobachtung: Reale Übertragungskanäle erlauben nicht die Übertragung beliebig großer Frequenzen



Wir idealisieren:

- Signale werden bis zu einer bestimmten Frequenz ohne Dämpfung übertragen.
- Frequenzen darüber werden abgeschnitten.
- Die maximal übertragbare Frequenz wird *Bandbreite* genannt.

Theoretische Obergrenzen für Datenraten

Beispiele:

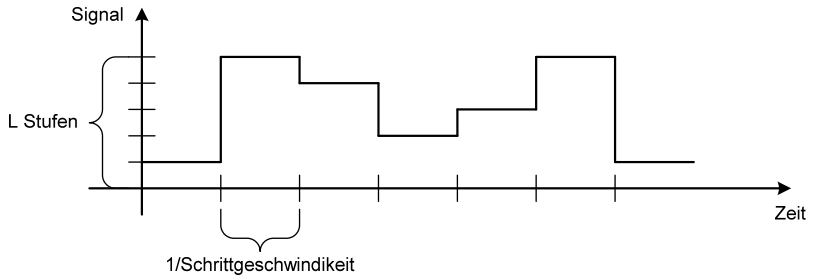
- Cat5 Kabel (Kabel für Fast- und GBit-Ethernet): 100 MHz
- ADSL2+ (DSL mit 16MBit/s): 2,2 MHz

Wir nehmen zuerst an, dass der Übertragungskanal *nicht* gestört ist.

- Ein Übertragskanal der Bandbreite *B* (*Nyquist-Frequenz*) gestattet eine maximal Schrittgeschwindigkeit v_{max} von

$$v_{max} = 2 \cdot B$$

Theoretische Obergrenzen für Datenraten



Besteht ein Signal aus L diskreten Stufen, so kann man eine maximale Datenrate D_{max} von

$$D_{max} = 2 \cdot B \cdot \log_2(L)$$

Theoretische Obergrenzen für Datenraten

Bemerkung:

- Mit diesem Modell sind beliebig hohe Datenraten denkbar (wenn man beliebig viele diskrete Signalstufen einsetzt).

Wir nehmen jetzt an, dass der Übertragungskanal durch Rauschen gestört wird.

Beobachtung:

- In der Realität gibt es nicht nur eine Bandbreiten-Begrenzung – ein Übertragungskanal "überträgt" neben dem Nutzsignal ein Rauschsignal, das das Nutzsignal stört.

Theoretische Obergrenzen für Datenraten

Man modelliert einen gestörten Kanal durch den so genannten *Signal-Rausch-Abstand* (*Signal-Noise-Ratio*, *SNR*)

$$SNR = S/N$$

mit

- *S*: Signalstärke des Nutzsignals
- *N*: Signalstärke des Rauschens

Da man oft sehr große Verhältnisse erhält, verwendet man zur besseren Vergleichbarkeit die Pseudo-Einheit dB

$$SNR\ [\text{dB}] = 10 \cdot \log_{10}(S/N)$$

Theoretische Obergrenzen für Datenraten

Beispiele:

- DSL-Verbindungen sollten einen Rauschabstand größer 6 dB haben (entspricht 4/1)
- Gute Telefonleitungen haben 30 dB (entspricht 1000/1)

Shannon-Hartley-Gesetz:

(maximale Datenrate bei bandbreitenbegrenztem, gestörten Übertragungskanal)

$$D_{max} = B \cdot \log_2(1+S/N)$$

Achtung:

- die max. Datenrate hängt nur noch von der Nyquist-Frequenz und dem Signal-Rausch-Abstand ab, *nicht* mehr von der Zahl der Signalstufen

Theoretische Obergrenzen für Datenraten

Beispiel:

- Analogtelefon mit Bandbreite 3000 Hz und Signal-Rausch-Abstand von 30 dB (entspricht 1000/1):

$$D_{max} = B \cdot \log_2(1+S/N)$$
$$= 3000\ \text{Hz} \cdot \log_2(1+1000)$$
$$\approx 30000\ \text{Bit/s}$$

Bemerkung:

- In der Analog-Modem-Zeit waren auch Datenraten bis 56000 Bit/s möglich, da moderne Vermittlungsstellen bessere Qualitäten unterstützten.

Theoretische Obergrenzen für Datenraten

Beispiel:

GBit-Ethernet verwendet Cat5-Kabel

- 4 Adernpaare für die Datenübertragung
- Genutzte Bandbreite 62,5 MHz
- 5 Signalstufen

Ungestörter Kanal:

$$D_{max} = 2 \cdot B \cdot \log_2(L)$$
$$= 2 \cdot 62{,}5\ \text{MHz} \cdot \log_2(5)$$
$$= 290\ \text{MBit/s}$$

Bei 4 Adernpaare wären das theoretisch 1160 MBit/s

Theoretische Obergrenzen für Datenraten

- Von den 5 Signalstufen werden nicht alle $5^4 = 625$ Kombinationen (bei 4 Adernpaaren) zur Datenübertragung genutzt
- Es werden 256 Codes (8 Bit) unterschieden, die restlichen Kombinationen dienen der Fehlerkorrektur
- Daher ist die Datenrate effektiv 1000 MBit/s

Gestörter Kanal:

- Wir nehmen einen Signal-Rausch-Abstand von 20 dB (100/1) an

$$\begin{aligned} D_{max} &= B \cdot \log_2(1+S/N) \\ &= 62{,}5\ \text{MHz} \cdot \log_2(1+100) \\ &= 416\ \text{MBit/s} \end{aligned}$$

Multiplexverfahren

Soll ein Übertragungsmedium für mehrere Kanäle eingesetzt werden, muss man *multiplexen*

- *Frequenzmultiplex*
  (*Frequency Division Multiplex*, *FDM*)
- *Zeitmultiplex*
  (*Time Division Multiplex*, *TDM*)

Bei drahtlosen Medien gibt es zusätzlich:

- *Raummultiplex*
  (*Space Division Multiplex*, *SDM*)
- *Codemultiplex*
  (*Code Division Multiplex*, *CDM*)

Häufig werden diese Verfahren kombiniert

Multiplexverfahren

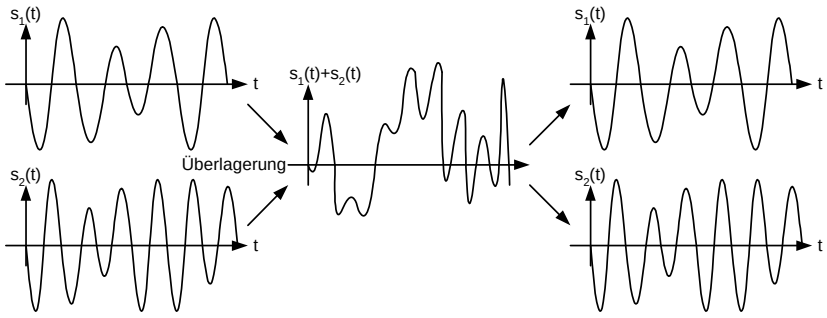
Duplex vs. Multiple Access

- *Duplex*: 2 Kommunikationspartner greifen auf einen bidirektionalen Kanal zu
- *Multiple Access*: mehrere Sender

Multiplex	Duplex	Multiple Access
Frequenz	FDD	FDMA
Zeit	TDD	TDMA
Raum	-	SDMA
Code	-	CDMA

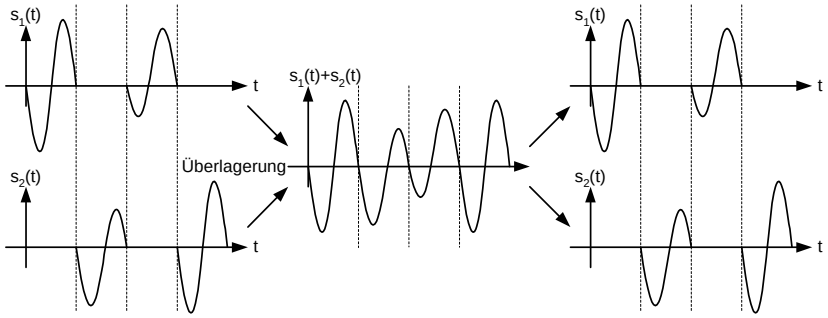
Frequenzmultiplex

Unterschiedliche Frequenzen können beim Empfänger wieder getrennt werden. Dieser Effekt wird z.B. beim Rundfunk ausgenutzt.



Zeitmultiplex

Sender benutzen *nacheinander* das Übertragungsmedium für eine bestimmte Zeit (genannt *Slot*)



Zeitmultiplex

Unbeabsichtigte Mehrfachbelegung des Mediums (genannt *Kollision*) macht die Übertragung unbrauchbar.

Möglichkeiten der Synchronisation:

- Zentrale Verfahren:
  ein ausgezeichneter Sender erteilt das Senderecht
  $\Rightarrow$ sinnvoll, wenn es sowieso schon einen ausgezeichneten Sender gibt
- Dezentrale Verfahren: die Sender vereinbaren untereinander die Sendereihenfolge
  $\Rightarrow$ flexibler aber komplexer
- U.U. müssen Signallaufzeiten aufgrund des Abstandes von Sender und Empfänger berücksichtigt werden

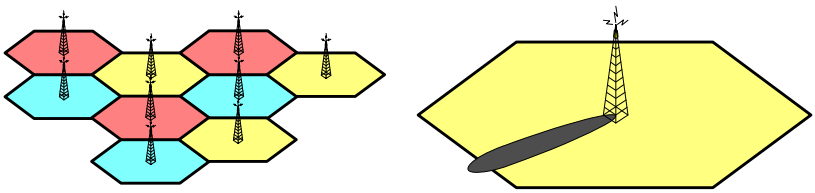
Raummultiplex

Im mobilen Umfeld (z.B. Mobiltelefonie, Wireless LANs) verwendet man Raummultiplex:

- Die Wirkung elektromagnetischer Signale nimmt mit dem Abstand ab, idealisiert: quadratisch, real: 4te Potenz.

$\Rightarrow$ Dieselben Funkressourcen können in einem "Sicherheitsabstand" noch einmal verwendet werden.

- Antennen mit Richtungscharakteristik



Codemultiplex

Funk-Frequenzen sind sehr wertvolle Ressourcen
Beispiel: Versteigerung der UMTS-Lizenzen in Deutschland $\Rightarrow$ 16 Mrd. DM/Frequenz

- Codemultiplex: *Gleichzeitige* Übertragung auf *einer* Frequenz. Einsatz z.B. UMTS, Satellitennavigation GPS

Nur die Grundidee (ohne Details):

- Jeder Sender *spreizt* das Signal mit einem eigenen Code.
- Ein Empfänger kann einen bestimmten Code aus einer Überlagerung herausfiltern.
- Übertragungen anderer Codes werden als Rauschen ausgefiltert.

Sicherung der Übertragung

5 Probleme müssen gelöst werden:

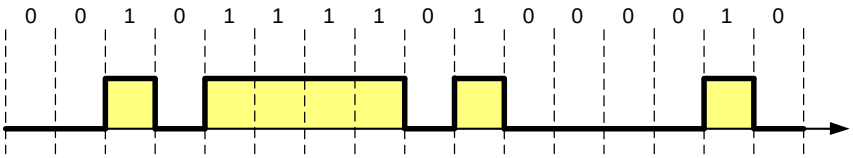
- *Kodierung*: wie müssen Bits kodiert werden, damit sie auf der Gegenseite verstanden werden?
- *Framing*: wie werden Bits zu Frames zusammengefasst?
- *Fehlererkennung*: Wie erkennt man, ob Frames bei der Übertragung verfälscht wurden?
- *Zuverlässige Zustellung*: Wie erreicht man eine Zustellung trotz Fehler?
- *Medienzugriff*: Wie greift man auf ein Medium zu, auf das auch andere Sender zugreifen möchten?

Die Funktionen aller fünf Fragestellungen wird heutzutage auf der Netzwerkkarte (bzw. Chipsatz) bereitgestellt.

Kodierung – NRZ

Kodierung von Bits:

- Einfache Idee: 0-Bit → low, 1-Bit → high
- high, low bezeichnen Signalpegel, z.B. low: 0V, high +5V



- Dieses Verfahren heißt (rätselhafterweise) *Non-Return to Zero* (*NRZ*)

Kodierung – NRZ

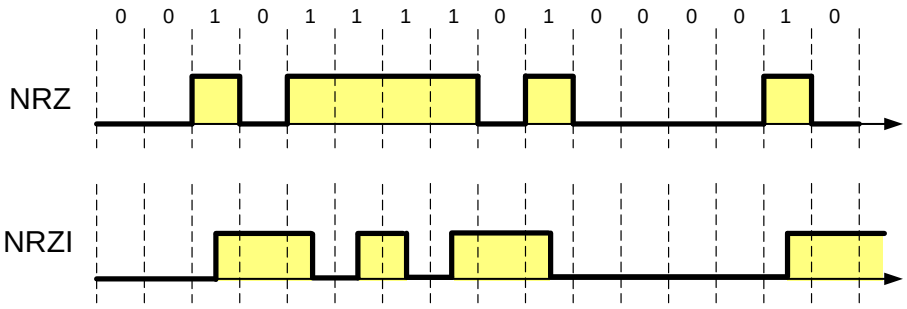
Nachteile: lange Sequenzen von 0 oder 1

- Der Empfänger berechnet den Durchschnitt der letzten Signale, um zwischen 0 und 1 unterscheiden zu können – bei langen Sequenzen desselben Pegels wird dieser Durchschnitt ungünstig verschoben.
- Taktwiederherstellung: der Empfänger benutzt die Wechsel zwischen 0 und 1, um sich auf den Takt des Senders zu synchronisieren. Bei langen Phasen ohne Wechsel driften die Takte auseinander.
- Mögliche Lösung: Takt über eigene Leitung transportieren
  $\Rightarrow$ höhere Verkabelungskosten

Kodierung – NRZI

Alternative: *Non-Return to Zero Inverted* (*NRZI*)

- 0-Bit $\Rightarrow$ Wiederholung des letzten Signals
- 1-Bit $\Rightarrow$ Wechsel des Signals

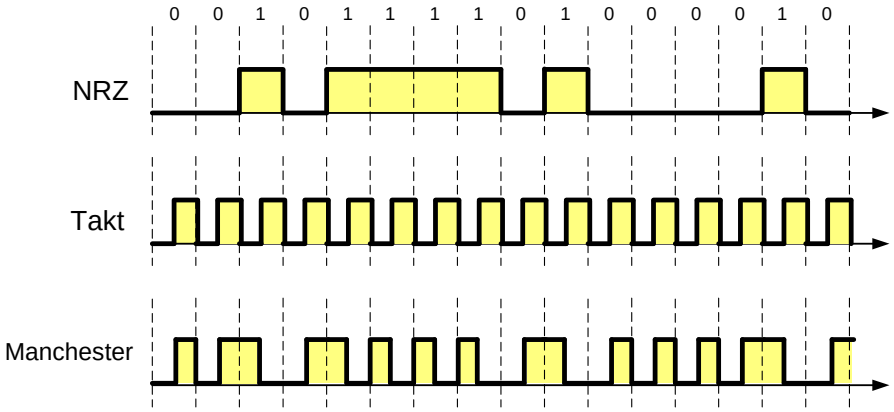


- Lange 0-Sequenzen führen immer noch nicht zu einem Wechsel

Manchester-Kodierung

Manchester-Kodierung:

- 0-Bit ⇒ Signal wechselt von low → high
- 1-Bit ⇒ Signal wechselt von high → low



Manchester-Kodierung

- Weitere mögliche Interpretation: XOR von NRZ und Takt
- Vorteil: in jedem Takt ein Wechsel
- Nachteil: die Änderungsrate der Signalwechsel (genannt *Baudrate*) verdoppelt sich
  - Manchester: Bitrate = Baudrate/2
  - NRZ, NRZI: Bitrate = Baudrate
- Aus der Sicht der Nachrichtentechnik bedeutet dies, die Kanal-Bandbreite zu verdoppeln (d.h. höhere Frequenzen müssen durch das Medium transportiert werden).

4B/5B-Kodierung

4B/5B:

- Jeweils 4 Daten-Bits werden 5 Code-Bits zugeordnet.
- Maximal eine 0 am Anfang, maximal zwei 0en am Ende ⇒ nie mehr als drei 0en in Folge
- Codes werden mit NRZI übertragen (NRZI hat das Problem vieler 1en ja schon gelöst).
- Von den 32 möglichen Codes werden nur 16 benutzt – die anderen 16 können für Steuersymbole verwendet werden (z.B. 11111: Leitung untätig).

4-Bit-Daten	5-Bit-Code
0000	11110
0001	01001
0010	10100
0011	10101
0100	01010
0101	01011
0110	01110
0111	01111
1000	10010
1001	10011
1010	10110
1011	10111
1100	11010
1101	11011
1110	11100
1111	11101

Ähnliche Kodierungen

8B/10B:

- 8 Bit Daten werden jeweils mit 10 Bits dargestellt
- eingesetzt bei GBit-Ethernet (nicht Twisted Pair)
- die 10-Bit-Kombinationen nutzen nur 0-Bits/1-Bits mit folgender Verteilung: 4/6, 5/5, 6/4
- im Mittel werden dadurch gleich viele 0-Bits und 1-Bits gesendet ("gleichstromfrei", siehe später)
- maximale Anzahl gleicher Bits: 5

64B/66B:

- eingesetzt bei GBit-, 10-GBit-Ethernet
- kleinerer Overhead

Ternäre Codes

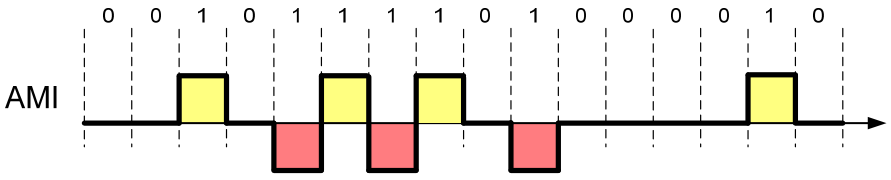
Ternäre Codes:

- Es gibt drei Signalzustände: "-1", "0" und "1", z.B. +/-5 V und 0 V (stromlos).
- Ziel der ternären Code: möglichst *gleichstromfrei* übertragen
  - Gleichstromanteile können nicht über lange Leitungen transportiert werden.
  - Transformatoren auf dem Übertragungsweg löschen Gleichstromanteile.
- Ideal:
  - möglichst häufige Wechsel bezüglich -1, 1
  - Statistisch: -1, 1 gleich häufig

AMI-Code

AMI-Code (*Alternate Mark Inversion*)

- 0-Bit $\Rightarrow$ 0-Signal
- 1-Bit $\Rightarrow$ 1- oder -1-Signal, Wechsel gegenüber dem letzten 1-Bit



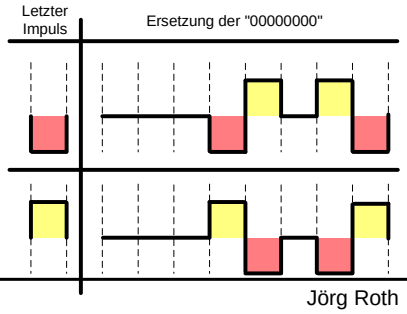
- Nachteil: keine Taktwiederherstellung bei langen 0-Ketten

B8ZS-Code

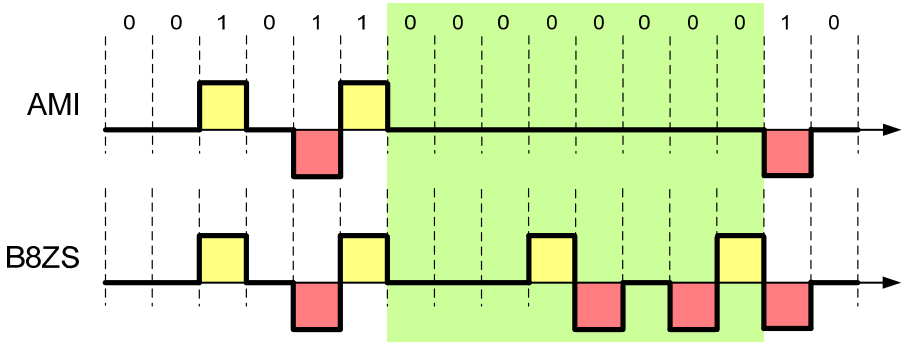
- Bei AMI muss einer 1 (abgesehen von 0en) eine -1 folgen (u.U.)
  - man kann diese Eigenschaft gezielt *verletzen*: 10…01 und -10…0-1
  - Code-Verletzungen können verwendet werden, um lange 0-Folgen anzuzeigen.

B8ZS (*Bipolar With 8 Zeros Substitution*):

- Jeweils acht 0-Bits werden durch Code-Verletzungen kodiert.
- Sonst wie AMI.



B8ZS-Code

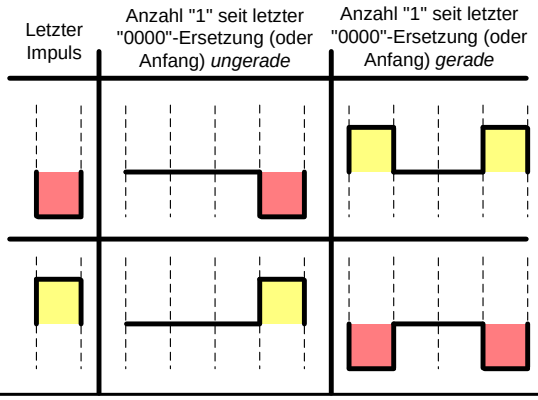


- Es werden maximal sieben 0-Signale hintereinander gesendet

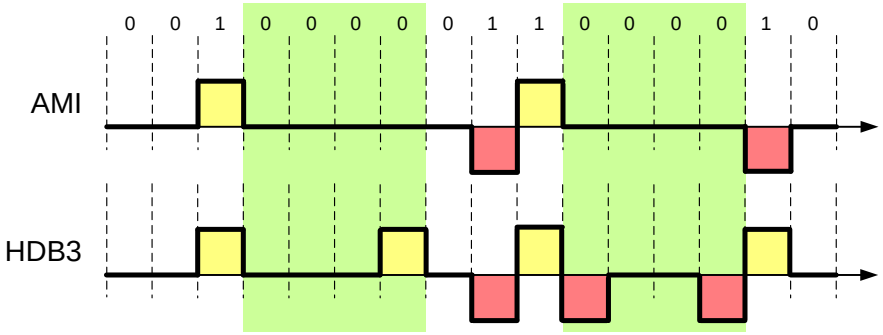
HDB3-Code

HDB3 (*High Density Bipolar 3*):

- Es werden maximal drei 0-Signale hintereinander gesendet.
- Vier 0-Bits werden jeweils ersetzt:



HDB3-Code



- Sonst wie AMI.
- Warum die gerade/ungerade-Unterscheidung?
  - Ohne diese Unterscheidung könnten die Code-Verletzer immer denselben Pegel (z.B. immer 1) haben.

Rechnernetze

WS 2012/13

Kapitel 4

Direktverbindungsnetzwerke

– Sicherung der Übertragung

Kapitel 4: Direktverbindungsnetzwerke – Sicherung der Übertragung

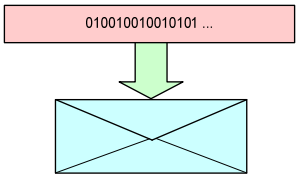
Ziel: Darstellung von Mechanismen zur Sicherung von Übertragungen zwischen physikalisch verbundenen Rechnern

- Aufbau von Rahmen
- Erkennen von Fehlern
- ARQ und Sliding Window

Erzeugen von Frames

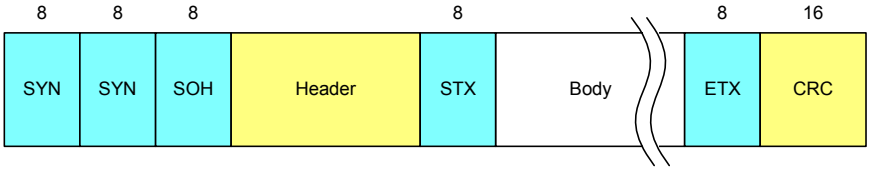
Es ist nicht sinnvoll, die Nutzdaten "unverpackt" zu versenden, Datenblöcke werden daher in *Frames* (zu deutsch *Rahmen*) verpackt und übertragen

- Möglichkeiten, Frames aufzubauen:
  - Byte-orientiert (die ältere Methode)
    - Sentinel-Methode
    - Byte-Zähl-Methode
  - Bit-orientiert
    - HDLC



Sentinel-Methode

- Die Sentinel-Methode:
  - Spezielle *Sentinel*-Zeichen (Start of Text, STX; End of Text, ETX) rahmen die Nutzdaten ein
  - Beispiel: *BISYNC* (*Binary Synchronous Communication*), IBM, Ende der 60er Jahre



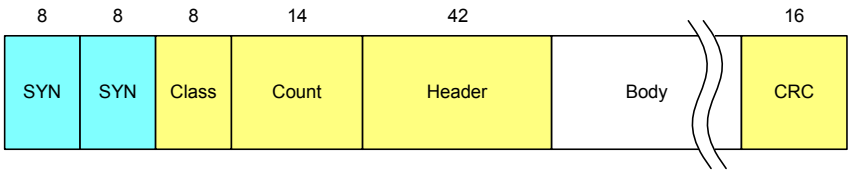
- Problem: was, wenn Sentinel-Zeichen im Nutzdatenteil vorkommen?

Sentinel-Methode

- *Zeichenstopfen* (*Character Stuffing*)
  - ETX wird durch DLE ETX im Nutzdatenteil kodiert (DLE: Data Link Escape)
  - DLE selbst muss durch DLE DLE dargestellt werden
- CRC (Cyclic Redundancy Check): dient der Erkennung von Übertragungsfehlern (Beschreibung folgt).
- Frame-Fehler: Übertragungsfehler in Sentinel-Zeichen, z.B. ETX wird nicht erkannt $\Rightarrow$ der Nutzdatenteil wird weitergelesen, bis das nächste ETX kommt.
- Erkennung der Frame-Fehler:
  - nach dem nächsten ETX wird die CRC-Summe auf den vermeintlichen Datenblock angewendet

Byte-Zähl-Methode

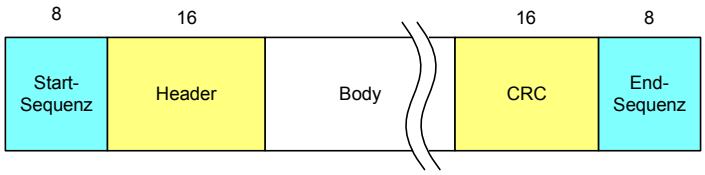
- Byte-Zähl-Methode
  - Kein Ende-Zeichen, statt dessen wird die Anzahl der Bytes in der Nachricht angegeben.
  - Beispiel: *DDCMP* (*Digital Data Communication Message Protocol*), DECNET



- Frame-Fehler: z.B. Fehler im Count-Teil

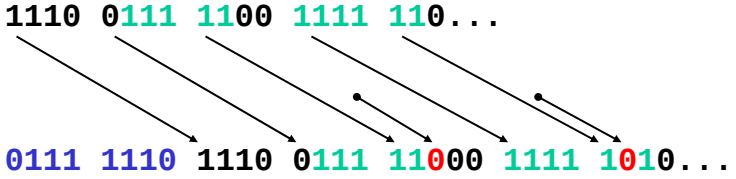
HDLC

- Bitorientierte Frame-Erzeugung: *HDLC* (*High-Level Data Link Control*)
  - Nachfolger von *SDLC* (*Synchronous Data Link Control*) von IBM
  - Keine Byte-Grenzen: die Nutzdaten werden als eine Folge von Bits angesehen



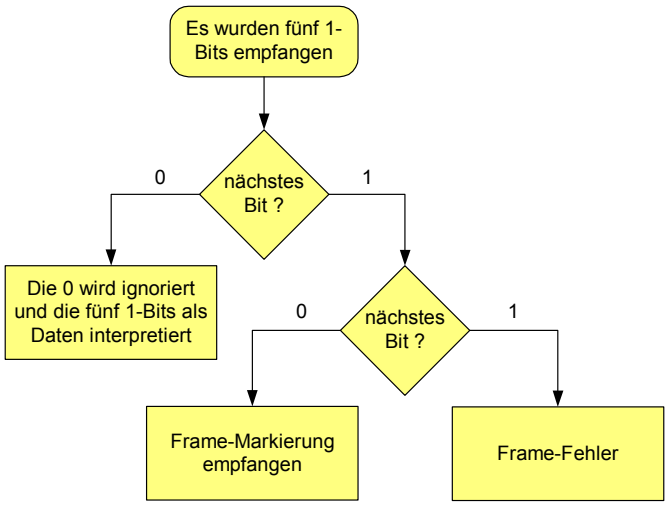
HDLC

- Start- und Ende-Sequenz: 01111110
- *Bit-Stopfen* (*Bit Stuffing*): kommen fünf 1-Bits im Datenteil vor, so wird nach jedem fünften Bit ein 0-Bit eingefügt
- Genauer:
  - 01111110: als Start- und Ende-Sequenz bleibt 01111110
  - 11111 im Datenteil wird zu 111110
  - Z.B. Senden der Daten



HDLC

- Vorgehen beim Empfang



Erkennen von Fehlern

- Der Empfänger eines Frames muss die Möglichkeit haben, Fehler zu erkennen, um sie ggfs. nachfordern zu können.
- Hierzu wird neben den Nutzdaten zusätzlich *redundante* Information übertragen.
- Beispiele:
  - Parität
  - Zyklische Blocksicherung
- Anderer Ansatz: die Übertragung enthält genügend Redundanz, um einige Fehler kompensieren zu können (*Forward Error Correction*, *FEC*)
  - Faltungscodes
  - Hamming-Codes

Parität

Einfache (eindimensionale) Parität

- Die Bits eines Informationsblocks (z.B. eines Bytes) werden um 1 Bit erweitert.
  - *Gerade Parität* (*Even Parity*): das zusätzliche Bit wird so gewählt, dass die Anzahl der 1-Bits gerade ist
  - *Ungerade Parität* (*Odd Parity*): umgekehrt
  - Z.B.
    1011 010 → **1011 0100** (Even Parity)
    0110 110 → **0110 1101** (Odd Parity)
- Nachteil: schon 2 umgekippte Bits können nicht erkannt werden.

Parität

Zweidimensionale Parität

- Zu jedem Paritätsbits einzelner Informationsblöcke wird noch mal die Parität über alle Blöcke gebildet.
- Alle 1-, 2- und 3-Bit-Fehler können erkannt werden.
- Die meisten 4-Bit-Fehler werden erkannt.

0101 101	0
0110 100	1
0111 111	0
0000 000	0
1111 111	1
0100 101	1
0111 111	0
1000 011	1

TCP/IP-Prüfsummen

Prüfsummen in IP-Paketen und TCP-Nachrichten:

- Für TCP/IP wurde ein eigenes Prüfsummen-Verfahren entwickelt – Ziel:
  - einfach zu berechnen
  - einfach zu überprüfen
- 16-Bit-Blöcke werden durch eine 16-Bit-Prüfsumme gesichert.
- Das Verfahren:
  - Einer-Komplement-Summe über alle 16-Bit-Blöcke
  - davon das Einer-Komplement bilden (alle Bits invertieren)

TCP/IP-Prüfsummen

Achtung: der Begriff "Einer-Komplement" beschreibt zweimal etwas anderes:

- *Einer-Komplement bilden*: Bit-weise invertieren
- *Summe im Einer-Komplement*: Summenbildung in einer bestimmten Darstellung von Zahlen (insb. von negativen Zahlen)

Zur Erinnerung: *Einer-Komplement-Darstellung*

- Positive Zahlen werden durch die 2-adische Entwicklung dargestellt
- Negative Zahlen: Betrag als 2-adische, Entwicklung, bitweise invertiert

TCP/IP-Prüfsummen

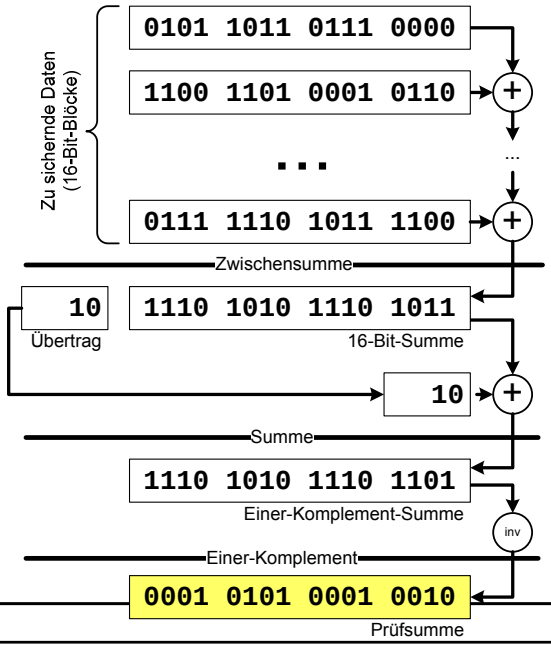
Zwei Darstellungen des Wertes 0 in der Einer-Komplement-Darstellung:

- **`0…0`**
- **`1…1`** (eigentlich "-0")

Addition in der Einer-Komplement-Darstellung:

- Bitweise addieren
- Übertrag an Einer-Stelle auf Summe addieren
- Addiert man mehrere Summanden auf einmal, kann die Übertrags-Addition wieder zu einem Übertrag führen, der wieder addiert werden muss

TCP/IP-Prüfsummen



TCP/IP-Prüfsummen

- In der Einer-Komplementdarstellung bedeutet das Invertieren aller Bits eine Multiplikation mit -1.
- Für zu sichernde Felder $field_i$ gilt damit:

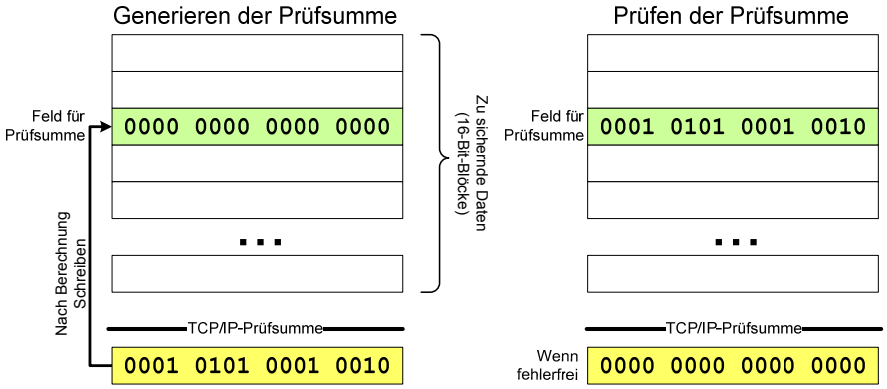
$$checksum = -\sum field_i$$

oder

$$checksum + \sum field_i = 0$$

- Vorgehensweise beim Senden: setze das *checksum*-Feld auf **0…0**, bilde Prüfsumme über alle Felder und schreibe das Ergebnis in das *checksum*-Feld.
- Vorgehen beim Prüfen: bilde die Prüfsumme über alle Felder (inkl. dem *checksum*-Feld). Wenn kein Fehler vorliegt, muss **0…0** herauskommen.

TCP/IP-Prüfsummen



- Bei der Überprüfung $checksum + \sum field_i = 0$ könnte im ersten Ansatz auch **1…1** als 0 herauskommen – das kann aber nicht passieren (Übungsaufgabe).
- Achtung: *checksum*-Feld muss im 16-Bit-Raster liegen

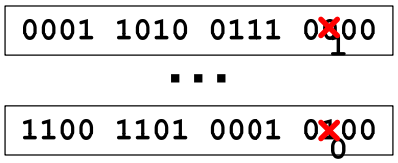
TCP/IP-Prüfsummen

Welche Fehler werden durch das Verfahren erkannt:

- Maximal 15 aufeinanderfolgende Fehlerbits
- Fast alle 16 aufeinanderfolgende Fehlerbits, Ausnahme: 16 Bits **0…0** ↔ **1…1**

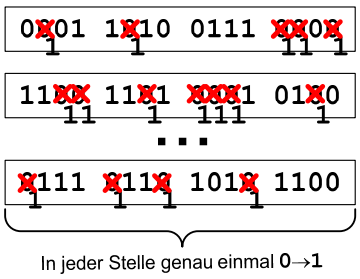
Welche Fehler werden z.B. *nicht* erkannt:

- 2 Fehlerbits mit **0**→**1** und **1**→ **0**, die im Abstand von $n \cdot 16$ vorkommen (n>0)



TCP/IP-Prüfsummen

- 16 Fehlerbits **0**→**1**, die bei allen 16 Stellen vorkommen (außer: alle anderen Bits sind **0**).



- Analog: 16 Fehlerbits **1**→**0**, die bei allen 16 Stellen vorkommen (außer: alle anderen Bits sind **0**).

Zusammenhang für das Nicht-Erkennen von Fehlern:

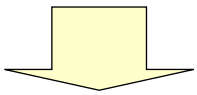
- Jede Veränderung, die die *Zwischensumme* um 0 oder Vielfache von 65535 erhöht, wirken sich nicht auf die Prüfsumme aus.

Zyklische Blocksicherung

Zyklische Blocksicherung, *Cyclic Redundancy Check* (*CRC*)

- Sehr beliebtes Verfahren, vielfach auch außerhalb von Netzwerken eingesetzt, z.B. bei Dateisystemen.
- Grundidee: wir interpretieren eine Nachricht der Länge n+1 durch ein binäres Polynom des Grades n, z.B.

1011 0011



$x^7 + x^5 + x^4 + x + 1$

Zyklische Blocksicherung

- Rechnungen werden grundsätzlich *modulo 2* durchgeführt. Einige Rechenregeln:
  - 1 + 1 = 0
  - a + b = a – b für beliebige a, b
- Wir unterscheiden folgende Polynome:
  - Das *Informationspolynom* I(x) vom Grad n repräsentiert die übertragenen Nutzdaten.
  - Das *Generatorpolynom* G(x) vom Grad k ist vorher von Sender und Empfänger vereinbart worden und ist fest.
  - Das *Codepolynom* C(x) wird vom Sender an den Empfänger übertragen und enthält neben den Nutzdaten die redundante Information.

Zyklische Blocksicherung

Folgende Schritte werden durchgeführt:

- Wir multiplizieren das Informationspolynom mit x^k und teilen das Ergebnis durch das Generatorpolynom. Dies führt zu folgender Darstellung:

  $$I(x)\cdot x^k = P(x)\cdot G(x)+R(x)$$

  Hiervon interessiert uns nur das Restpolynom R(x).

- Wir übertragen das Codepolynom

  $$C(x) = I(x)\cdot x^k + R(x)$$

Zyklische Blocksicherung

- Der Empfänger teilt das Codepolynom durch das Generatorpolynom. Bei fehlerfreier Übertragung darf kein Rest entstehen, es gilt nämlich

  $$\begin{aligned} C(x) / G(x) &= (I(x)\cdot x^k + R(x)) / G(x) \\ &= (P(x) \cdot G(x) + R(x) + R(x)) / G(x) \\ &= (P(x) \cdot G(x)) / G(x) \\ &= P(x) \end{aligned}$$

- Es gilt: Bei Generatorpolynomen $G(x) = x^k+...+1$ (d.h. erster und letzter Koeffizient sind 1), wird ein *Fehlerbündel der Länge k* sicher erkannt.

Zyklische Blocksicherung – Fehlerbündel

Länge eines Fehlerbündels: Abstand des ersten und letzten fehlerhaften Bits

Gesendet: **0101001110101010.....10110101010101**
Empfangen: **0101011110101010.....10110100010101**

Fehlerbündel

- Selbst wenn nur 2 Bits umgekippt sind, kann dieser Fehler u.U. nicht erkannt werden, wenn diese zu weit entfernt sind
  - möglich: kürzere Blöcke einzeln absichern
  - reale Fehler treten lokal auf

Zyklische Blocksicherung – Beispiel

- Sei $G(x) = x^3 + x + 1$
- Es sollen die Bits 1001 1001 01 übertragen werden, damit $I(x) = x^9 + x^6 + x^5 + x^2 + 1$
  und $I(x) \cdot x^k = x^{12} + x^9 + x^8 + x^5 + x^3$
- Wir teilen

  $I(x) \cdot x^k : G(x)$

  und erhalten
  - P(x) nicht interessant
  - R(x) der Rest, die eigentliche Prüfsumme

Zyklische Blocksicherung – Beispiel

$x^{12} + x^9 + x^8 + x^5 + x^3 : (x^3 + x + 1) = x^9 + x^7 + x^4 + x + 1$
$x^{12} + x^{10} + x^9$
=========
$x^{10} + x^8 + x^5$
$x^{10} + x^8 + x^7$
=========
$x^7 + x^5 + x^3$
$x^7 + x^5 + x^4$
========
$x^4 + x^3$
$x^4 + x^2 + x$
=======
$x^3 + x^2 + x$
$x^3 + x + 1$
=========
$x^2 + 1$ ← Rest

Zyklische Blocksicherung

- Also:
  - $I(x) = x^9 + x^6 + x^5 + x^2 + 1$, $R(x) = x^2 + 1$
  - Damit $C(x) = I(x) \cdot x^k + R(x)$
    $= x^{12} + x^9 + x^8 + x^5 + x^3 + x^2 + 1$
- Der Sender überträgt C(x)
- Der Empfänger teilt C(x) durch G(x) und erhält den Rest 0 $\Rightarrow$ kein Fehler

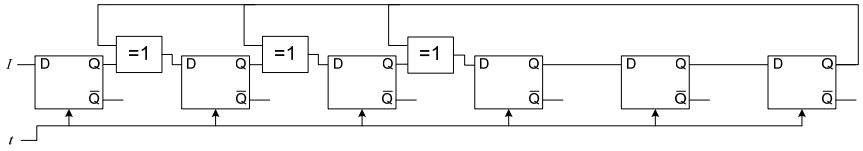
Realisierung von CRC in Hardware:

- Abziehen von Polynomen $\rightarrow$ bitweises XOR der Koeffizienten
- Der gesamte Divisionsvorgang kann durch Schieberegister dargestellt werden.

Zyklische Blocksicherung

- Die folgende Schaltung erzeugt eine Blocksicherung gemäß

$$G(x) = x^5 + x^2 + x + 1$$



Zuverlässige Übertragung

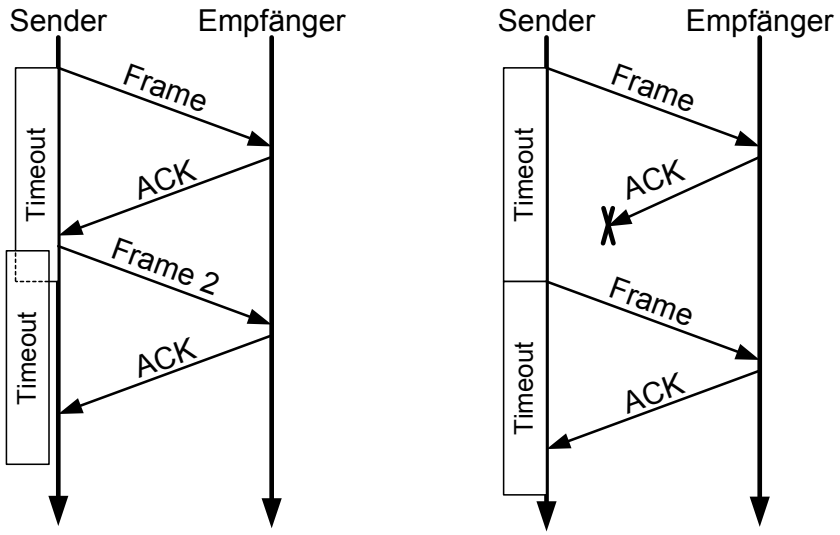
- Der Empfänger ist jetzt in der Lage, (mit großer Wahrscheinlichkeit) Fehler bei der Übertragung zu erkennen.
- Fehlerhaft übertragene Frames werden nicht berücksichtigt und verworfen – die entsprechenden Daten müssen daher erneut übertragen werden.
- Zwei Grundmechanismen:
  - Bestätigungen (ACKs, Acknowledgements)
  - Timeout
- Bestätigungen können an Nutzlast-Frames angehängt werden, die in Gegenrichtung transportiert werden (*Huckepackverfahren*, *Piggyback*).

ARQ

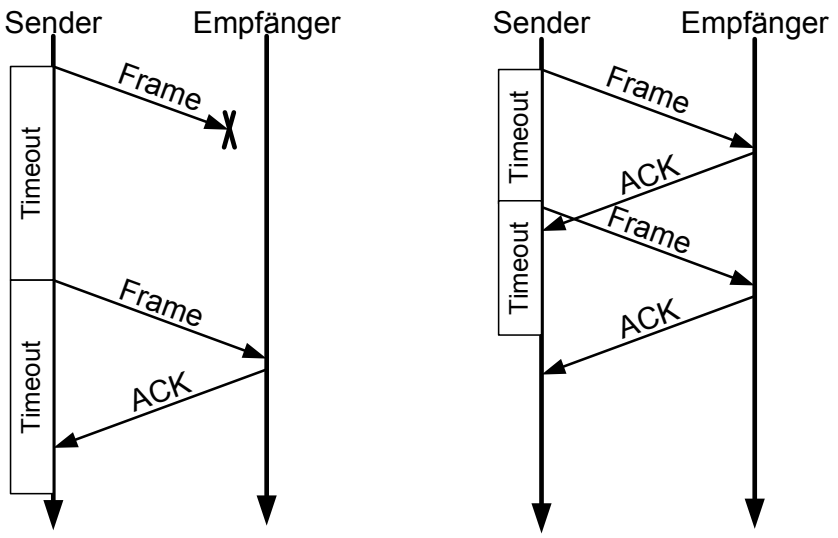
Prinzipielles Vorgehen: *Automatic Repeat Request* (*ARQ*)

- Der Empfänger sendet ein ACK, wenn ein Frame erfolgreich empfangen wurde.
- Erhält der Sender das ACK, so gilt die Zustellung als erfolgreich.
- Erhält der Sender innerhalb einer bestimmten Zeit kein ACK (Timeout), so sendet er den Frame erneut.
- Einfachster Mechanismus: *Stop-and-Wait*: jeder Frame wird separat bestätigt. Erst nach der Bestätigung wird ein neuer Frame gesendet.

ARQ



ARQ



ARQ

- Hat der Empfänger einen Frame erfolgreich empfangen und geht das ACK verloren, so sendet der Sender denselben Frame erneut.
- Der Empfänger könnte diesen Frame als neuen Frame ansehen und die Kopie fälschlicherweise noch einmal weiterverarbeiten.
- Lösung: der Sender stattet jeden Frame mit einer Sequenznummer aus (bei Stop-and-Wait reicht 1-Bit).
- Genereller Nachteil von Stop-and-Wait: bei Medien mit langer Roundtrip-Zeit (RTT), wird es kaum ausgelastet
  - Satellitenverbindungen
  - Interkontinental-Kabel

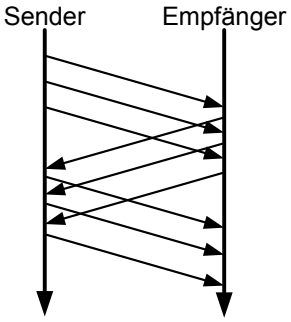
Sliding Window

Sliding Window

- Man erlaubt es dem Sender, mehrere Frames *unbestätigt* auszusenden.
- Jeder Frame hat eine Sequenznummer
- Der Sender verwaltet drei Größen
  - letzter bestätigter Frame (LAR)
  - letzter gesendeter Frame (LFS)
  - Größe des Sendefenster (SWS)

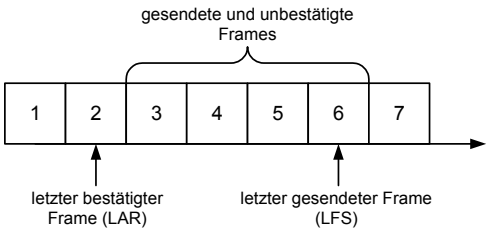
  Es gilt:

  LFS – LAR ≤ SWS



Sliding Window

- Sender:



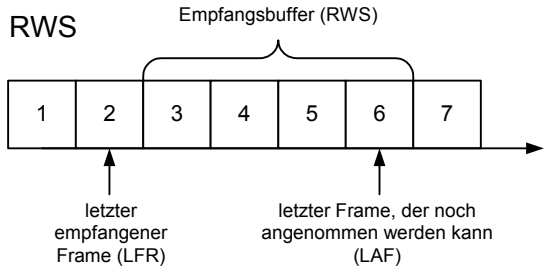
- Kommt ein ACK beim Sender an:
  - wird LAR angepasst
  - sendet er ggfs. einen weiteren Frame und passt LFS an
- Nach jedem Senden wird der Timer gestartet und nach Ablauf der Frame erneut gesendet.
- Der Sender muss bis zu SWS Frames speichern können.

Sliding Window

- Der Empfänger verwaltet drei Größen
  - letzter empfangener Frame (LFR) einer *ununterbrochenen Folge* empfangener Frames
  - größte Framenummer, die noch akzeptiert werden kann (LAF)
  - Größe des Empfangsfensters (RWS)

  Es gilt:

  LAF = LFR + RWS

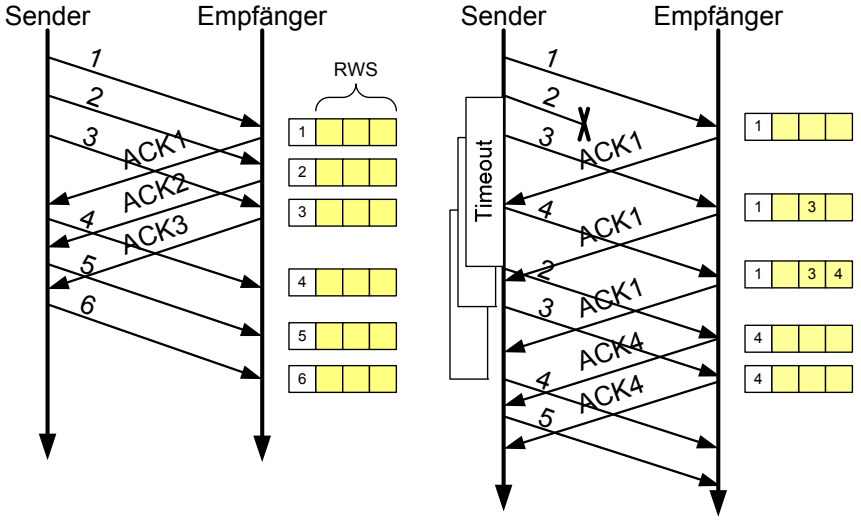


Sliding Window

- Kommt ein Frame beim Empfänger an:
  - Liegt die Sequenznummer außerhalb von [LFR+1, LAF] wird der Frame verworfen (entweder doppelt empfangen, oder außerhalb des Empfangspuffers)
  - Sonst wird der Frame bestätigt, ggfs. LFR und LAF angepasst
- Möglichkeiten der Bestätigungen
  - Kumulative ACKs: es werden nur *ununterbrochene Folgen* erfolgreich empfangener Frames bestätigt, z.B.
    1, 2, 3, 4, 6, 7, 8 → ACK(4)
  - Selektive ACKs: genau die empfangenen Frames werden bestätigt, z.B.
    1, 2, 3, 4, 6, 7, 8 → ACK(1, 2, 3, 4, 6, 7, 8)
  - Negative ACKs (NAKs): das Fehlen wird explizit quittiert, z.B.
    1, 2, 3, 4, 6, 7, 8 → NAK(5)

Sliding Window

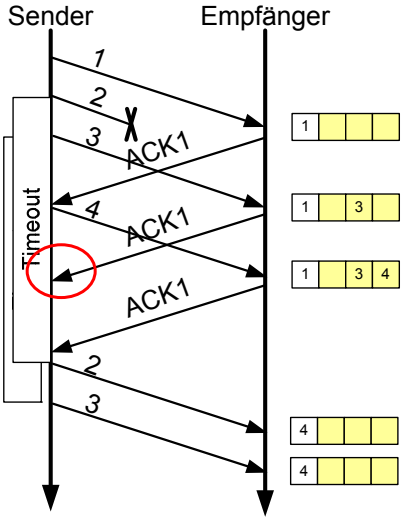
- Beispiel SWS=3, RWS=3, kumulative ACKs



Sliding Window

Duplicate ACKs:

- Der Sender könnte nach dem zweiten ACK1 sofort schließen, dass Frame 2 verloren ging – solche ACKs werden *Duplicate ACKs* genannt.
- Man müsste daher nicht auf den Timeout warten und der Empfänger würde Frame 2 etwas früher empfangen.



Duplicate ACKs

- Voraussetzung für die Auswertung von Duplicate ACKs: das Direktverbindungsnetzwerk erhält die Reihenfolge der Frames – dies trifft in der Regel zu.
- Problem, wenn der Sender bei *jedem* Duplicate ACK den nächsten Rahmen noch einmal sendet:
  - Bei großen Sendefenstern werden fehlende Rahmen sehr häufig nachgesendet.
  - Das belastet das Medium unnötig.
- Deshalb muss man die Nachsenderate limitieren beispielsweise auf 1 Mal/Timeout.
  - Eine entsprechende Regelung erfordert zusätzliche Verwaltungsinformationen im Sender und erhöht die Komplexität der Sender-Hardware.

Rechnernetze

WS 2012/13

Kapitel 5

Direktverbindungsnetzwerke

– Medienzugriff

Kapitel 5: Direktverbindungsnetzwerke – Medienzugriff

Ziel: Darstellung von Verfahren zum Medienzugriff an ausgewählten Beispielen:

- Ethernet
- Wireless LAN
- Master/Slave
- Token Ring
- Dedizierte Medien

Medienzugriff

Problem des Medienzugriffs (*Media Access Control*, *MAC*):

- Mehrere gleichzeitige Sendungen auf einem Medium (genannt *Kollisionen*) machen die Übertragungen unbrauchbar.
- Zwei Alternativen
  - Man lässt Kollisionen zu, integriert aber Mechanismen, diese zu *entdecken* und zu behandeln.
    Ein entsprechendes Verfahren wurde für das Ethernet entwickelt: *CSMA/CD* (*Carrier Sense Multiple Access with Collision Detection*).
  - Man versucht Kollisionen zu *verhindern.*
    Ein entsprechendes Verfahren wurde für das Wireless LAN entwickelt: *CSMA/CA* (*Carrier Sense Multiple Access with Collision Avoidance*).

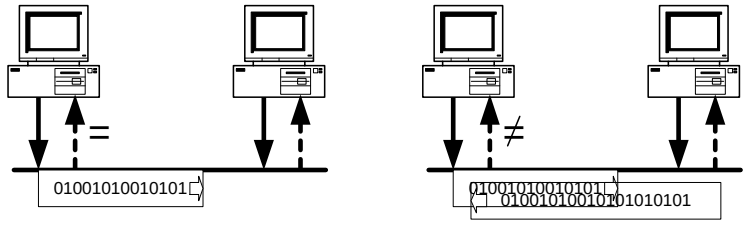
Ethernet

Ethernet:

- Eine der erfolgreichsten LAN-Technologien der letzten 20 Jahre.
- Entwickelt Mitte der 70er Jahre von Xerox Palo Alto Research Center (PARC).
- Die Grundidee geht auf das Paket-vermittelte Funknetz ALOHA zurück.
- Ethernet ist ein Mehrfachzugriffsnetz. Ein Ethernet ist ursprünglich ein *Bus* – erst moderne Varianten (z.B. Gigabit-Ethernet) wurden als Punkt-zu-Punkt-Konfigurationen im Vollduplex-Betrieb ausgelegt.
- Das Ethernet wird im Standard IEEE 802.3 beschrieben.

Ethernet

- Jede Ethernet-Station verfügt über eine Einrichtung, die gesendeten Signale direkt wieder vom Medium zu lesen.



- Sind die Signale verfälscht worden, liegt eine Kollision vor. Wichtig hier: die Sender erkennen die Kollisionen.
- Da sich Signale "nur" mit Lichtgeschwindigkeit bewegen, dürfen Stationen nicht zu weit entfernt sein (2,5 km) – sonst könnten Kollisionen unerkannt bleiben.

Ethernet

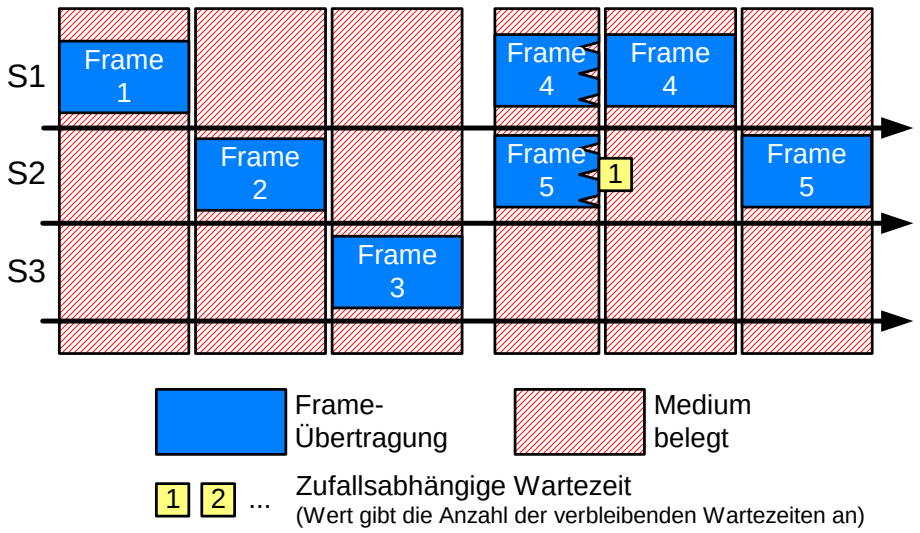
Vorgehen beim Medienzugriff

- Ist das Medium frei, wird sofort gesendet.
- Ist das Medium nicht frei, wird gewartet, bis es frei ist, und dann sofort gesendet.
- Wird während des Sendens eine Kollision erkannt, wird das Senden abgebrochen und ein Stausignal gesendet.
- Nach einer Kollision wird eine Zufallszahl z aus {0,1} ausgewählt und $z \cdot t_{wait}$ gewartet (t_{wait}=51,2 µs).
- *Exponential Backoff*: bei wiederholten Kollisionen, wird die Basismenge für die Zufallszahlen verdoppelt – bei n Kollisionen wird die Zufallszahl aus {0,... 2^n-1} genommen.

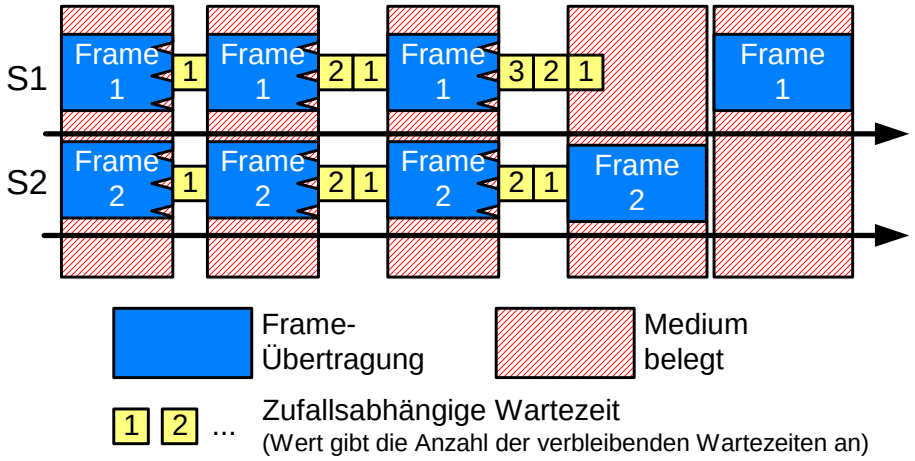
Ethernet – einige Details

- Die Daten werden Manchester-kodiert.
- Das Vorhandensein einer Übertragung wird über das Trägersignal erkannt (*Carrier Sense*).
- Jeder Sender sendet bei freiem Medium (kein Carrier entdeckt) zunächst eine Präambel (64 Bit).
- Wird während der Präambel eine Kollision erkannt, werden die 64 Bit Präambel zuerst beendet, bevor ein Stausignal (32 Bit) gesendet wird.
- Wird nach der Präambel keine Kollision erkannt, beginnt der Sender mit der Übertragung des Frames, dieser hat mindestens 512 Bit. Der Sender achtet dabei weiter auf Kollisionen.
- Grund für die Mindestgröße: auch entfernte Stationen sollen Kollisionen erkennen können (Übungsaufgabe).

CSMA/CD



CSMA/CD



WLAN – Medienzugriff

Warum ist die Kollisions*erkennung* bei Funknetzen ein Problem?

- Eine Station müsste gleichzeitig senden und empfangen können ⇒ verursacht Hardwarekosten.
- Sendet sie einen Frame aus, müsste sie das Funkmedium abhören und erkennen, ob gleichzeitig eine andere Übertragung abläuft.
- Das Problem: am Ort des Senders ist die Signalstärke des eigenen Signals so stark, dass alle anderen Übertragungen überdeckt werden.

⇒ Kollisions*vermeidung* statt -erkennung

WLAN – Medienzugriff

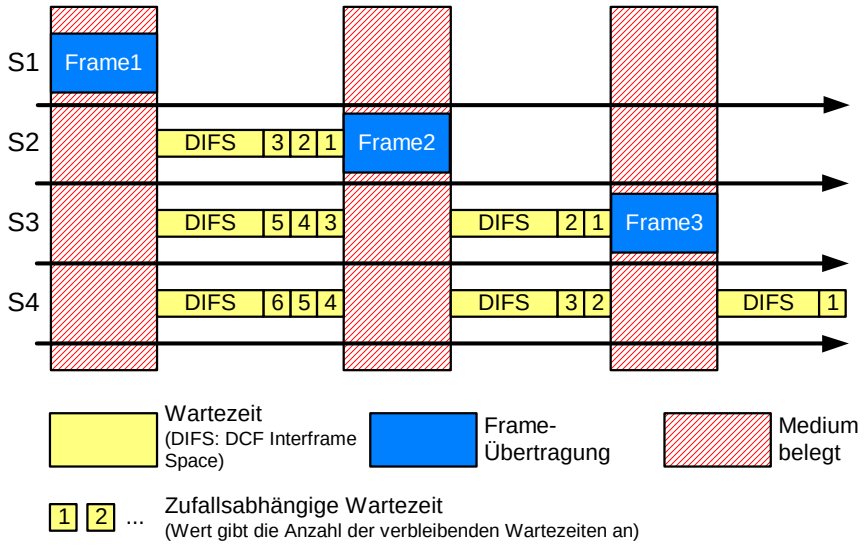
CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*)

- Wird in IEEE 802.11 beschrieben.
- *Distributed Coordination Function* (*DCF*):
  es gibt keine zentrale Instanz, die den Zugriff regelt
  (auch im so genannten *Ad-hoc-Modus* d.h. ohne Access Point möglich).
- *Point Coordination Function* (*PCF*, hier nicht dargestellt):
  der Access Point regelt den Zugriff auf das Medium
  (nur im Infrastruktur-Modus mit Access Point möglich).

WLAN – Einfaches CSMA/CA

- Ein Sender hört erst das Medium ab;
  ist es frei $\Rightarrow$ sofort senden
- Ist es nicht frei:
  - warte auf das Ende der Übertragung
  - zusätzlich wird am Ende eine weitere Zeit abgewartet
  - immer noch frei: senden
- Wird das Medium während der Wartezeit von einem anderen Sender belegt $\Rightarrow$ *Backoff*
  - Ende der Übertragung und zusätzliche Zeit warten (mit einer anderen Wartezeit)
  - Backoff wird solange wiederholt, bis Senden möglich ist
- Wartezeiten bilden sich aus *Interframe Spaces* und einer zufallsabhängigen Wartezeit (0..n).

WLAN – Einfaches CSMA/CA



WLAN – Einfaches CSMA/CA

- Nach jeder Belegung reduziert sich die neue Wartezeit.
- Das Verfahren vermeidet Kollisionen nicht garantiert:
  - z.B. zwei Sender starten gleichzeitig mit derselben Zufallszahl

  ⇒ hier müssen überlagerte Verfahren eingesetzt werden
- Die maximale Wartezeit nach DIFS wird *Contention Window* genannt.
- Die Wahl des Contention Windows hat Einfluss auf die Kollisionshäufigkeit:
  - Kleine CW: Wartezeit im Mittel klein, aber Kollisionen häufig
  - Große CW: Wartezeit im Mittel groß, aber Kollisionen selten

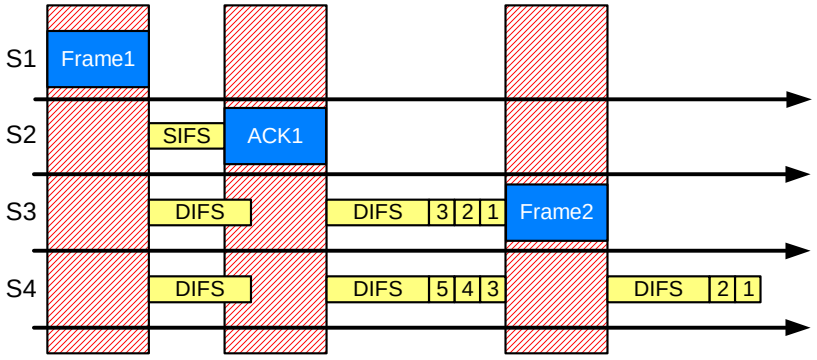
WLAN – Einfaches CSMA/CA

Exponential Backoff:

- Mögliche Werte für CW sind 31, 63, 127, 255, 511, 1023
- Man beginnt mit CW=31
- Bei einem Fehler, der auf eine Kollision hindeutet (z.B. fehlendes ACK), wird der nächsthöhere Wert genommen (bis zum Maximum).
- Bei Belastungen des Mediums durch viele sendewillige Stationen reagieren die Sender damit adaptiv.
  - Der Gesamtdurchsatz wird zwar reduziert
  - aber die Kollisionswahrscheinlichkeit nimmt ab
- Der initiale CW Wert wird wieder verwendet, wenn die Kollisionen nicht mehr auftreten.

WLAN – CSMA/CA mit Bestätigung

- Bisher: unbestätigte Frames (sinnvoll bei Broadcasts)
- Möglich: Bestätigung mit ACKs
- Damit ACKs priorisiert behandelt werden, wird eine kurze Wartezeit *SIFS* (*Short Interframe Space*) benutzt.



Master/Slave

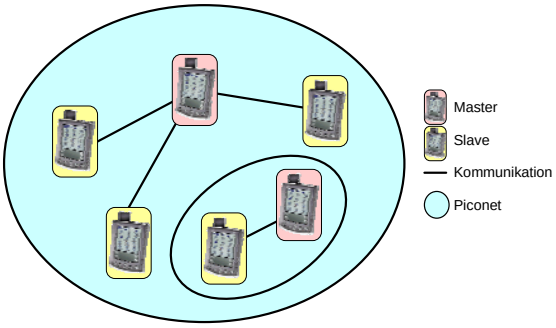
Medienzugriff mit Master/Slave:

- Ein vorher ausgewählter Teilnehmer (*Master*) erteilt wechselweise die Senderechte an alle weiteren Teilnehmer (*Slaves*).
- Wird z.B. bei Bluetooth verwendet.
- Vorteil: das Verfahren ist einfach zu realisieren.
- Nachteil: Auswahl eines Masters notwendig.

Master/Slave – Bluetooth

Beispiel Bluetooth:

- Master mit verbundenen Slaves wird *Piconet* genannt
- Mehrere Piconets an einem Ort möglich

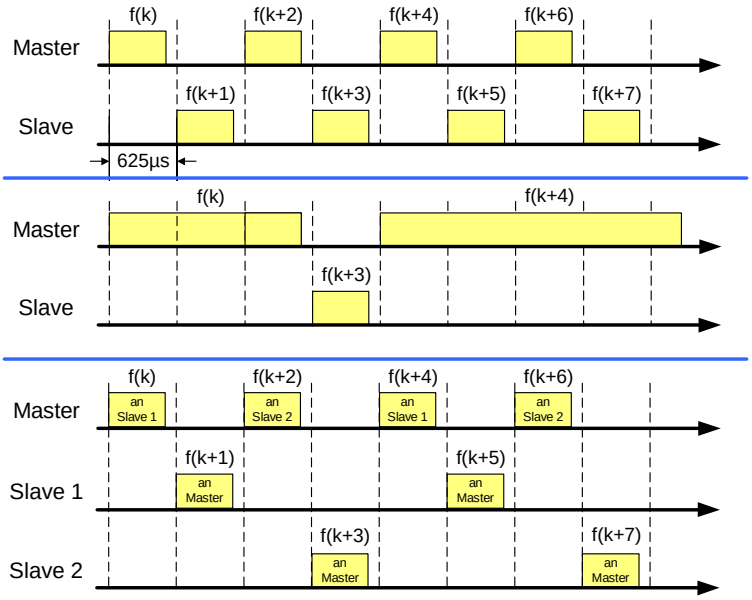


- Auch möglich: ein Rechner gehört zu mehreren Piconets

Master/Slave – Bluetooth

- Problem: Rechner im Piconets *und* Rechner verschiedener Piconet dürfen sich nicht stören.
- Lösung:
  - *im* Piconet: Zeitmultiplex
  - *zwischen* Piconets: Raummultiplex und Frequenzmultiplex durch Frequency Hopping
- *Fast Frequency Hopping*: alle 1/1600 s wird die Frequenz gewechselt.
- Der Wechsel erfolgt über eine Pseudo-Zufallsfolge, die allen Kommunikationspartnern im Piconet bekannt ist.

Master/Slave – Bluetooth

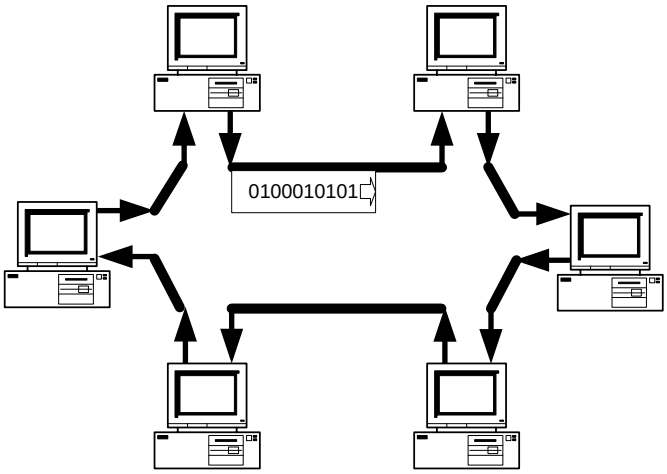


Token Ring

Token Ring

- Das gemeinsame Medium ist ein Ring – damit unterscheidet sich der Ansatz signifikant vom Ethernet oder WLAN.
- Token Ring wurde durch IBM eingeführt und im Standard IEEE 802.5 spezifiziert.
- Grundidee:
  - Alle Rechner eines Netzwerks sind zu einem Ring verbunden.
  - Daten fließen in einer festgelegten Richtung im Ring.
  - Die Sendeberechtigung wird durch ein *Token* repräsentiert, das im Ring kreist.

Token Ring

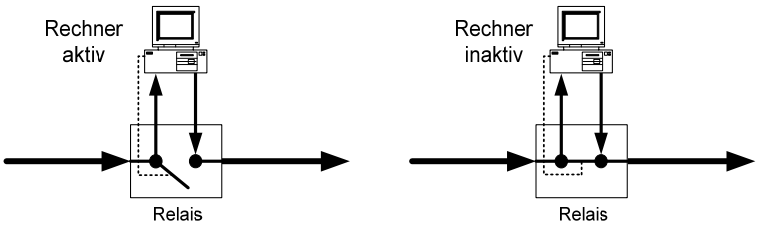


Token Ring

- Will eine Station nichts senden, so gibt sie das empfangene Token an den nächsten Nachbarn weiter.
- Will eine Station einen Frame versenden
  - wartet sie auf das Token und nimmt es vom Ring
  - sendet statt dessen einen Nutzlast-Frame
- Eine Station, die einen Frame empfängt
  - gibt diesen an den nächsten Nachbarn weiter
  - ist sie der Empfänger, kopiert sie den Inhalt
- Bekommt der ursprüngliche Sender seinen Frame zurück, verwandelt er ihn wieder in ein Token.
- Daraufhin könnte der nächste Nachbar senden.

Token Ring

- Da jede Station aktiv das Token oder einen Frame weiterleiten muss, stellen ausgeschaltete Rechner ein Problem dar.
- Lösung: ein Relais
  - Strom: der Schalter ist offen
  - stromlos: der Schalter überbrückt die Netzwerkkarte



Token Ring

Medienzugriff im Detail:

- Der Ring muss genügend "Speicherkapazität" haben, um ein Token (24 Bit) speichern zu können.
- Üblicherweise nimmt jede Station nur 1 Bit auf, bevor sie es weitergibt – gibt es zu wenig Stationen im Netz, reicht die Kapazität daher nicht aus.
- Eine so genannte *Monitorstation* speichert ggfs. zusätzliche Bits des Tokens.
- Das Ergreifen eines Tokens bedeutet, ein bestimmtes Bit zu modifizieren, danach werden Nutzlastdaten angehängt.

Token Ring – weitere Details

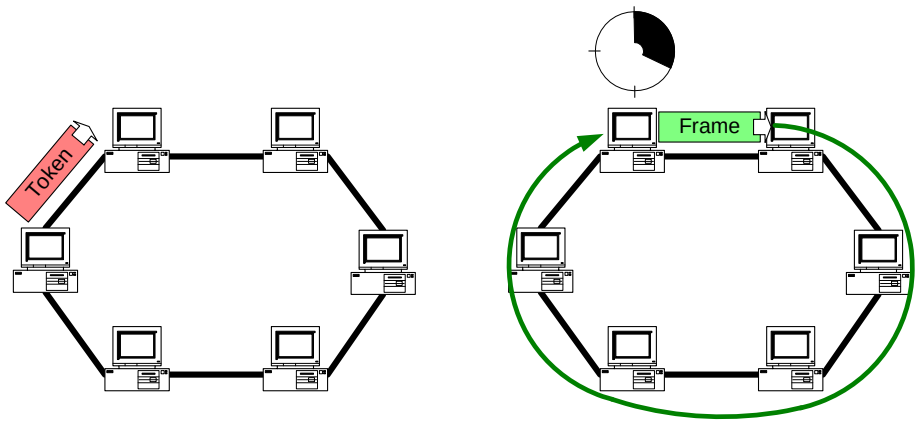
- Der Sender ist dafür verantwortlich, den gesendeten Frame vom Ring zu nehmen.
- Eine Station, die ihrer Adresse im Frame wiederfindet, setzt ein ACK-Bit (nimmt aber keine Nutzdatenbits vom Ring). Der Sender kann am gesetzten Bit erkennen, dass der Empfang erfolgt ist.
- Jeder Frame enthält eine Zieladresse. Sie kann auch eine Multicast-Adresse sein.

Token Ring – Token-Haltezeit

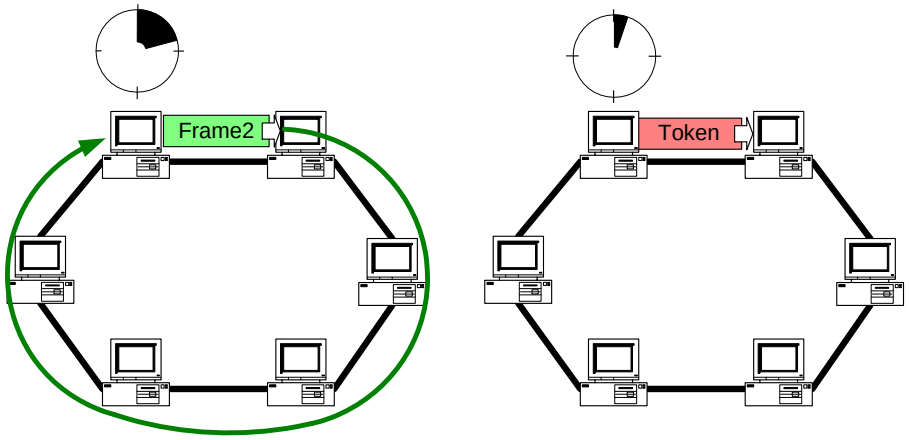
Token-Haltezeit

- Hat eine sendewillige Station das Token, so kann sie es für eine bestimmte Zeit (aufeinanderfolgender Sendevorgänge) behalten (*Token-Holding-Time*, *THT*).
  - Eine Station transformiert einen Nutzlast-Rahen in einen neuen und gibt kein Token an den Nachbarn weiter.
- Durch die Wahl einer geeigneten THT
  - kann der Ring besser ausgelastet werden,
  - hungert eine ständig sendewillige Station nicht das Netz aus
- Bei 802.5 beträgt die THT 10 ms.

Token Ring – Token-Haltezeit



Token Ring – Token-Haltezeit



Token Ring – Token-Haltezeit

- Ein Sender muss vorher berechnen, ob die Zeit für einen weiteren Umlauf noch ausreicht.
- Tokenumlaufzeit (*Token Rotation Time*, *TRT*): die maximale Zeit, die es dauert, bis ein Token einmal den Ring umrundet hat (inkl. aller Nutzlastübertragungen).
- Es gilt:

$$TRT \leq AktiveKnoten \cdot THT + RingLatenz$$

RingLatenz ist die Ausbreitungsverzögerung des Rings

Token Ring – Monitorstationen

Monitorstationen

- Jede Station im Ring muss prinzipiell in der Lage sein, eine Monitorstation zu werden; im fehlerfrei laufenden Ring gibt es genau eine Monitorstation.
- Monitorstationen sind notwendig, um die Ringumlaufzeit künstlich zu erhöhen, damit ein Token gespeichert werden kann.
- Sie überwachen zusätzlich den Ring:
  - Ist noch ein Token vorhanden?
  - Gibt es ewig kreisende Frames (durch Ausfall des Senders)?

Token Ring – Monitorstationen

- Erkennung verwaister Frames:
  - Jeder Nutzdatenframe darf die Monitorstation nur einmal passieren.
  - Die Monitorstation setzt bei jedem Frame das Monitor-Bit.
  - Erhält sie einen Frame mit gesetztem Monitorbit, so ist die sendende Station ausgefallen.
- Tokenausfall:
  - Eine Monitorstation startet nach Erhalt eines Tokens einen Timer und wartet die maximal mögliche Tokenumlaufzeit ab (Formel siehe vorne).
  - Ist bis dahin kein Token eingetroffen, wird es neu erzeugt.

Token Ring – Monitorstationen

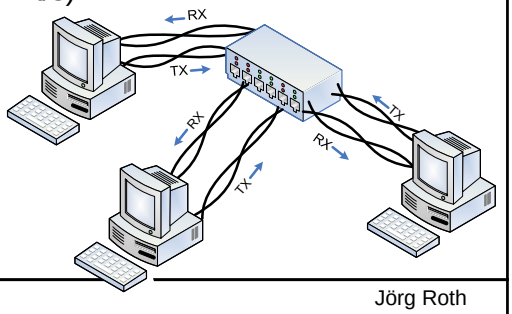
Ausfall von Monitorstationen:

- Eine Monitorstation zeigt ständig durch spezielle Steuernachrichten an, dass sie noch "lebt".
- Entdeckt eine Station das Ausbleiben solch einer Nachricht, beansprucht sie selbst den Status einer Monitorstation über einen *Claim-Frame.*
- Empfängt sie selbst einen Claim-Frame, so wird z.B. über die Adresse eine Auswahl getroffen. Ggfs. wird der Claim-Frame dann nicht weitergeleitet.
- Erhält eine Station ihren Claim-Frame zurück, so wird sie die Monitorstation.

Dedizierte Medien

Trend für schnellere Übertragung: *dedizierte Medien*

- Möglich bei kabelbasierter Übertragung, bei Funk schwierig.
- Beispiel Ethernet:
  - altes Ethernet: Koaxkabel, alle Sender hängen an einem Kabel (Bus)
  - ab *Fast Ethernet* (100 MBit/s): mehrere Twisted Pair, Stationen werden sternförmig verbunden



Dedizierte Medien – Ethernet

Zwei Möglichkeiten der Benutzung:

- *Halbduplex*: alle TX/RX werden zusammengeschlossen (eigentlich ein Bus), d.h. keine zwei Stationen dürfen gleichzeitig senden
  → es wird CSMA/CD für den Medienzugriff verwendet
- *Vollduplex*: jede Station hat ihr eigenes Medium, Station in der Mitte ist ein *Switch* (siehe später)
  → alle Stationen dürfen gleichzeitig senden

RJ45-PIN	Bedeutung
1	TX+
2	TX-
3	RX+
4	
5	
6	RX-
7	
8	

Dedizierte Medien – Ethernet

- In der Regel werden schnelle Ethernets im Vollduplex-Modus betrieben – damit gibt es kein Medienzugriffs-problem mehr.
- Es entsteht aber ein neues Problem: Überlastung des Switches:
  - Beispiel: zwei Stationen schicken gleichzeitig mit höchster Geschwindigkeit Rahmen über einen Switch an eine dritte Station, der Switch empfängt dann doppelt so schnell Rahmen, wie er sie senden kann.
  - Lösung *Flusskontrolle*: ein Switch kann vor einer Überlastung allen angeschlossenen Stationen durch einen "Pause"-Rahmen signalisieren, dass er keine weiteren Rahmen mehr empfangen kann.

Rechnernetze

WS 2012/13

Kapitel 6

Kommunikation zwischen nicht-direkt verbundenen Rechnern

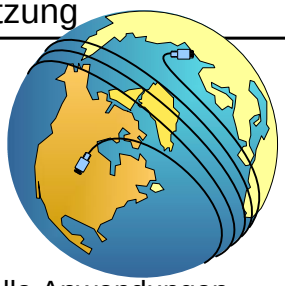
– Grundlagen

Kapitel 6: Kommunikation zwischen nicht-direkt verbundenen Rechnern – Packet und Circuit Switching

Ziel: Vermitteln der Grundlagen der Vernetzung nicht direkt verbundener Rechner

- Geräte zur Verbindung von Netzen
- Circuit Switching vs. Packet Switching
- Circuit Switching am Beispiel von ATM

Die nächste Stufe der Vernetzung

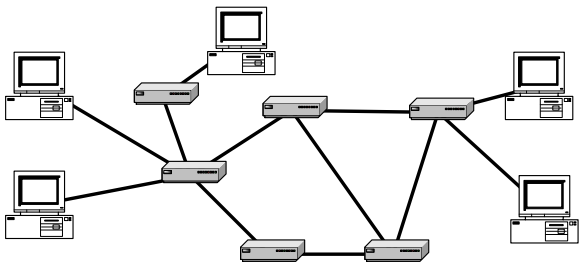


- Bisher wurden nur direkt verbundene Rechner betrachtet.
- Gründe dafür, dass man nicht mit einem *einzigen* Medium oder einer Vernetzungstechnologie auskommt:
  - Es gibt nicht *das eine Medium*, das für alle Anwendungen geeignet ist. Beispielsweise haben mobile Benutzer signifikant andere Anforderungen an ein Kommunikations-medium als der Betreiber eines Online-Warenhauses.
  - Der Reichweite sind bei einem Medium technologische Grenzen gesetzt, z.B. durch Signaldämpfung, Laufzeiten.
  - Je höher die Zahl der Netzwerkteilnehmer an einem Medium desto niedriger der Durchsatz pro Teilnehmer.

  $\Rightarrow$ Größere Netzwerke sind aus vielen Medien aufgebaut

Die nächste Stufe der Vernetzung

- In der Regel sind daher kommunizierende Rechner nicht direkt verbunden, sondern benutzen eine Reihe von Zwischenstationen, die paarweise direkt verbunden sind.



- Als Zwischenstationen werden in der Regel dedizierte Geräte (z.B. Router, Switches verwendet).

Hardware der OSI-Schichten 1-3

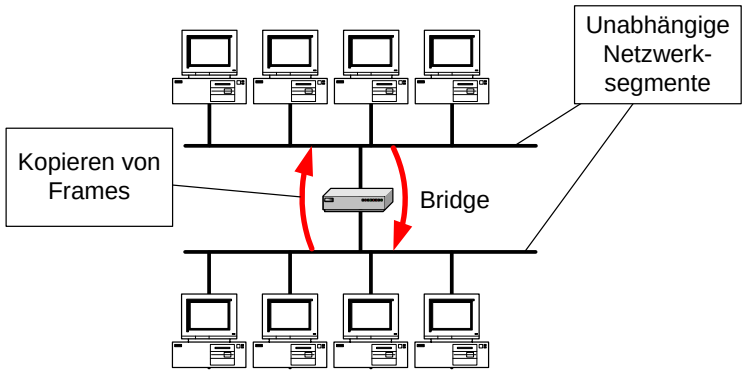
Schicht 1:

- *Repeater*: ein Verstärker für digitale Signale, wertet Frames nicht aus
- *Hub*:
  - passiv: ein Kabel-Verteiler
  - aktiv: ein mehrwegiger Repeater

Schicht 2:

- *Bridge*:
  - verbindet unabhängige Netzwerksegmente
  - kopiert Frames aus einem Segment in das jeweils andere
  - Störungen, Kollisionen, fehlerhafte Frames bleiben auf ein Segment beschränkt

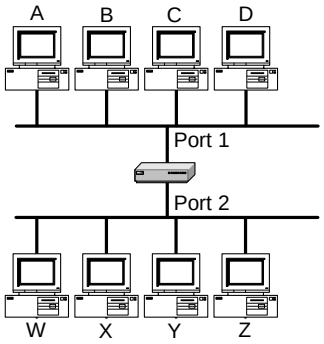
Bridges



- naiver Ansatz: Kopieren *aller* Frames in das jeweils andere Segment
- optimiert: Pflegen einer Adress-Tabelle durch den Administrator

Bridges

- selbstlernend: sendet ein Rechner aus einem Segment, kann sich die Bridge diese Netzwerkadresse in einer Tabelle merken
- Einträge haben ein Verfallsdatum, damit Rechner auch bewegt werden können
- Die Tabelle muss nicht vollständig gefüllt sein

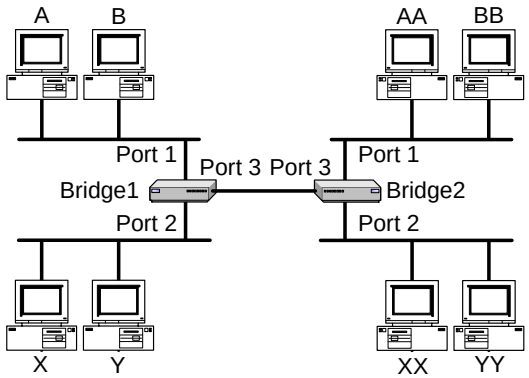


Host	Port
A	1
B	1
X	2
Y	2

Bridges

Bridges können auch mehr als 2 Ports haben; Beispiel:

- A sendet an YY, A war bisher unbekannt
- Bridge 1 sendet den Frame an Port 2 und 3 (*nicht* 1)
- Bridge 2 sendet den Frame an Port 1 und 2 (*nicht* 3)



Bridges

- Bridge 1 merkt sich, dass A über Port 1 erreichbar ist
- Bridge 2 merkt sich, dass A über Port 3 erreichbar ist
- Sendet YY nun eine Antwort, wird der direkte Weg gewählt
- Bemerkung: die Bridges ersetzen beim Weitereichen der Frames *nicht* die Absenderadresse
- Problem: wenn Bridges in Schleifen angeordnet sind, werden Frames u.U. unendlich oft weiterkopiert
  - Spanning-Tree-Algorithmen berechnen aufspannende Bäume – nur entlang dieser Bäume wird kopiert
  - Diese erfordern zusätzliche Kontrollnachrichten zwischen den Bridges

Switch

Weitere Verbindungsgeräte zwischen Netzwerken

- *Switch*:
  - Bridges werden auch *LAN-Switches* genannt (insb. mit mehr als 2 Ports)
  - Switches werden auch für die OSI-Schicht 3 verwendet, z.B. bei verbindungsorientierten Technologien wie ATM

5 port LAN switch

Schicht 3:

- *Router*:
  - vermittelt Pakete zwischen LANs
  - kann Netzwerke unterschiedlicher Technologien verbinden (z.B. Ethernet mit WLAN)
  - wichtiger Vertreter: IP-Router

Netzwerkgeräte auf verschiedenen Schichten

Oft ist es nicht klar, auf welcher Schicht ein Netzwerkgerät operiert.

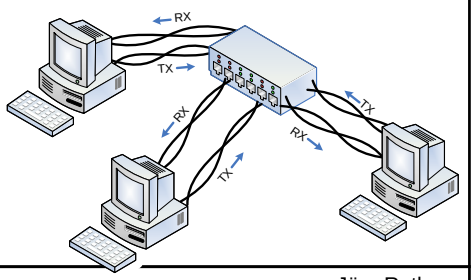
Beispiel: WLAN-Router mit NAT-Funktionalität

- Schicht 2: ein Switch für Ethernet und WLAN
- Schicht 3: ein IP-Router
- Schicht 4: NAT-Funktion (überwacht Transport-Kanäle und ordnet Nachrichten verschiedenen Endgeräte zu, genaue Erklärung siehe später)

Switching vs. Routing

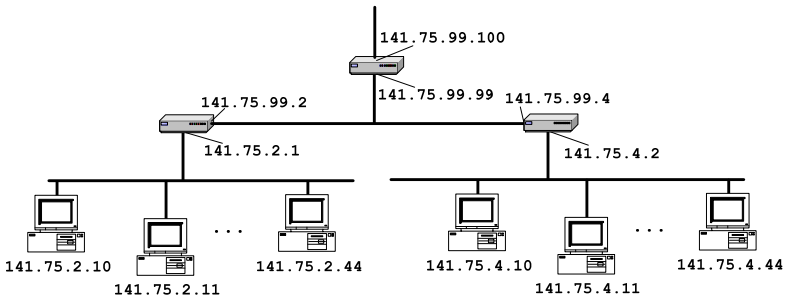
Routing vs. Switching:

- Früher: Layer-2-Vernetzungen waren häufig Bus-orientiert, d.h. die Anzahl der Teilnehmer, die sich gegenseitig stören können, war groß.
- Router waren daher eine gute Möglichkeit, die Layer-2-Netze gegenseitig voneinander abzuschotten.
- Heute: Dedizierte Medien sind über Switches verbunden.



Beispiel: Vernetzung eines Campus

- Mögliche Konfiguration:
  auf dem Campus werden Router verteilt z.B.
  - pro Subnetz einen,
  - sowie einen Ausgangsrouter für die weltweite Anbindung

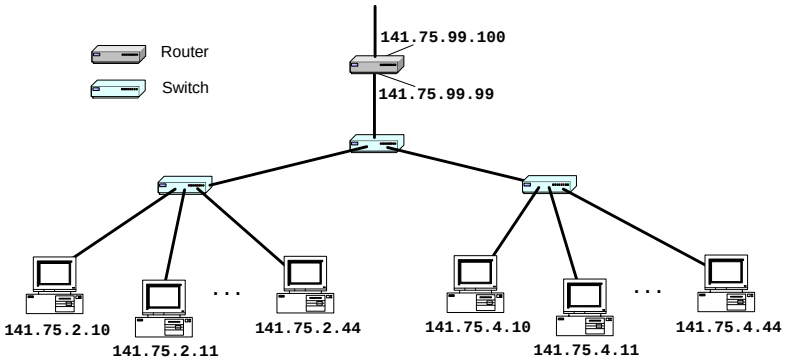


Vorteile:

- Die einzelnen Subnetze sind aus Sicht von OSI-Schicht 2 voneinander abgeschottet.
- Aber auch aus Sicht der höheren Schichten gibt es eine Abschottung. So können die Router Sicherheitsaufgaben übernehmen und nicht alle Pakete in ein Subnetz durchlassen.

Durch den Einsatz von modernen Switches ist aber auch eine alternative Architektur möglich.

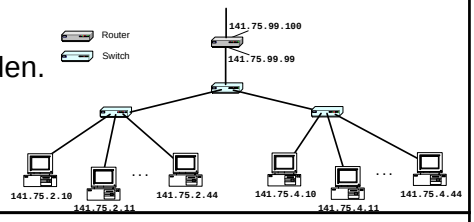
Switching



- Es gibt nur noch einen Ausgangsrouter
- Innerhalb des Campus gibt es ein "geswitches" Netzwerk

Switching

- Auf dem Campus haben wir ein Layer-2-Netzwerk, d.h. jeder Rechner ist logisch mit jedem anderen *direkt* verbunden.
- Dies ist effizient möglich, da wir pro Switch eine Sterntopologie haben:
  - Pro Verbindung gibt es nur zwei Teilnehmer (1 Rechner – 1 Switch).
  - Zwischen diesen wird im *Duplex-Modus* kommuniziert (d.h. wir haben pro Sender eine eigene Leitung).
  - Dadurch müssen keine Kollisionen behandelt werden.



Switching

Vorteile:

- Einfach zu administrieren
  - wir benötigen keine IP-Subnetze mehr zum Routing
  - aus organisatorischen Gründen (z.B. Nummernkreis-Verwaltung) dennoch sinnvoll
- Effiziente Punkt-zu-Punkt-Kommunikation

Nachteile

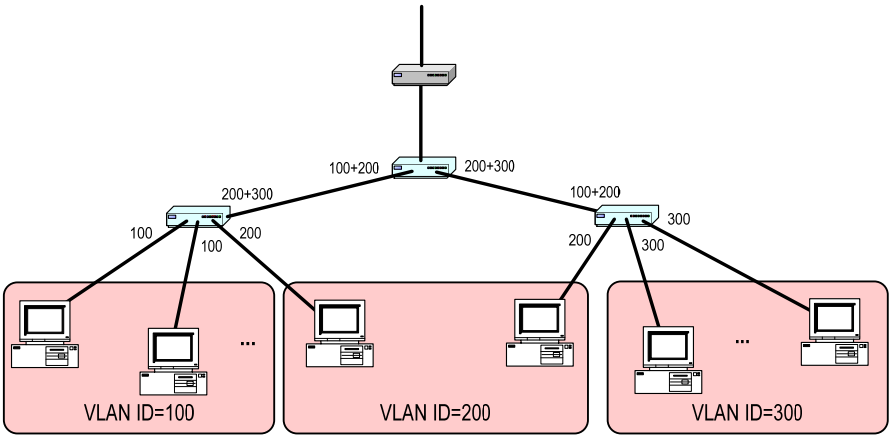
- Keine Abschottung der Subnetze gegeneinander
- Broadcasts betreffen sehr viele Teilnehmer
- Sicherheitsprobleme (jeder Rechner ist mit jedem direkt verbunden)

VLANs

Mögliche Lösung: *VLANs* (*virtuelle LANs*)

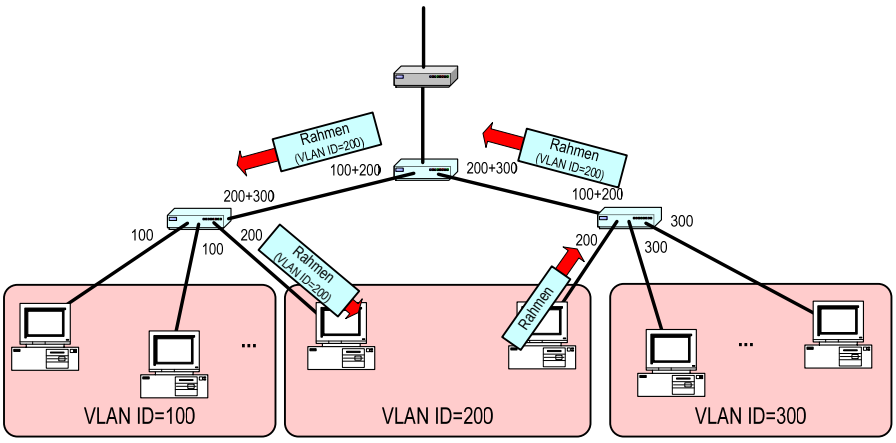
- Jedem Port eines Switches wird eine VLAN-ID zugeordnet.
- Bei Empfang von Rahmen, die über Ports mit VLAN-IDs kommen, werden die VLAN-IDs in die weitergeleiteten Rahmen integriert.
- Ein Switch leitet Rahmen nur an solche Ports weiter, die die entsprechende VLAN-ID besitzen.
- Heute typisch: Switches lernen, welche VLANs an welchem Port verbunden sind, selbständig.

VLANs



Kapitel 6, Folie 19 Jörg Roth

VLANs



Kapitel 6, Folie 20 Jörg Roth

VLANs

Durch VLANs hat man die Vorteile aus "beiden Welten"

- Effiziente Punkt-zu-Punkt-Kommunikation
- Effizientes Broadcasting
- Gegenseitige Abschottung von logischen Subnetzen

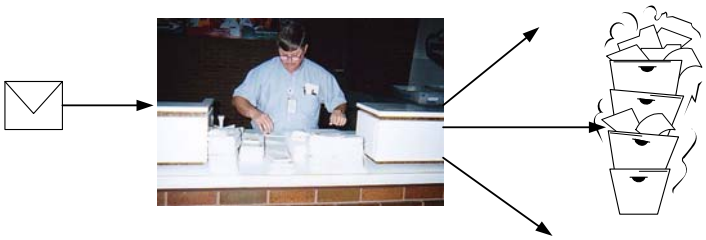
Circuit Switching vs. Packet Switching

Zurück zum eigentlichen Thema: Packet und Circuit Switching
Zwei unterschiedliche Ansätze

- *Packet Switching*:
  - jedes Paket wird separat über das Vermittlungsnetz zugestellt
  - die Folge von Paketen, die im Rahmen einer Sitzung zum selben Empfänger zugestellt werden, interessiert dabei nicht
- *Circuit Switching*:
  - vor der Datenübertragung wird eine Verbindung aufgebaut (*Signalisierung*)
  - das Vermittlungsnetz schaltet die Verbindung (mit dem zugehörigen Pfad) für die Dauer einer Sitzung

Circuit Switching vs. Packet Switching

- Packet Switching:



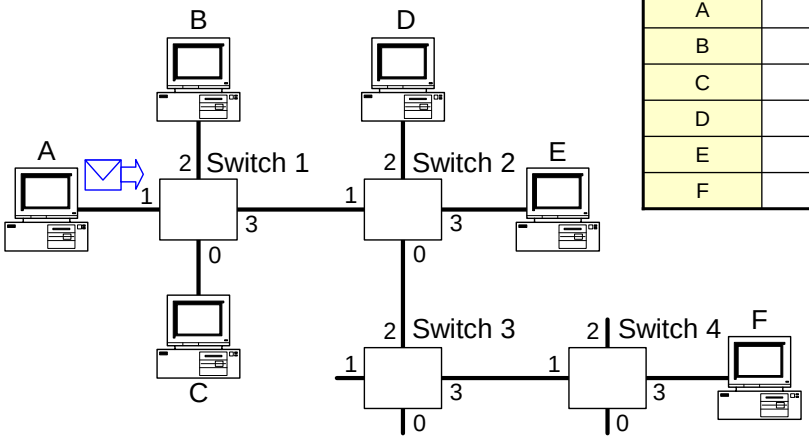
- Circuit Switching:

Circuit Switching vs. Packet Switching

- Packet Switching

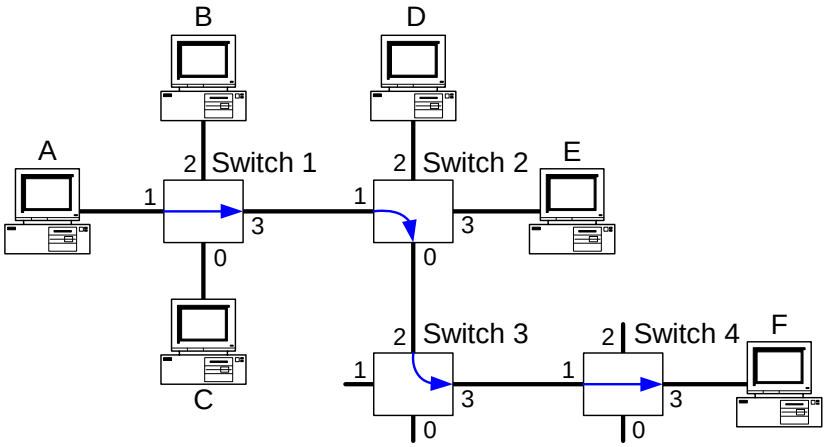
Weiterleitungstabelle für Switch 2

Ziel	Port
A	1
B	1
C	1
D	2
E	3
F	0



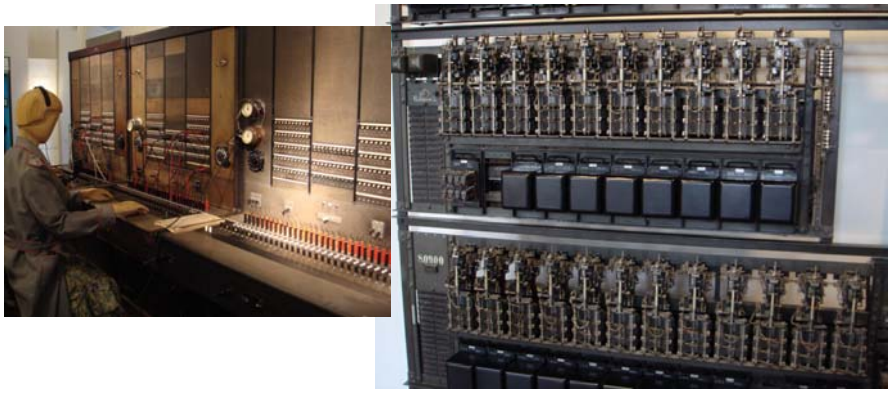
Circuit Switching vs. Packet Switching

- Circuit Switching



Circuit Switching

- Das Schalten "echter" elektrischer Verbindungen für die Dauer einer Sitzung (wie im alten Telefonnetz) ist bei Computernetzen unüblich.



Circuit Switching

- *Virtual Circuit Switching*:
  - Ein Rechner beantragt beim Vermittlungsnetzwerk durch Signalisieren eine *virtuelle Verbindung.*
  - Jeder Switch auf dem Weg zum Ziel richtet einen Teil der virtuellen Verbindung ein.
  - Jedes Paket, das vom Sender losgeschickt wird, besitzt einen *virtuellen Leitungsbezeichner*, an dem der Switch ablesen kann, über welchen Ausgangsport das Paket weitergeleitet wird.
  - Verbindungswege sind für die Sitzung fest – fällt ein Switch aus, muss eine neue virtuelle Verbindung geschaltet werden.

Circuit Switching

Vorteile von (Virtual) Circuit Switching:

- Durch das Konstrukt der Verbindung können Eigenschaften beim Verbindungsaufbau spezifiziert werden, die während der Verbindung erhalten bleiben (z.B. Quality of Service, siehe später).
- Die Reihenfolge von Paketen bleibt erhalten, da die Laufwege für eine Verbindung fest bleiben. Beim Packet Switching wird dagegen der Laufweg pro Paket ermittelt und kann sich bei einer Paketfolge ändern.

Nachteile:

- Rekonfigurierung einer Verbindung bei Ausfällen oder Engpässen problematisch.

ATM

ATM (*Asynchronous Transfer Mode*)

- Eine *verbindungsorientierte*, *Zellen-vermittelte* Technologie.
- Wurde Ende der 80er Jahre entwickelt. Im ersten Ansatz wurde ATM für die Telefonindustrie konzipiert, die eine schnelle, leitungsorientierte Datenübertragung benötigte.
- Eng verbunden mit ATM sind bestimmte Schicht-1-Übertragungsmedien (z.B. Glasfaser), typische Datenraten sind 155 oder 622 MBit/s.
- ATM unterstützt *Quality of Service*
  - Beim Verbindungsaufbau kann man bestimmte Anforderungen stellen.
  - Werden diese akzeptiert, so werden sie für die Dauer der Verbindung gewährleistet.

ATM-Zellen

ATM-Zellen

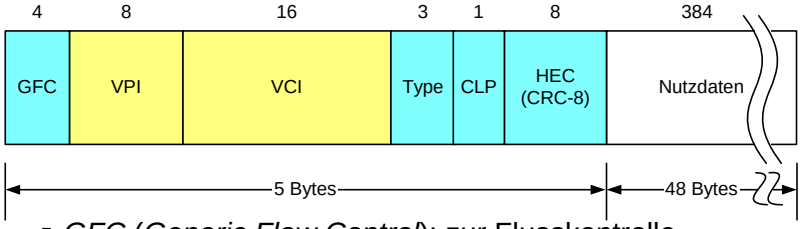
- ATM verwendet Pakete *fester* Länge (53 Byte: 5 Bytes Header, 48 Bytes Nutzdaten), sie werden *Zellen* genannt
- Vorteile von Zellen:
  - Implementierung von Switching-Aufgaben in Hardware ist einfacher.
  - Die Ausführungszeit von Switching-Aufgaben ist für jede Zelle gleich. Damit können Aufgaben einfacher parallelisiert werden.
  - Häufig müssen Switches Worst-Case-Abschätzungen darüber machen, wie lange das Versenden eines Paketes dauert – bei festen Größen sind diese Abschätzungen exakter.

ATM-Zellen

- Die optimale Größe von Zellen
  - Zu klein: der Overhead durch Zellen-Header steigt im Verhältnis zur Nutzlast
  - Zu groß: ggfs. müssen Bits zum Auffüllen benutzt werden, falls weniger als die Zellengröße übertragen werden soll
- Der "48-Byte-Kompromiss" für die Nutzdaten
  - US-Telefongesellschaften wollten 64 Byte
  - Die Europäer wollten 32 Byte
  - 48 lag genau in der Mitte

ATM-Zellen

- Das Zellen-Format



  - *GFC* (*Generic Flow Control*): zur Flusskontrolle
  - *VPI* (*Virtual Path Identifier*), *VCI* (*Virtual Circuit Identifier*): zusammen spezifizieren sie die virtuelle Leitung
  - *Type*: acht verschiedene Zellen-Typen
  - *CLP* (*Cell Loss Priority*): sollen Zellen bei Überlast bevorzugt verworfen werden?
  - *HEC* (*Header Error Check*): CRC-Prüfsumme des Headers

ATM-Zellen-Sicherung

- Warum wird nur der Header gesichert?
  - man geht von sehr niedrigen Bit-Fehlerraten aus (10^{-10})
  - Bit-Fehler im Kopf sind aus Sicht der Switches viel kritischer (wenn z.B. der Leitungsbezeichner verfälscht wurde)
  - Häufig kann die Nutzlast ungesichert übertragen werden, z.B. bei Audio oder Video
  - Soll Nutzlast gesichert werden, muss dies durch überlagerte Protokolle erfolgen
- Wie erkennt man Zellengrenzen?
  - man führt ständig die CRC-Berechnung auf den letzten vier Bytes durch – ergibt sich das fünfte Byte, so ist es wahrscheinlich die CRC-Summe
  - Stimmt dies für weitere Intervalle im Abstand von 53 Bytes, so ist (sehr wahrscheinlich) die Zellengrenze gefunden

Einrichten von Verbindungen

Zwei Möglichkeiten

- *Permanent Virtual Circuits*: eine virtuelle Verbindung wird durch den Netzwerkdienstleister permanent in die Konfiguration der Switches eingetragen
  - eine feste Mietleitung zur Datenübertragung
  - zwei entfernte Rechner können so dauerhaft miteinander verbunden werden
  - keine Signalisierung
- *Switched Virtual Circuit*
  - eine virtuelle Verbindung wird per Signalisierung angefordert
  - das Verbindungsnetzwerk konfiguriert sich bei Bedarf

ATM-QoS

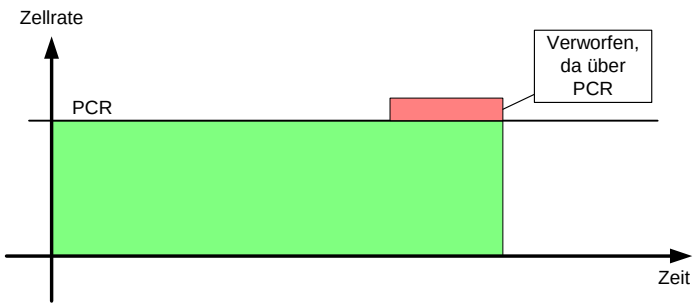
Dienstgüte, Quality of Service:

- Bei der Signalisierung können Anforderungen an die virtuelle Verbindung spezifiziert werden.
- Werden diese Anforderungen akzeptiert, werden diese für die Dauer der Verbindung gewährleistet.
- Die Switches müssen dazu entsprechende Kapazitäten reservieren.
- Steht die Kapazität nicht zur Verfügung, wird die Anforderung abgelehnt und keine Verbindung eingerichtet.

ATM-Dienstkategorien

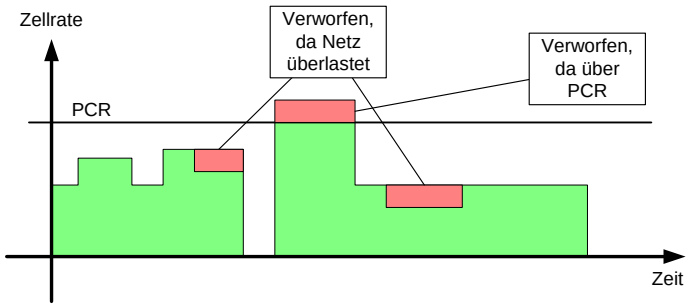
ATM-Dienstkategorien

- *CBR* (*Constant Bit Rate*): die Anwendung benötigt eine konstante Bit-Rate, die vom Netz permanent zur Verfügung gestellt wird
  - die Anwendung gibt die *Peak Cell Rate* (*PCR*) an
  - das Netz garantiert die Zellrate bis zur PCR



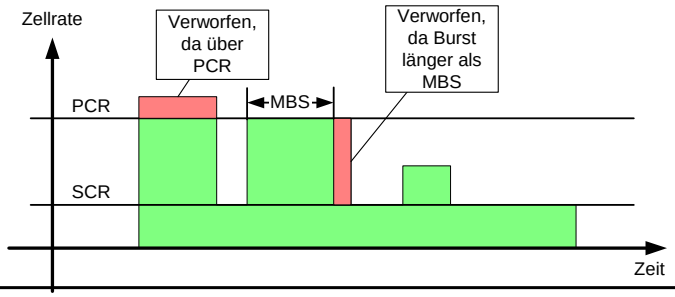
ATM-Dienstkategorien

- *UBR* (*Unspecified Bit Rate*): die Anwendung hat keine zeitkritischen Anforderungen (z.B. E-Mail oder Dateitransfer)
  - die Anwendung gibt die *Peak Cell Rate* (*PCR*) an
  - auch unterhalb von PCR können Zellen verworfen werden



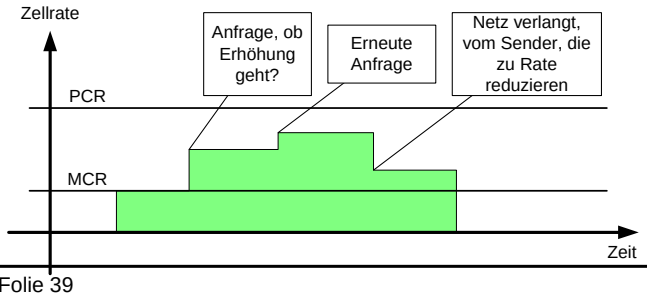
ATM-Dienstkategorien

- *VBR* (*Variable Bit Rate*): die Anwendung hat einen Bandbreiten-Bedarf, der um einen Mittelwert schwanken kann
  - die Anwendung gibt die *Peak Cell Rate* (*PCR*) an
  - die *Sustained Cell Rate* (*SCR*) wird garantiert
  - Zellraten über SCR werden nur für die Zeit bis zur *Maximum Burst Size* (*MBS*) garantiert



ATM-Dienstkategorien

- *ABR* (*Available Bit Rate*): der Bandbreitenbedarf der Anwendung ist vage und kann sich während der Sitzung ändern
  - die Anwendung gibt die *Minimum Cell Rate* (*MCR*) und die *Peak Cell Rate* (*PCR*) an, MCR wird garantiert
  - die Anwendung kann zusätzliche Bandbreiten bis PCR anfordern
  - das Netz kann eine Reduzierung bis MCR vorschreiben



ATM Adaption Layer

ATM Adaption Layer (*AAL*):

- Für die allermeisten übergeordneten Protokolle (z.B. IP) sind 48 Bytes Nutzdaten zu wenig.
- Der *ATM Adaption Layer* nimmt größerer Einheiten entgegen und versendet sie in einer Folge von Zellen.
- Bestimmte Zellen markieren das Ende einer Folge, zusätzlich werden die Daten über eine Prüfsumme gesichert.
  - Vorgehen wie bei der IP-Fragmentierung (siehe später) – wenn eine Zelle fehlt, wird das gesamte Paket verworfen
- Aus der Sicht übergeordneter Protokolle ist AAL wie ein Layer-2-Netzwerk zu sehen. Typische Anwendung: *IP over ATM*

ATM Kritik

Kritik zum ATM-Einsatz im LAN:

- ATM-Switches sind teuerer als Ethernet-Switches
- Verbindungsauf- und -abbau kostet Zeit
- Wahl der Dienstgüte-Parameter oft schwierig
- Broadcasting problematisch
- Konkurrenz GBit-Ethernet
- Bei "IP-over-ATM" wird ATM nicht voll ausgenutzt

$\Rightarrow$ ATM wird vorwiegend im Backbone-Bereich eingesetzt (z.B. Telefongesellschaften, große Unternehmen)

Rechnernetze

WS 2012/13

Kapitel 7

Kommunikation zwischen nicht-direkt verbundenen Rechnern

– Routing

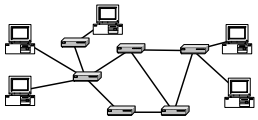
Kapitel 7: Kommunikation zwischen nicht-direkt verbundenen Rechnern – Routing

Ziel: Darstellung der Routing-Funktionalität

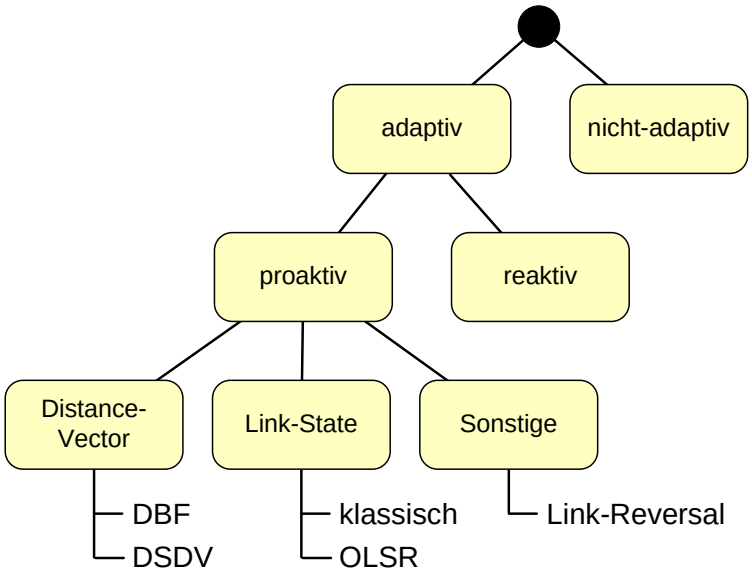
- Klassifikation von Routing-Verfahren
- Distance-Vector-Routing
- Link-State-Routing
- Link-Reversal-Routing

Routing vs. Switching

- Unter *Routing* wird die Festlegung eines Übertragungsweges bei der Versendung von Paketen in Computernetzen verstanden.
- Im Gegensatz zum Switching befasst sich Routing mit der Wegeauswahl – Switching befasst sich vorwiegend mit der Weiterleitung von Paketen.
- Beispiel: Routing im Internet mit IP-Routing
- Abstraktion des Routing-Problems:
  - ein Computer-Netzwerk wird durch einen Graphen aus Knoten und Kanten repräsentiert
  - die Kanten können Gewichte tragen, die z.B. die Übertragungszeit darstellen



Klassifikation von Routing-Verfahren



Klassifikation von Routing-Verfahren

Adaptivität

- *Nicht-adaptive* Verfahren: feste Routing-Tabellen
- *Adaptive* Verfahren: stellen sich automatisch auf die Netzwerktopologie ein

Zeitpunkt der Routen-Berechnung

- *Proaktiv*: Routen werden ständig im Hintergrund berechnet, insbesondere bevor eine Route benötigt wird
- *Reaktiv* (*on-demand*): Routen werden erst bei Bedarf berechnet, d.h. wenn ein Paket zur Vermittlung ansteht

Für große Netze sind vor allem *adaptive*, *proaktive* Verfahren interessant.

Routing-Verfahren

Gemeinsamkeit dieser Routing-Verfahren:

- Ziel ist, dass ein Router jeweils nur den Router findet, der ein Paket näher zu Ziel bringt (Hop-by-Hop-Routing)

Andere Möglichkeit: *Source Routing*:

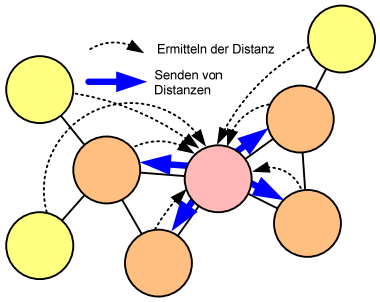
- Der gesamte Weg zum Ziel (nicht nur der nächste Hop) wird schon beim Sender (*Source*) ermittelt.
- Das Paket trägt die komplette Weginformationen zum Ziel
  - Router einfach (nur Liste abarbeiten)
  - Sender übernimmt die gesamte Wegfindung

→ Source Routing ist eher eine Randerscheinung

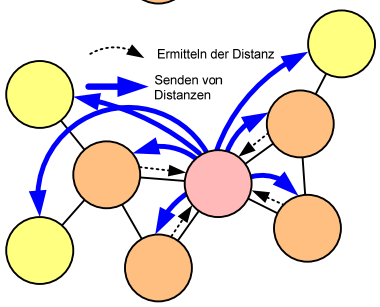
Klassifikation von Routing-Verfahren

Distance-Vector vs. Link-State

- *Distance-Vector*: Ermitteln der Distanz zu allen Knoten, Austausch von Distanz-informationen nur mit den Nachbarn



- *Link-State*: Messen der Distanz zu den Nachbarn, Austausch der Distanzinformationen mit allen Knoten

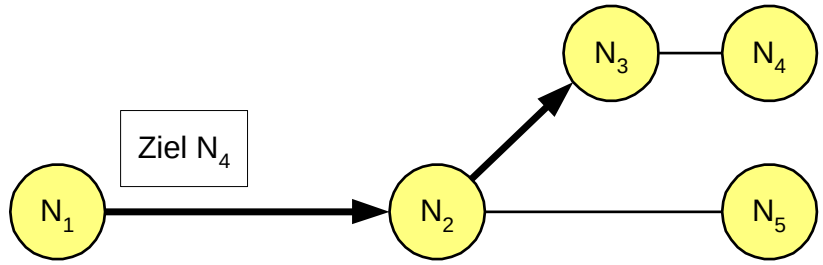


Distance Vector

Distributed Bellman-Ford (*DBF*)

- Das bedeutendste Distance-Vector-Verfahren
- Jeder Knoten kennt seine Nachbarn und den "Abstand" zu diesen.
- Die Abstände zu den Nachbarn können über bestimmte Pakete ermittelt werden, die über Broadcast verteilt werden.
- Jeder Knoten verwaltet eine Routing-Tabelle mit den Spalten
  - *Ziel*: an welchen Zielknoten soll ein Paket gesendet werden
  - *Hop*: über welchen direkten Nachbarn wird gesendet
  - *Metrik*: welche Distanz hat das Ziel

DBF



Routing-Tabelle von N_1

Ziel	Hop	Metrik
N_1	N_1	0
N_2	N_2	1
N_3	N_2	2
N_4	N_2	3
N_5	N_2	2

Routing-Tabelle von N_2

Ziel	Hop	Metrik
N_1	N_1	1
N_2	N_2	0
N_3	N_3	1
N_4	N_3	2
N_5	N_5	1

DBF

Wie entstehen die Einträge in den Routing-Tabellen?

- Für eine Routing-Tabelle eines Knotens N_a bezeichne
  - H_{ab} den Eintrag der Spalte *Hop* für einen Zielknoten N_b
  - M_{ab} den Eintrag der Spalte *Metrik* für einen Zielknoten N_b
- Eine Tabelle wird zu anfangs mit
  - $H_{ab} \leftarrow ?$
  - $M_{ab} \leftarrow \infty$

  initialisiert

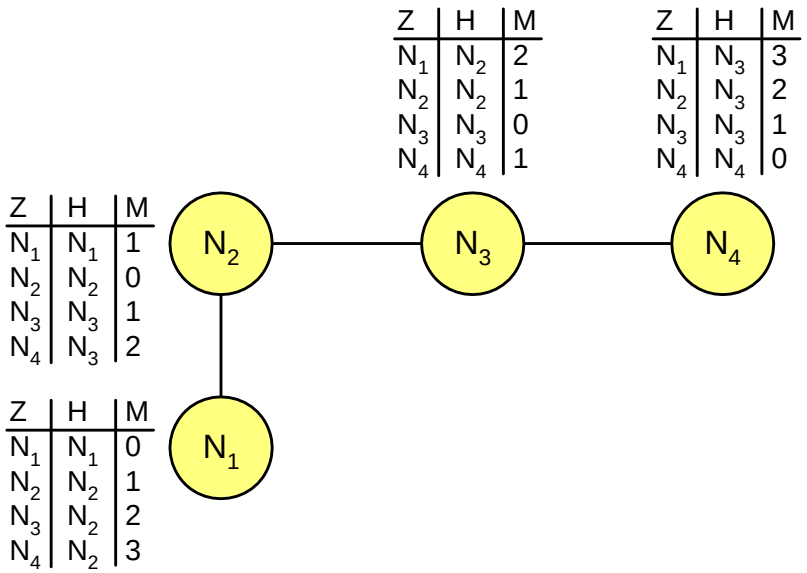
Routing-Tabelle von N_a

Ziel	Hop	Metrik
...	...	...
N_b	H_{ab}	M_{ab}
...	...	...

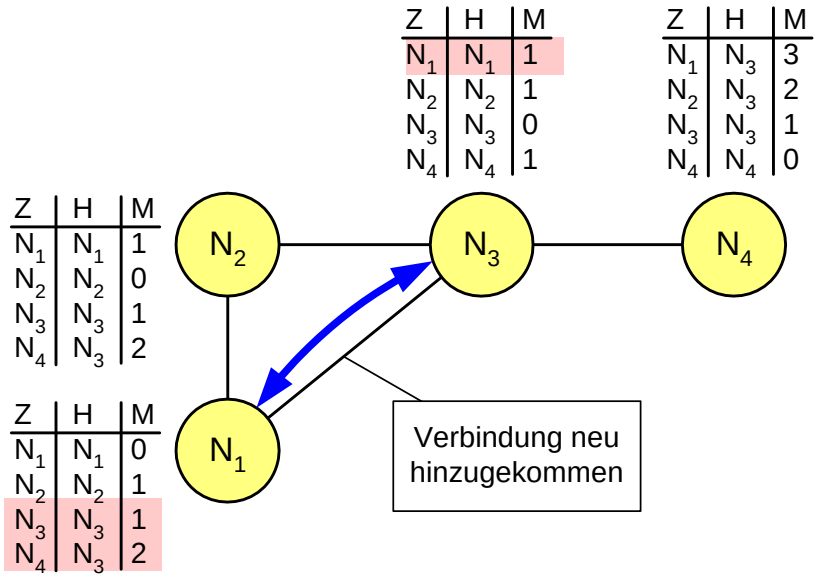
DBF

- Wir betrachten nun einen Knoten N_i:
  Dieser trägt sich zunächst selbst in die Tabelle ein mit
  - $H_{ii} \leftarrow N_i$
  - $M_{ii} \leftarrow 0$
- Für jeden direkten Nachbarn N_j von N_i wird eingetragen:
  - $H_{ij} \leftarrow N_j$
  - $M_{ij} \leftarrow 1$ (wir benutzen hier die Hop-Metrik)
- Jeder direkte Nachbar N_j von N_i sendet seine Routing-Tabelle an N_i. Für einen Tabelleneintrag zu N_k wird überprüft, ob $M_{ij} + M_{jk} < M_{ik}$. Wenn ja:
  - $H_{ik} \leftarrow N_j$
  - $M_{ik} \leftarrow M_{ij} + M_{jk}$ (= 1 + M_{jk} für Hop-Metrik)

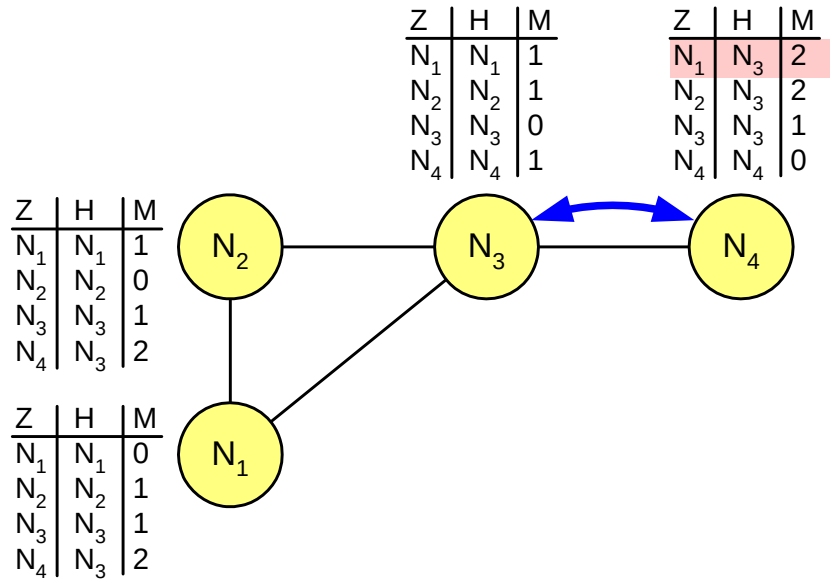
DBF – Beispiel 1



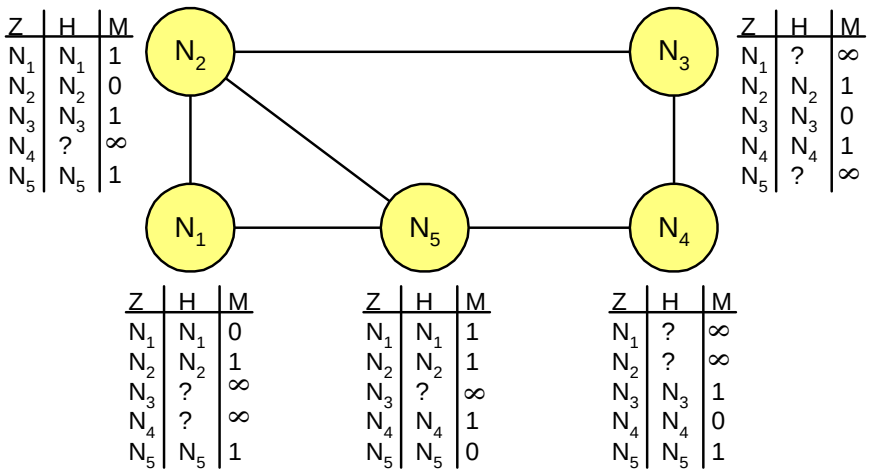
DBF – Beispiel 1



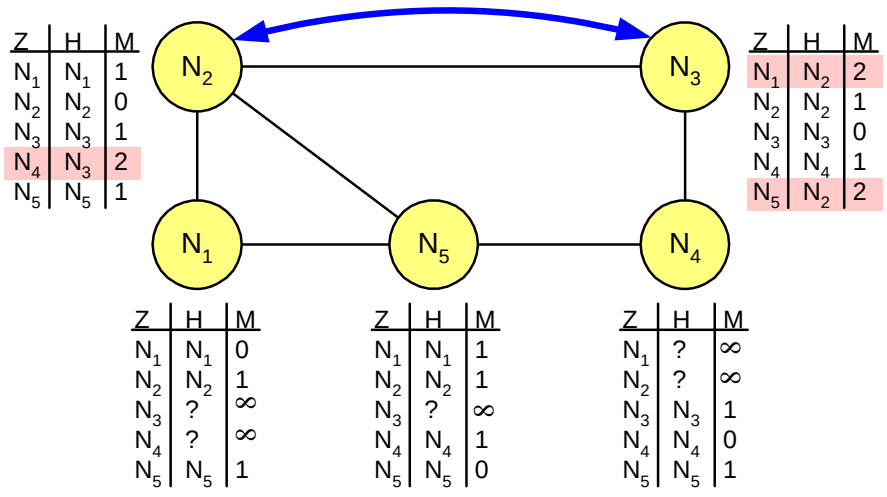
DBF – Beispiel 1



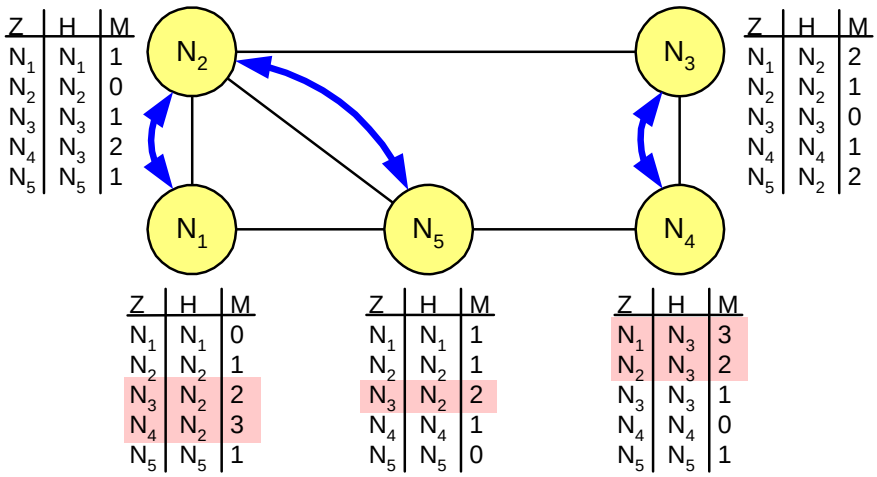
DBF – Beispiel 2



DBF – Beispiel 2



DBF – Beispiel 2



Z	H	M
N_1	N_1	1
N_2	N_2	0
N_3	N_3	1
N_4	N_3	2
N_5	N_5	1

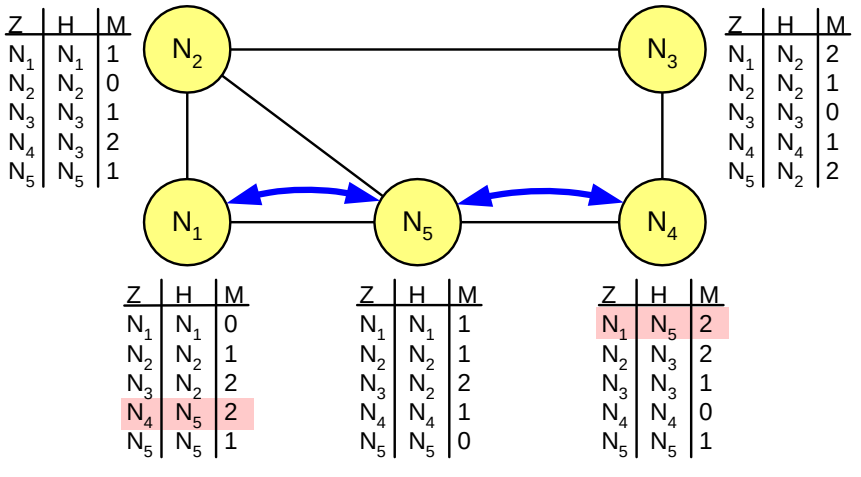
Z	H	M
N_1	N_2	2
N_2	N_2	1
N_3	N_3	0
N_4	N_4	1
N_5	N_2	2

Z	H	M
N_1	N_1	0
N_2	N_2	1
N_3	N_2	2
N_4	N_2	3
N_5	N_5	1

Z	H	M
N_1	N_1	1
N_2	N_2	1
N_3	N_2	2
N_4	N_4	1
N_5	N_5	0

Z	H	M
N_1	N_3	3
N_2	N_3	2
N_3	N_3	1
N_4	N_4	0
N_5	N_5	1

DBF – Beispiel 2



Z	H	M
N_1	N_1	1
N_2	N_2	0
N_3	N_3	1
N_4	N_3	2
N_5	N_5	1

Z	H	M
N_1	N_2	2
N_2	N_2	1
N_3	N_3	0
N_4	N_4	1
N_5	N_2	2

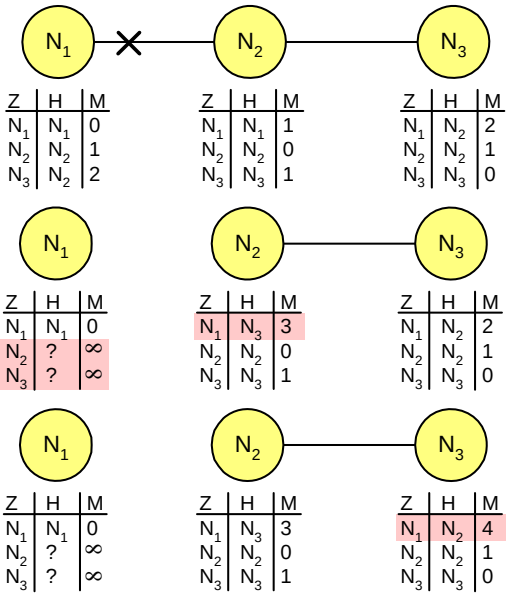
Z	H	M
N_1	N_1	0
N_2	N_2	1
N_3	N_2	2
N_4	N_5	2
N_5	N_5	1

Z	H	M
N_1	N_1	1
N_2	N_2	1
N_3	N_2	2
N_4	N_4	1
N_5	N_5	0

Z	H	M
N_1	N_5	2
N_2	N_3	2
N_3	N_3	1
N_4	N_4	0
N_5	N_5	1

Count-to-Infinity-Problem

- Die Verbindung zwischen N_1 und N_2 wird getrennt
- Dennoch besitzen N_2 und N_3 eine Route zu N_1
- Die Distanzen erhöhen sich mit jedem Austausch, werden aber nie unendlich



Count-to-Infinity-Problem

Mögliche Lösungen:

- Man definiert einen kleinen, endlichen Wert (z.B. 16) als unendlich.
- *Split Horizon*: sendet ein Knoten seine Routing-Tabelle an seine Nachbarn, schickt er nicht die Einträge, die er vom Nachbarn erhalten hat
  - hiermit werden "Schleifen" zwischen zwei Nachbarn vermieden
  - komplexere Problemfälle werden aber weiterhin nicht erkannt
- Man definiert Versionsnummern der Einträge – neue Versionen setzen sich immer durch ⇒ *DSDV* (*Destination-Sequenced Distance Vector*).

DSDV

DSDV (*Destination-Sequenced Distance Vector*) behebt das Count-to-Infinity-Problem:

- Periodisch oder immer dann, wenn ein Knoten eine Topologieänderung erkannt hat, werden Distanzinformationen ausgetauscht.
- DSDV erweitert die Routing-Tabelle um einen Eintrag *Sequenznummer* (später genannt S_{ij}). Die Sequenznummer wird i.d.R. von dem Knoten vergeben, zu dem die Distanz gemessen wurde.
- Bei jeder Verbreitung wird die Sequenznummer erhöht. Damit kann ermittelt werden, wie aktuell eine Eintrag ist. Ein Eintrag wird nur übernommen, wenn
  - die Sequenznummer sich erhöht hat
  - oder die Sequenznummer gleich geblieben ist und die Gesamtdistanz sich verringert hat

DSDV

- Jeder Knoten N_i initialisiert seine Tabelle mit
  - $H_{ii} \leftarrow N_i$
  - $M_{ii} \leftarrow 0$
  - $S_{ii} \leftarrow 0$

  und trägt für alle anderen Knoten N_m ein
  - $H_{im} \leftarrow ?$
  - $M_{im} \leftarrow \infty$
  - $S_{im} \leftarrow -1$

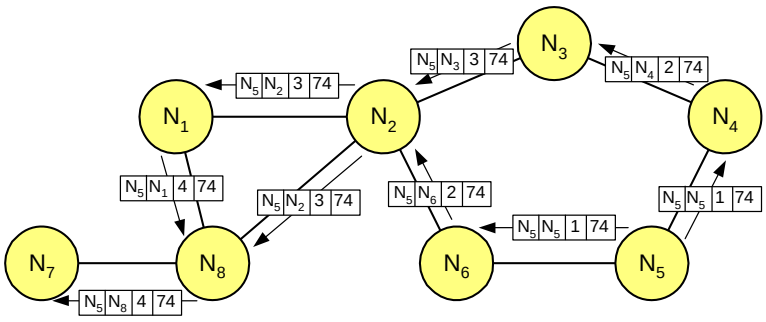
Ziel	Hop	Metrik	SeqNr
...	...	...	...
N_m	?	∞	-1
...	...	...	...
N_i	N_i	0	0
...	...	...	...

DSDV

- Ein Nachbar N_j von N_i sendet nun Distanzinformationen bzgl. eines Knotens N_k an N_i. N_i prüft, ob
  - $S_{ik}<S_{jk}$ oder
  - $S_{ik}=S_{jk}$ und $M_{ij}+M_{jk}<M_{ik}$

  Ist dies der Fall, aktualisiert N_i die Tabelle wie folgt
  - $H_{ik} \leftarrow N_j$
  - $M_{ik} \leftarrow M_{ij} + M_{jk}$ (= 1 + M_{jk} für Hop-Metrik)
  - $S_{ik} \leftarrow S_{jk}$
- Hat ein Knoten N_j den eigenen Eintrag an alle Nachbarn gesendet, erhöht er *seine* Sequenznummer um 2, d.h.
  - $S_{jj} \leftarrow S_{jj}+2$
  - Begründung für "2" erfolgt später

DSDV – Beispiel



Routing-Tabelle von N_2

Ziel	Hop	Metrik	SeqNr
N_1	N_1	1	70
N_2	N_2	0	72
N_3	N_3	1	70
N_4	N_3	2	76
N_5	N_6	2	74
N_6	N_6	1	56
N_7	N_8	2	56
N_8	N_8	1	58

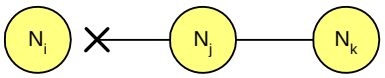
Routing-Tabelle von N_5

Ziel	Hop	Metrik	SeqNr
N_1	N_6	3	70
N_2	N_6	2	72
N_3	N_4	2	70
N_4	N_4	1	76
N_5	N_5	0	74
N_6	N_6	1	56
N_7	N_6	4	56
N_8	N_6	3	58

DSDV – Unterbrochene Verbindungen

Entdeckt ein Knoten N_i eine Unterbrechung zu N_j wird folgendes ausgeführt

- Es werden alle Zeilen der Routing-Tabelle gesucht, die N_j als nächsten Hop ausweisen
- In diesen Zeilen wird vermerkt, dass N_j nicht mehr erreichbar ist
- Konkret:
  für alle k mit $H_{ik}=N_j$:
  - $H_{ik} \leftarrow$?
  - $M_{ik} \leftarrow \infty$
  - $S_{ik} \leftarrow S_{ik}+1$



Routing-Tabelle von N_i

Ziel	Hop	Metrik	SeqNr
...	...	...	...
N_i	N_i	0	74
...	...	...	...
N_j	N_j	1	80
...	...	...	...
N_k	N_j	2	62
...	...	...	...

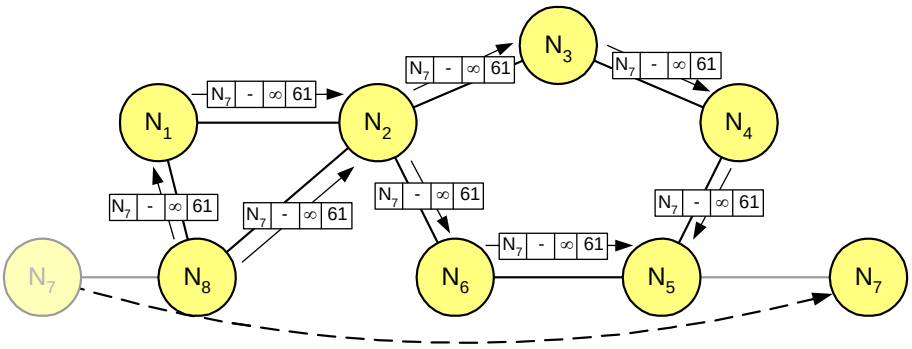
Routing-Tabelle von N_i

Ziel	Hop	Metrik	SeqNr
...	...	...	...
N_i	N_i	0	74
...	...	...	...
N_j	?	∞	81
...	...	...	...
N_k	?	∞	63
...	...	...	...

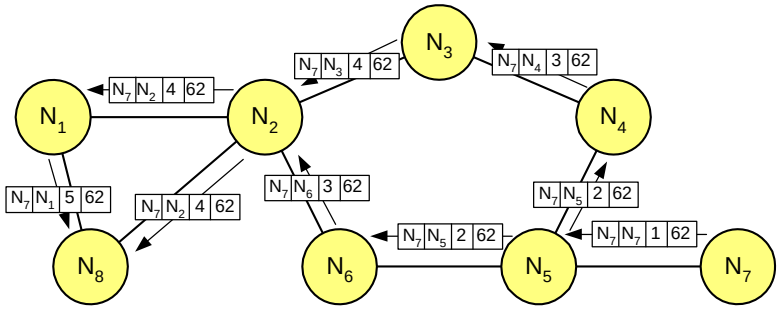
DSDV – Unterbrochene Verbindungen

- Die Verbindungsunterbrechung von N_i zu N_j wird bekannt:
  - keiner benutzt mehr N_i als nächsten Hop, um ein Paket an N_j zu schicken
  - umgekehrt wird N_i nicht mehr N_j als nächsten Hop verwenden um N_k zu erreichen
- Diese Regel ist die einzige Ausnahme, in der S_{ik} nicht von N_k erhöht wird.
- Warum gerade um 1 erhöhen?
  - Die Sequenznummer ist nun höher als die alter Routen zu N_j oder N_k
  - Die Sequenznummer ist kleiner als die einer neuen Route zu N_i (die ja um 2 erhöht wurde)

DSDV – Beispiel



DSDV – Beispiel



Routing-Tabelle von N_2

Ziel	Hop	Metrik	SeqNr
N_1	N_1	1	74
N_2	N_2	0	76
N_3	N_3	1	74
N_4	N_3	2	80
N_5	N_6	2	80
N_6	N_6	1	60
N_7	N_6	3	62
N_8	N_8	1	62

Routing-Tabelle von N_5

Ziel	Hop	Metrik	SeqNr
N_1	N_6	3	74
N_2	N_6	2	76
N_3	N_4	2	74
N_4	N_4	1	80
N_5	N_5	0	80
N_6	N_6	1	60
N_7	N_7	1	62
N_8	N_6	3	62

RIP

Routing Information Protocol (*RIP*):

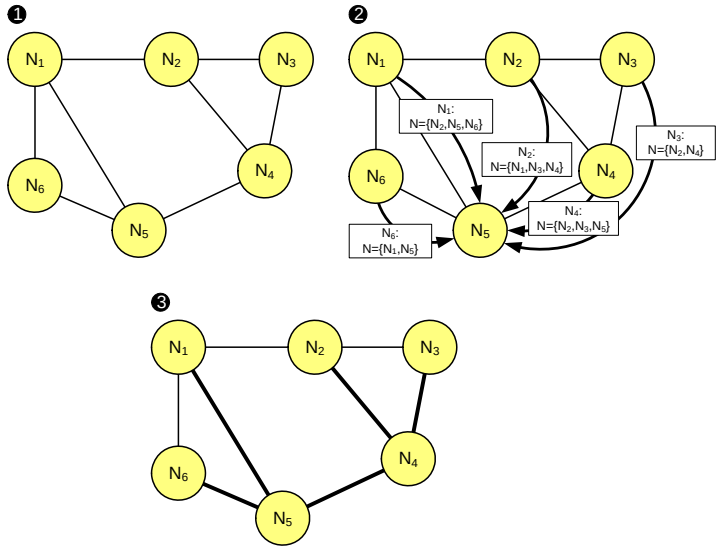
- Die IP-Variante eines Distance-Vector-Verfahrens
- Basiert auf DBF
- Router versenden alle 30 s eine Aktualisierung, außerdem immer dann, wenn eine andere Aktualisierungsnachricht zu einer Änderung der Routing-Tabelle geführt hat.
- Entfernungen
  - Werte liegen zwischen 1 und 15
  - Hop-Metrik
  - 16 bedeutet unendlich

Link-State-Routing

Ein anderer Ansatz: Link-State-Routing

- Wie bei Distance Vector kann jeder Knoten die Menge seiner Nachbarn und den Abstand zu ihnen bestimmen.
- Diese Informationen werden nun im gesamten Netzwerk verteilt.
- Erhält jeder Knoten die Menge der Nachbarn aller anderen Knoten, kann jeder Knoten die Netzwerktopologie des gesamten Netzwerks *lokal darstellen.*
- Über Kürzeste-Wege-Algorithmen kann sich jeder Knoten *lokal* die kürzesten Wege zu allen Zielen berechnen.

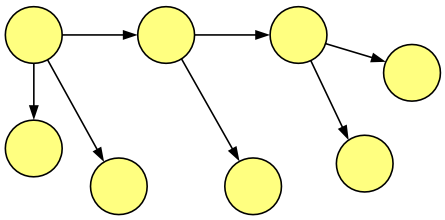
Link-State-Routing



Fluten

Wie verteilt man die Nachbarschaftsinformation im Netzwerk?

- hierzu wird das so genannte *Fluten* verwendet
- vereinfacht: ein Paket wird an alle Nachbarn versendet, die wiederum dieses Paket an ihre Nachbarn weitergeben etc. – hierdurch wird das gesamte Netzwerk erreicht



Fluten

Fluten im Detail:

- Ein Knoten, der seine Nachbarschaftsbeziehungen im Netz propagieren möchte, erzeugt ein *Link-State-Paket* (*LSP*). Es enthält
  - seine eindeutige Knoten-Kennung (*ID*)
  - die Liste der direkt verbundenen Nachbarn
  - eine Sequenznummer (*s*)
  - eine Lebensdauer des Pakets (*TTL*, *Time to Live*)

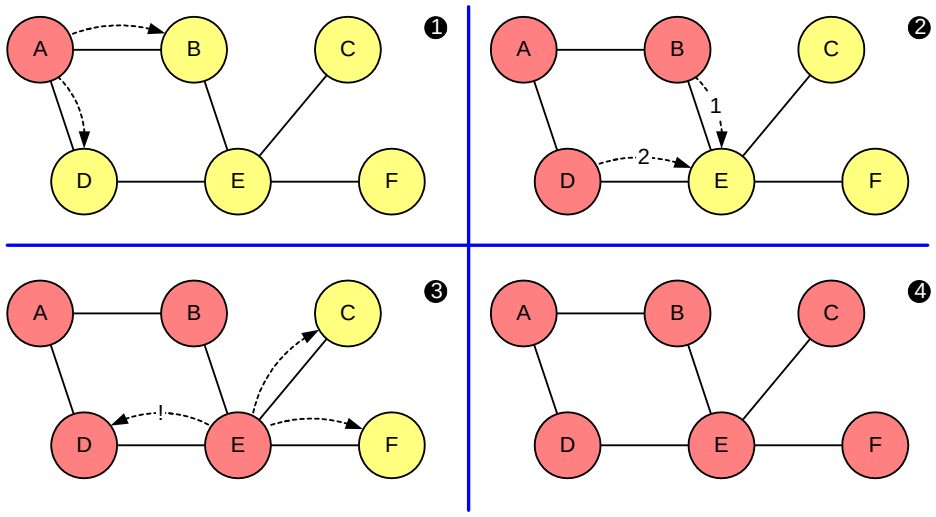
Fluten

Fluten für Link-State-Routing:

- Ein Knoten sendet LSPs an seine Nachbarn
  - initial
  - periodisch
  - bei Änderung seiner Nachbarschaftsmenge

  nach jedem Senden wird s um 1 erhöht
- Wenn ein Knoten ein LSP empfängt:
  - hat er schon ein LSP von diesem Knoten empfangen und ist $s_{neu} \leq s_{alt} \Rightarrow$ das Paket wird ignoriert
  - sonst: er speichert das LSP in seinem lokalen Datenbereich
  - er vermindert TTL um 1. Ist TTL>0 sendet er das LSP an alle Nachbarn, außer an den, von dem er das LSP erhalten hat

Fluten



Fluten

Zusammengefasst die Maßnahmen, um Fluten zu optimieren:

- Man schickt ein Paket nicht an den Nachbarn zurück, von dem man es erhalten hat.
- Man verwirft Pakete, die man schon einmal erhalten hat.
- TTL stellt sicher, dass LSPs nicht ewig zirkulieren.

Zusätzlich:

- LSPs werden selten (im Abstand von Stunden) geflutet.
- Die Sequenznummern stellen sicher, dass sich neue Informationen durchsetzen, selbst wenn alte noch zirkulieren.
- Sequenznummern dürfen nicht überlaufen. Beim längeren Routerausfall darf die Sequenznummer wieder bei 0 starten (alte Informationen wurden ja gelöscht).
- Topologieinformationen werden, wenn nicht erneuert, nach einer gewissen Zeit gelöscht.

Dijkstra

- Nachdem ein Knoten alle LSPs bekommen hat, kann er die kürzesten Wege zu allen Knoten lokal berechnen.
- Möglichkeit: Algorithmus von Dijkstra zur *Bestimmung von Laufweg-Quellbäumen.*
- Idee:
  - Ausgehend vom *Quellknoten* wird sukzessive der *Quellbaum* aufgebaut.
  - Es wird immer der Knoten mit dem kleinsten Abstand zum schon bestehenden Quellbaum ausgewählt.
  - Schritthaltend wird die Abstandstabelle aktualisiert.

Dijkstra

Im Detail:

- Sei
  - K_q der Quellknoten,
  - M die Menge der schon betrachteten Knoten,
  - T der sukzessive aufgebaute Quellbaum und
  - d_i der bisher berechnete Abstand von K_q zu N_i
- Initialisierung:
  - $M=\{K_q\}$
  - $T=\{K_q\}$
  - Für alle Nachbarn N_i von K_q: d_i=Abstand von K_q zu N_i
  - Für alle anderen Knoten: d_i=unendlich

Dijkstra

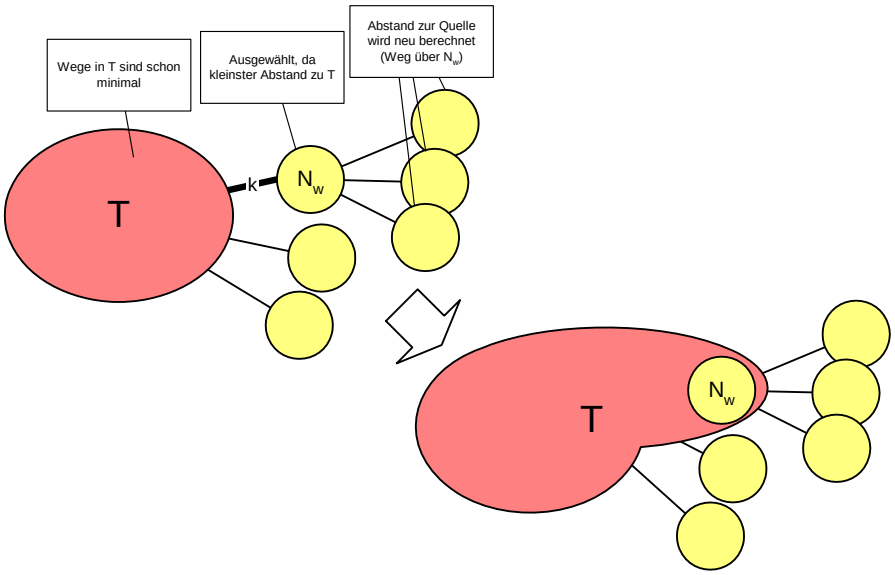
Solange noch nicht alle Knoten in M erfasst sind:

- Bestimme den Knoten N_w mit dem geringsten Abstand zu K_q, genauer: wähle N_w mit $d_w = \min\{d_i \mid N_i \notin M\}$
- Bestimme die Kante k mit dem kleinsten Gewicht, die N_w und ein Knoten aus T verbindet
- $M = M \cup \{N_w\}$, $T = T \cup \{k\}$
- $d_i = \min\{d_i, d_w + \text{Abstand}(N_w, N_i)\}$ für alle $N_i \notin M$

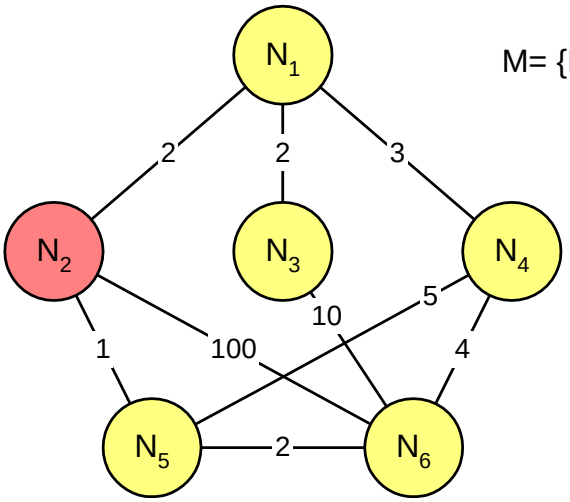
Danach:

- In T steht der gesuchte Quellbaum
- In d_i steht der kürzeste Abstand vom Quellknoten zu N_i

Dijkstra



Dijkstra – Beispiel

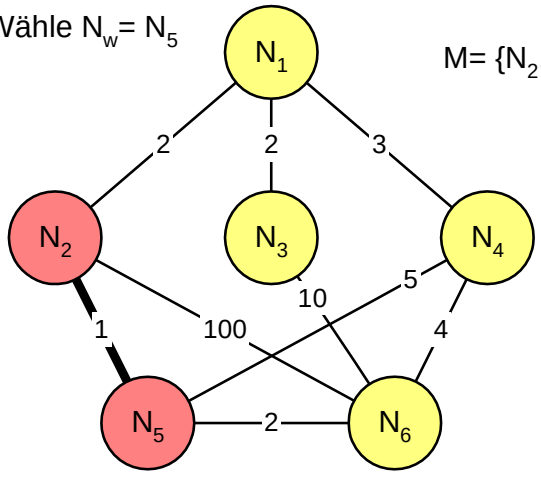


M= {N_2} T= {N_2}

d_1= 2
d_2= - ✓
d_3= ∞
d_4= ∞
d_5= 1 ◄min
d_6= 100

Dijkstra – Beispiel

Wähle N_w= N_5



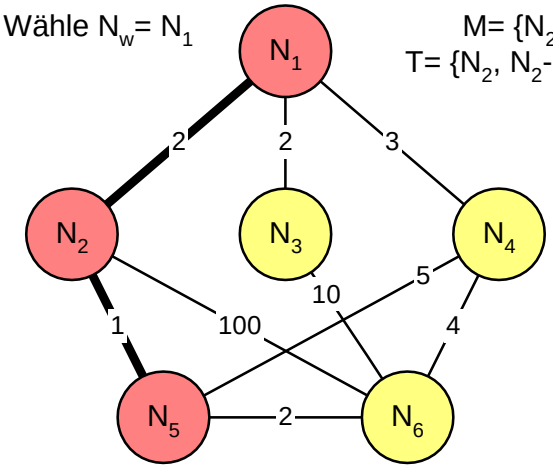
M= {N_2, N_5} T= {N_2, N_2->N_5}

d_1= 2 ◄min
d_2= - ✓
d_3= ∞
d_4= 6
d_5= 1 ✓
d_6= 3

Dijkstra – Beispiel

Wähle $N_w = N_1$

$M = \{N_2, N_5, N_1\}$
$T = \{N_2, N_2\text{->}N_5, N_2\text{->}N_1\}$

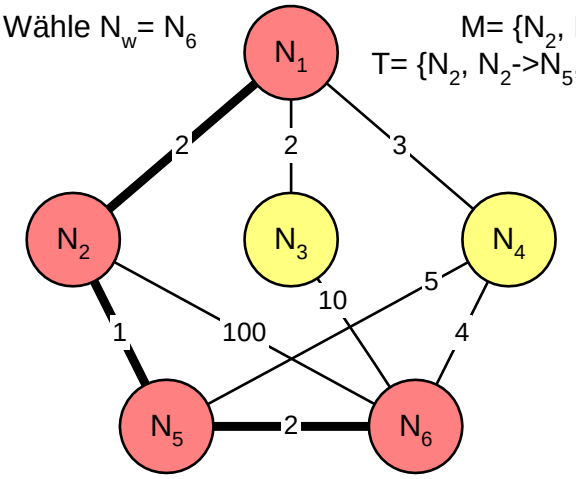


$d_1 = 2$ ✓
$d_2 = -$ ✓
$d_3 = 4$
$d_4 = 5$
$d_5 = 1$ ✓
$d_6 = 3$ ◄min

Dijkstra – Beispiel

Wähle $N_w = N_6$

$M = \{N_2, N_5, N_1, N_6\}$
$T = \{N_2, N_2\text{->}N_5, N_2\text{->}N_1, N_5\text{->}N_6\}$



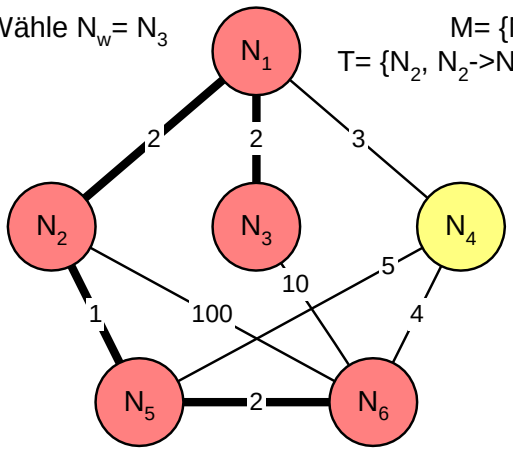
$d_1 = 2$ ✓
$d_2 = -$ ✓
$d_3 = 4$ ◄min
$d_4 = 5$
$d_5 = 1$ ✓
$d_6 = 3$ ✓

Dijkstra – Beispiel

Wähle $N_w = N_3$

$M = \{N_2, N_5, N_1, N_6, N_3\}$

$T = \{N_2, N_2\text{->}N_5, N_2\text{->}N_1, N_5\text{->}N_6, N_1\text{->}N_3\}$



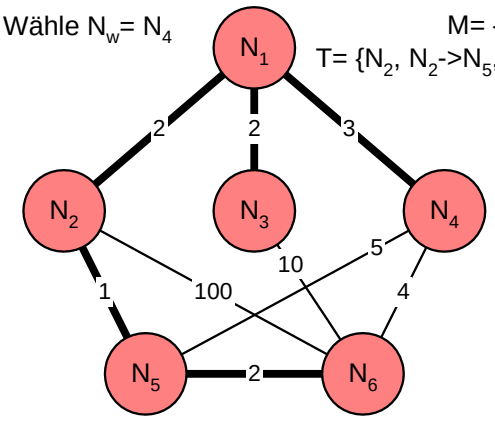
$d_1 = 2$ ✓
$d_2 = -$ ✓
$d_3 = 4$ ✓
$d_4 = 5$ ◄min
$d_5 = 1$ ✓
$d_6 = 3$ ✓

Dijkstra – Beispiel

Wähle $N_w = N_4$

$M = \{N_2, N_5, N_1, N_6, N_3, N_4\}$

$T = \{N_2, N_2\text{->}N_5, N_2\text{->}N_1, N_5\text{->}N_6, N_1\text{->}N_3, N_1\text{->}N_4\}$



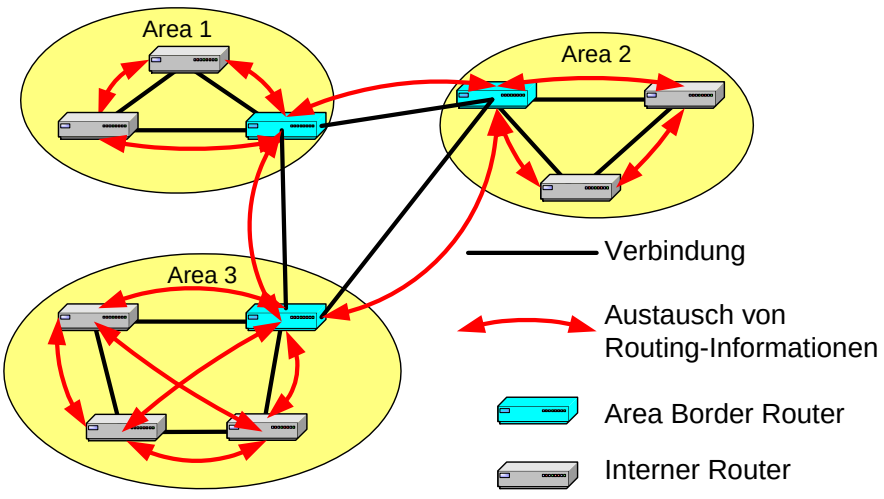
$d_1 = 2$ ✓
$d_2 = -$ ✓
$d_3 = 4$ ✓
$d_4 = 5$ ✓
$d_5 = 1$ ✓
$d_6 = 3$ ✓

OSPF

Ein Realisierung des Link-State-Ansatzes:
Open Shortest Path First (*OSPF*):

- *"Open"*: offener Standard
- *"Shortest Path First"*: in Anlehnung an Dijkstras SPF-Algorithmus zur Berechnung der kürzesten Pfade.
- OSPF unterstützt die Authentifikation von Routing-Informationen mittels Kryptografie.
- OSPF unterstützt hierarchisches Routing
  - Bereiche (Areas)
  - ausgezeichnete Router, die Bereiche verbinden

OSPF



OLSR

OLSR (*Optimized Link State Routing*): Optimierung gegenüber dem klassischem Link-State:

- Verkleinern der Link-State-Pakete: statt alle Nachbarn könnte eine *geeignete* Untermenge von Nachbarknoten spezifiziert werden. Diese Untermenge muss so gewählt werden, dass sich jeder Knoten immer noch eine Routing-Tabelle zusammenstellen kann.
- Optimieren des Flutens: Kontrollpakete könnten nicht an alle Nachbarn, sondern nur an eine *geeignete* Untermenge versendet werden. Die Untermenge muss so gewählt werden, dass immer noch jeder Knoten ein bestimmtes Kontrollpaket empfängt.

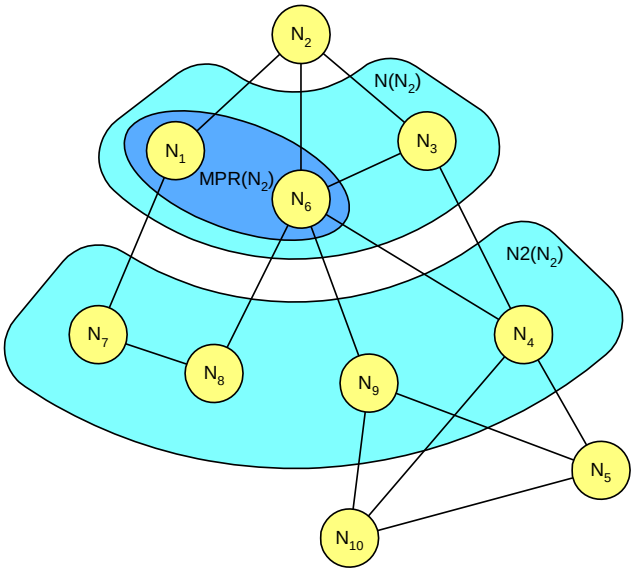
OLSR

Wir zeigen hier nur die zweite Optimierung.

Bezeichnungen

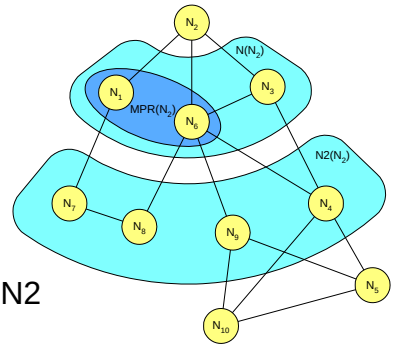
- $N(N_i)$ bezeichne die Nachbarn von N_i
- $N2(N_i)$ bezeichne die Knoten, die *exakt* 2 Schritte von N_i entfernt sind
- $MPR(N_i)$ bezeichne die Multipoint Relays von N_i; $MPR(N_i)$ ist dabei eine Untermenge von $N(N_i)$, die so gewählt wird, dass jeder Knoten aus $N2(N_i)$ eine direkte Verbindung zu $MPR(N_i)$ hat

OLSR – MPR



OLSR – MPR

- Es gibt u.U. verschiedene Möglichkeiten für MPR.
- OLSR verlangt nicht, dass MPR möglichst klein ist – aber: je kleiner MPR ist, desto effizienter ist OLSR.
- Bei jeder Änderung von N und N2 müssen MPR und MPRsel neu berechnet werden.
- OLSR macht einen Vorschlag zur Berechnung von MPR und MPRsel – hierzu werden die Hello-Pakete um die Felder N und MPR erweitert.

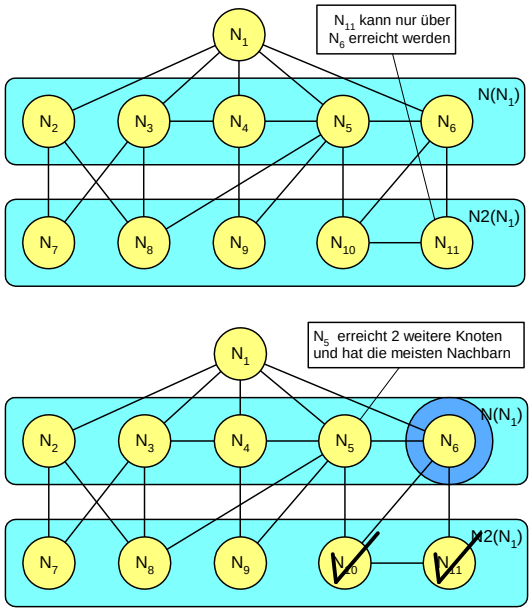


OLSR – MPR

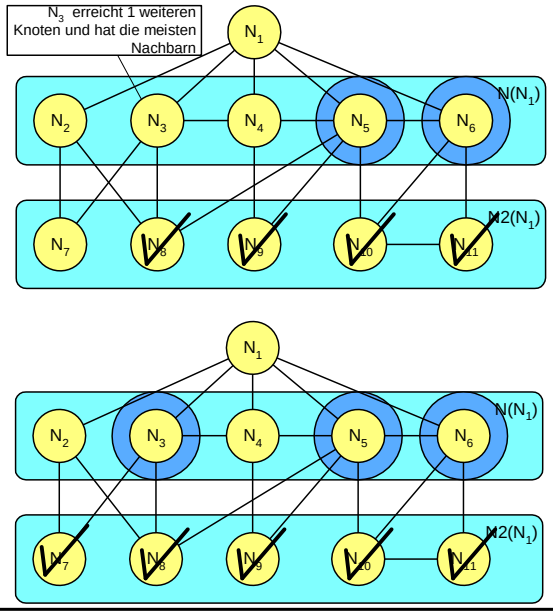
Berechnung von MPR

- Beginne mit einer leeren Menge MPR.
- Füge alle Knoten aus N hinzu, die eine einzige Verbindung zu Knoten aus N2 darstellen.
- Solange es noch Knoten in N2 gibt, die nicht über Knoten aus MPR erreichbar sind:
  - Wähle einen Knoten aus N, der die meisten noch nicht erreichbaren Knoten aus N2 erreicht.
  - Gibt es mehrere Knoten mit der gleichen Zahl, wähle den Knoten mit den meisten Nachbarn.
  - Füge diesen Knoten zu MPR hinzu.
- Wenn es Knoten N_j aus MPR gibt, so dass MPR\{N_j} immer noch alle Knoten aus N2 erreicht, dann lösche N_j.

OLSR – MPR



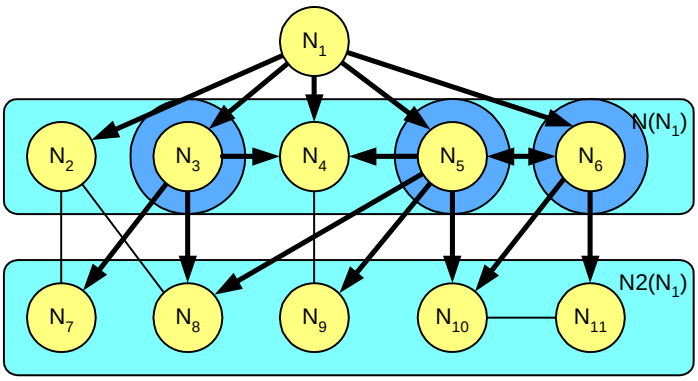
OLSR – MPR



OLSR – Funktionsweise

Welchen Nutzen bringen MPR?

- Nur Multipoint Relays werden zum Fluten benutzt.
- Alle anderen Knoten verarbeiten ein Link-State-Paket lediglich.



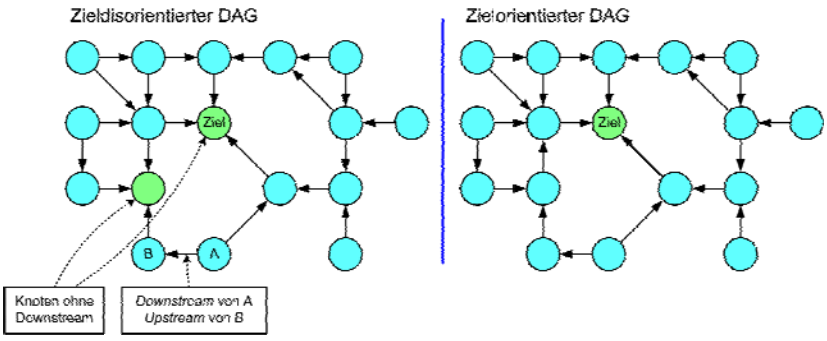
Link Reversal Routing

Link Reversal Routing – ein anderer Ansatz:

- Bisher vorgestellte Verfahren versuchen einen möglichst optimalen Weg zum Ziel zu finden.
- Ansatz des *Link Reversal Routing*: überhaupt einen Weg zum Ziel finden, u.U. suboptimal.
  - Damit kann das Aufkommen der Kontrollnachrichten drastisch reduziert werden.
  - Auf Fluten wird verzichtet.
  - Geeignet für Netze, deren Topologie sich schnell ändert.
- Die Weginformation wird durch einen *Directed Acyclic Graph* (*DAG*) repräsentiert.
  - Nachbarn werden über *gerichtete* Kanten verbunden.
  - Der Graph ist frei von Zyklen.

Link Reversal Routing

- Für ein DAG wird ein Zielknoten ausgewählt
  - *Zielorientierter DAG*: der Zielknoten ist der einzige Knoten, *der keine Downstreams hat.*
  - *Zieldisorientierter DAG*: es gibt mehrere Knoten, die keine Downstreams haben.



Link Reversal Routing

- Idee: Ein zieldisorienterter DAG kann durch Umkehrung einiger Kanten in einen zielorientierten DAG transformiert werden.
- Für jeden möglichen Zielknoten im Netz muss ein Knoten einen zielorientierten DAG aufbauen.
- Pakete werden entlang der gerichteten Kanten gesendet. Hat ein Knoten mehrere Downstreams, so wird ein beliebiger Downstream gewählt.
- Da der Zielknoten die einzige "Senke" ist und der Graph keine Zyklen hat, muss ein Paket immer zum Ziel gelangen.

Link Reversal - Varianten

Varianten

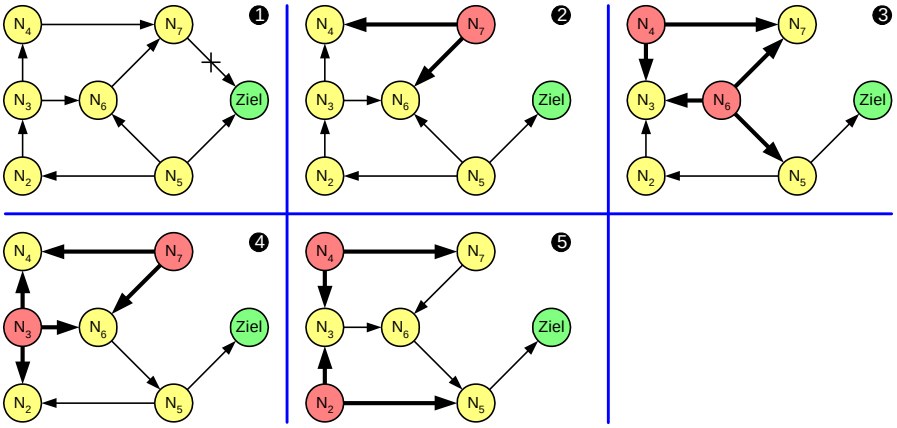
- Die Varianten unterscheiden sich in der Beantwortung folgender Frage:

> *Wie verhält sich ein Knoten ungleich dem Zielknoten, der aufgrund einer Topologie-Änderung alle Downstreams verloren hat?*

  - Full Reversal
  - Listenbasiertes Partial Reversal

Full Reversal

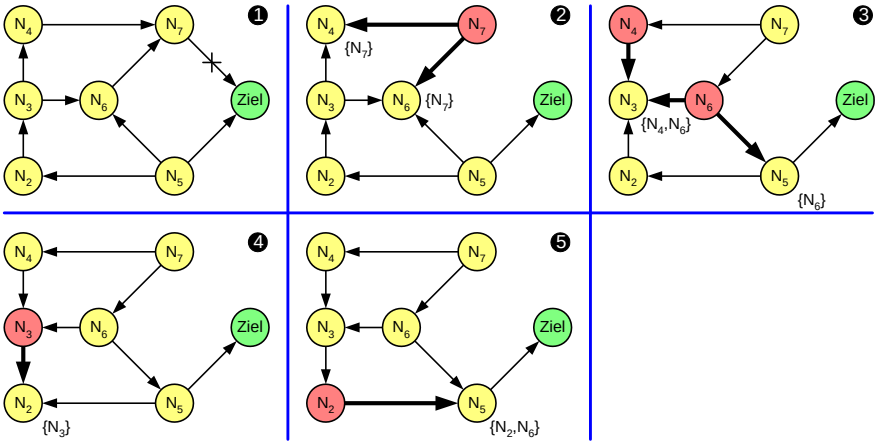
- Hat ein Knoten alle Downstreams verloren, kehrt er *alle* Kanten um.



Partial Reversal (listenbasiert)

- Nachteil des Full Reversals: einige Knoten müssen ihre Kanten mehrmals umkehren. Dies führt zu unnötigen Netzbelastungen.
- Idee des Partial Reversals (listenbasiert):
  - Jeder Knoten führt eine Liste von Knoten, die im letzten Schritt ihre Kanten umgekehrt haben.
  - Hat ein Knoten alle Downstreams verloren, so werden nur Kanten zu Knoten umgekehrt, die nicht in der Liste aufgeführt sind.
  - Ausnahme: alle Nachbarn befinden sich in der Liste. Dann werden alle Kanten umgekehrt.

Partial Reversal (listenbasiert)



Rechnernetze

WS 2012/13

Kapitel 8

Vermittlung im Internet

– IP

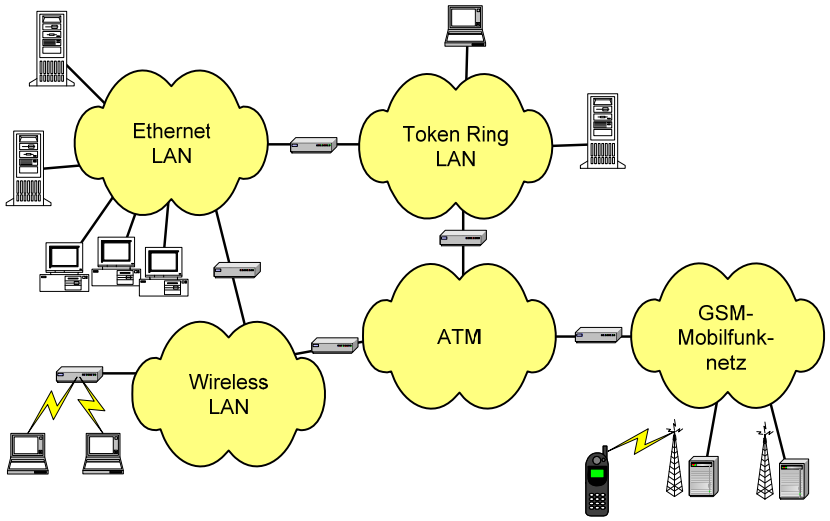
Kapitel 8: Vermittlung im Internet – IP

Ziel: Darstellung des Vermittlungsprotokolls des Internets.

- IP-Adressierung, Subnetting, Adressauflösung
- Aufbau von IP-Paketen
- Fragmentierung
- IPv6

Internet, Internetworking

Ein winziger Ausschnitt des Internets...

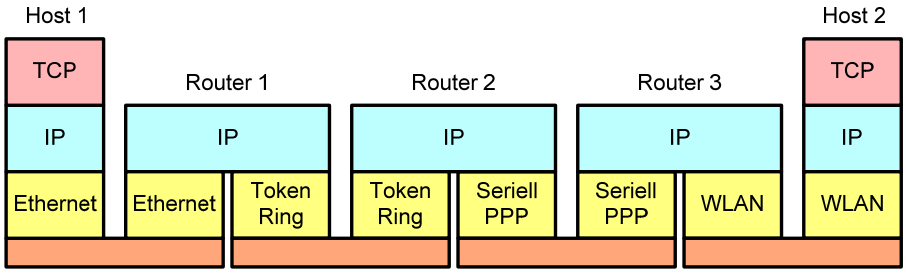


Internet, Internetworking

- Das Internet ist ein weltweiter Zusammenschluss verschiedenster Netzwerke wie Ethernet-, WLAN-, Token-Ring-Netze etc., verbunden durch TCP/IP-Protokolle.
- Zwei Hauptprobleme:
  - Heterogenität: Wie schafft man es, dass die verschiedenen Netzwerk-Technologien zusammen arbeiten?
  - Skalierbarkeit: Wie schafft man es, ein Netz mit mehreren Million Rechnern aufzubauen?

IP

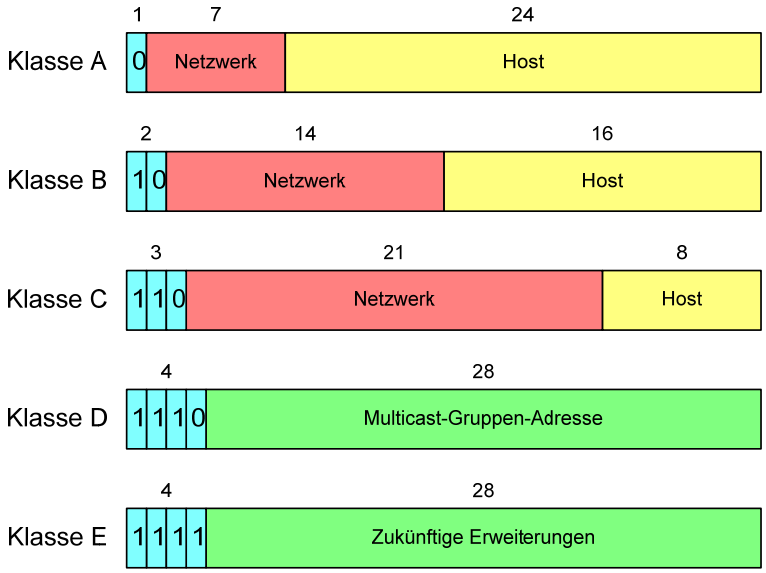
- IP läuft sowohl auf Hosts, als auch auf Routern.
- Router haben Netzwerkadapter zu mindestens zwei Netzwerken.



IP-Adressierung

- Jeder Host muss weltweit eindeutig identifizierbar sein (seine IP-Adresse, IPv4: 32 Bit).
- Warum nicht MAC-Adressen? (z.B. Ethernet-Adressen)
  - MAC-Adressen werden nach Herstellern vergeben – Karten mit ähnlichen Adressen können weit entfernt eingesetzt werden.
  - Eine MAC-Adresse lässt daher keinen Rückschluss auf den Standort im Netzwerk zu, sie ist daher zum Routing ungeeignet.
- IP-Adressen sind *hierarchisch*, d.h. sie spiegelt die Topologie des Internets wider.
- Router besitzen mindestens zwei IP-Adressen.
- Lange teilte man die 2^{32} möglichen Adressen in *Klassen* ein.

IP-Adressierung



IP-Adressierung

- Der Anteil "Netzwerk" adressiert das Netzwerk, an das der Host angeschlossen wurde.
- Der Anteil "Host" adressiert den Host im Netzwerk.
- Warum Klassen?
  - Bei der Vergabe von Adressen kann die Größe des Netzwerks berücksichtigt werden.
  - Anhand der Klasse kann man erkennen, wie groß der Hostanteil ist.

Klasse	Mögliche Netzwerke (weltweit)	Mögliche Hosts im Netzwerk
A	126	16 777 214
B	16 382	65 534
C	2 097 150	254

IP-Adressierung

Notation der 32-Bit-Adresse als 4 Oktetts:

- z.B. **`141.75.20.10`**
- Oktett: Dezimalzahl von 0 bis 255 (repräsentiert 8 Bit)

Einige Regeln:

- Nur 1-Bits (**`111…111`**) dürfen im Host-Anteil nicht vorkommen, um einen bestimmten Host zu adressieren. Diese Adresse beschreibt die *Broadcast-Adresse* des Netzwerks.
- Nur 0-Bits (**`000…000`**) dürfen auch nicht im Host-Anteil vorkommen. Diese Adresse repräsentiert das Netzwerk in Routing-Tabellen.

IP-Adressierung

- Die Adressen **`127.x.y.z`** dürfen nicht für Netzwerke vergeben werden. Die Adresse **`127.0.0.1`** repräsentiert die *Loop-Back-Adresse*: auf jedem Host kann damit der Host selbst adressiert werden.
- Folgende Adress-Bereiche können für "private" Hosts verwendet werden:

Von	Bis
`10.0.0.1`	`10.255.255.254`
`172.16.0.1`	`172.31.255.254`
`192.168.0.1`	`192.168.255.254`

  - IP-Pakete aus diesem Adressbereich werden nicht von Routern übertragen.
  - Diese Adressen können somit im Internet mehrfach verwendet werden, ein Kontakt nach "außen" ist jedoch nicht möglich.

Adressauflösung

Problem der Adressauflösung

- Bisher erkennen wir Rechner nur anhand der IP-Adressen.
- Ein IP-Paket muss jedoch letztendlich über Schicht-2-Mechanismen (z.B. Ethernet) versendet werden.

⇒ Jeder Netzwerkknoten (Router oder Host) braucht damit folgende Tabelle

IP-Adresse	MAC-Adresse
141.75.20.10	**00-01-02-AC-7C-7E**
141.75.20.11	**00-01-02-FE-8C-9F**
...	...

ARP

Address Resolution Protocol (*ARP*)

- Jeder Host führt einen Cache über schon aufgelöste Adressen.
- Adressen, die älter als 15 min sind, und zwischenzeitlich nicht aufgefrischt wurden, werden aus dem Cache gelöscht.
- Steht eine gesuchte IP-Adresse nicht im Cache, wird eine Suchanfrage über Broadcast versendet.
  - Bestandteil der Suchanfrage: gesuchte IP-, eigene IP- und eigene MAC-Adresse, so können andere passive mithörende Hosts diese Adresse cachen.
  - Der gesuchte Host antwortet mit seiner MAC-Adresse.
  - Das nutzt aus, dass viele Schicht-2-Netzwerke physikalisches Broadcast erlauben.

Subnetting

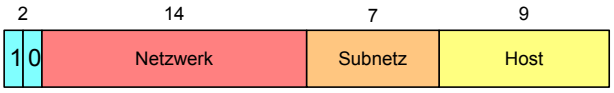
Vergabe von Adress-Blöcken:

- Beispiel: ein Campus möchte 15 Netzwerke mit je 300 Rechnern anmelden
  - 15 Klasse-C-Netze sind zu klein (sie erlauben 254 Hosts).
  - Es müssen daher 15 Klasse-B-Netze beantragt werden.
  - Damit sind schon fast 1 Promille aller Klasse-B-Netze weltweit für einen Campus verwendet.
  - Unschön ist auch, dass der Campus 15 einzelne Netzwerke anmelden muss. Schöner wäre eine einzelne Anmeldung durchzuführen – die weitere Unterteilung in Netze sollte intern geregelt werden können.

$\Rightarrow$ Subnetze

Subnetting

- Alle zusammengehörige Netzwerke werden durch eine Netzwerkadresse repräsentiert.
- Der Host-Anteil wird in einen Anteil *Subnetz* und einen Anteil *Host* aufgeteilt, z.B.



- Alle Subnetze müssen aus Sicht der Topologie eng beieinander legen, da sie von außen durch eine einzige Netzwerkadresse repräsentiert werden.
- Außenstehende Router vermitteln alle Pakete für diese Subnetze zum selben Router (der durch die Netzwerkadresse repräsentiert wird).

Subnetting

Wie wird der Anteil *Subnetz* erkannt?

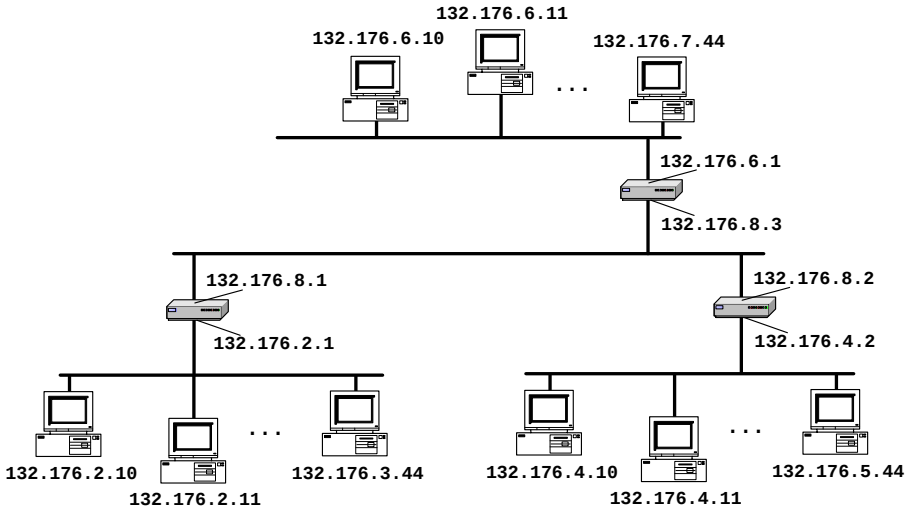
- Jedes Subnetz bekommt eine *Subnetzmaske* zugeteilt
  - 32-Bit, jede hat die Form **`11...100...0`**
  - Darstellung in Oktett-Schreibweise, z.B. **`255.255.255.0`**
  - die 1en repräsentieren die Teile *Netzwerk* und *Subnetz*
  - die 0en repräsentieren den Teil *Host*
- Ein Host soll ein Paket an eine bestimmte IP-Adresse senden:
  - er führt eine logische Und-Verknüpfung mit der Zieladresse und der Subnetzmaske (→ Subnetzadresse)
  - eigenes Subnetz → Übermitteln im eigenen Subnetz
  - fremdes Subnetz → Übergabe an den nächsten Router (genannt *Standard-Gateway*)

Subnetting

Zurück zu unserem Beispiel:

- 300 Hosts/Subnetz → 9 Bits für Host notwendig
- 15 Subnetze → weitere 5 Bits für die Subnetzadresse (alles 0 und alles 1 sind nicht erlaubt)
- Wir beantragen eine Klasse-B-Adresse
  - insg. 16 Bit für Host- und Subnetz-Anteil
  - d.h. wir bekommen mehr als notwendig
  - Wahl: 7 Bit Subnetzadresse, 9 Bit Host
  - damit 126 mögliche Subnetze, 510 mögliche Hosts/Subnetz
  - Subnetzmaske: **`255.255.254.0`**
- Subnetzmasken sind im Subnetz einheitlich, im Netzwerk können sie jedoch unterschiedlich sein.

Subnetting



Problem der zu wenigen IP-Adressen

- Bei der aktuellen IP-Variante (IPv4) gibt es nicht mehr als 2^{32} (ca. 4 Mrd.) mögliche Adresskombinationen.
- In der Realität sind es viel weniger
  - Private Adressen, Multicast-Adressen, Loop-Back
  - Unbenutzbare Kombinationen (mit 0 oder 255)
  - Zusätzlich werden Adressen im Block zugewiesen und viele davon nicht genutzt
- Viele Menschen benutzen mehr als ein Netzwerk-Gerät:
  - Typisch: Desktop, Notebook, Mobiltelefon, PDA, Router, …
  - Konsequenz: es gibt viel zu wenig IP-Adressen
  - Zwei Zwischenlösungen: Klassenlose Adressen und NAT

Klassenlose Adressen

Problem der Klasse-B-Netze:

- Wir benötigen $15 \cdot 300 = 4500$ Adressen und belegen eine Klasse-B-Adressblock von 65 534 Adressen, damit haben wir über 90% der Adressen umsonst belegt.
- Beobachtung: Sehr viele Klasse-B-Netze belegen nur einen Bruchteil der möglichen Adressen.
- Ausweg: man vergibt hinreichend viele Klasse-C-Adressen – in unserem Fall wären mindestens 18 notwendig (nach Umorganisation der Subnetze, $18 \cdot 254 \geq 4500$).
- Dadurch entsteht ein weiteres Problem: außenstehende Router müssen jetzt statt einem Eintrag für dieses Netz 18 Einträge verwenden, um jeden Host zu erreichen.

Klassenlose Adressen

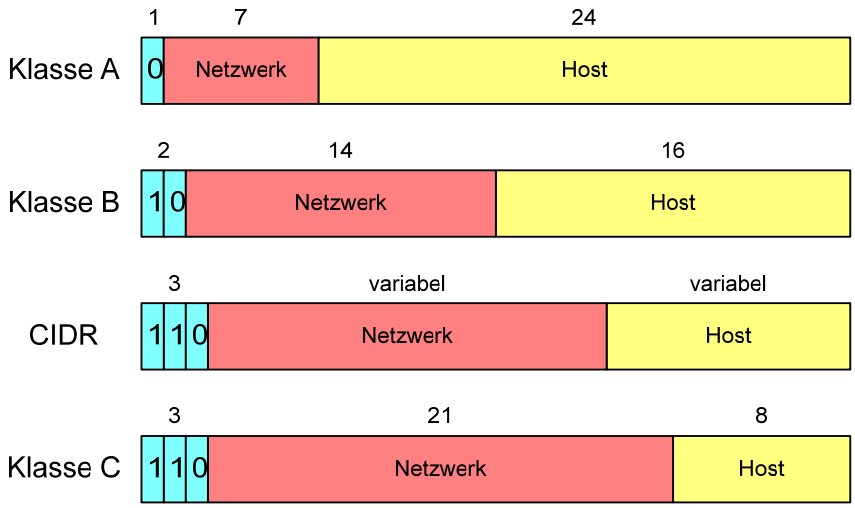
Konkretes Problem:

- Die drei Klassen Adressklassen A, B, C erlauben keine sinnvolle Einstellung der Netzwerk-Host-Anteile für "mittelkleine" Netze.

Ausweg *CIDR* (*Classless Interdomain Routing*):

- Wir vergeben ganze Blöcke von Klasse-C-Adressen mit dem selben Präfix, z.B.
  - **`221.176.128`**, **`221.176.129`**... **`221.176.159`**
  - bei diesen Adressen sind beispielsweise die vorderen 19 Bits gleich

Klassenlose Adressen



Klassenlose Adressen

- Ein Router muss sich jetzt nur die Adresse **`221.176.128`** und zusätzlich die Maske **`255.255.224.0`** (ersten 19 Bits gesetzt) merken, um alle Netze dieses Blocks zu erreichen
  - Schreibweise: **`221.176.128/19`**
- Diese so genannte *CIDR-Aggregation* kann hierarchisch erfolgen, z.B.
  - Internet Provider bekommt Adressen mit 18 gleichen Bits
  - er vergibt Adressen mit 20 gleichen Bits an weitere Unternehmen

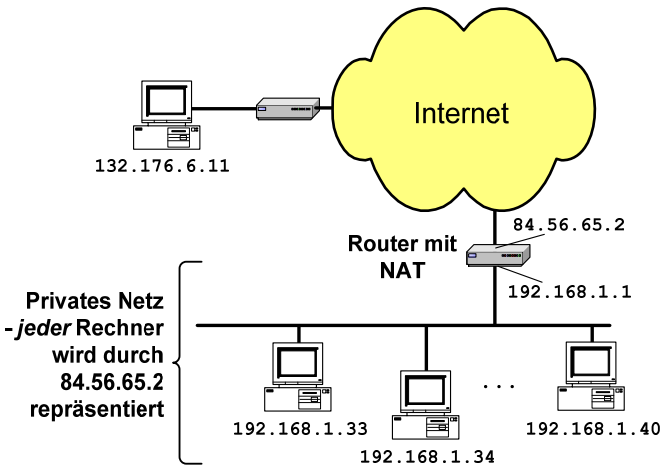
Klassenlose Adressen

- In der Endphase von IPv4 haben klassenlose Adressen die Klassenadressen abgelöst.
- Vorteil: Bei der Vergabe kann man die Größe von Blöcken wesentlich besser dosieren, also nicht nur Blöcke von 16,7 mio., 65 534 oder 254.
- Da die Größe des Netzwerkanteils nicht mehr anhand der ersten Bits erkennbar ist, muss die Größe generell mitgespeichert werden (analog zur Subnetz-Maske).
- Generelle Schreibweise für Adress-Blöcke: **`a.b.c.d/n`**
  - Eine "alte" A-Klasse Adresse hat damit z.B. die Schreibweise

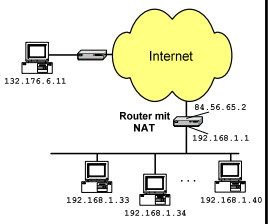
`101.0.0.0/8`

NAT

Zweite Lösung: *NAT* (*Network Address Translation*)

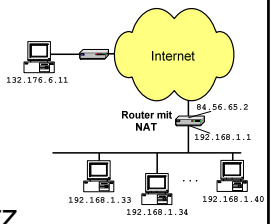


NAT



- Abgetrennte Subnetze bekommen nur eine einzige Adresse zugewiesen (die des NAT-Routers).
- Innerhalb des abgetrennten Netzes werden private Adressen verwendet (die weltweit beliebig oft eingesetzt werden können).
- Problem: Von außen können die Rechner des Subnetzes nicht unterschieden werden. Daher übernimmt der Router zusätzlich die Aufgabe, Pakete an die Router-Adresse im Subnetz zu verteilen.

NAT



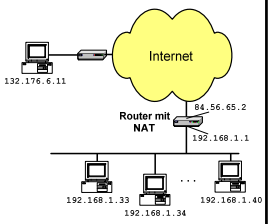
Fall 1: Rechner des Subnetzes kommunizieren untereinander

- In unserem Fall sei die Subnetz-Maske **`255.255.255.0`**
- Pakete können ohne Probleme im Subnetz zugestellt werden.

Fall 2: Kommunikation innen → außen:

- Beispiel: **`192.168.1.34`** sendet an **`132.176.6.11`**
- Subnetz-Maske erfordert Weiterleitung über den Router.
- Paket mit privater Adresse würde von nicht-NAT-Router verworfen werden.
- Der NAT-Router ersetzt die Absender-Adresse durch seine eigene (globale) **`84.56.65.2`** und sendet weiter.

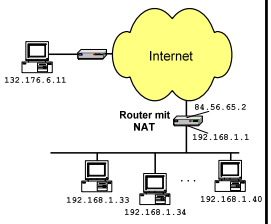
NAT



- **`132.176.6.11`** empfängt Paket und sendet ggfs. Antwort an **`84.56.65.2`** (*nicht* **`192.168.1.34`**!)
- Der Router hat sich gemerkt, dass das ursprüngliche Paket von **`192.168.1.34`** kam und leitet es weiter.
- Der Router muss sich eine Liste aller offenen "Verbindungen" halten.
  - Verbindungen gibt es auf IP-Ebene noch gar nicht
  - Erfordert Wissen von Schicht-4-Protokollen und höher
  - Liste kann u.U. sehr umfangreich werden

NAT

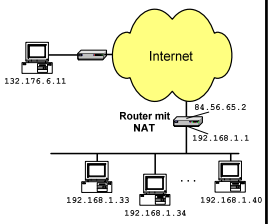
Fall 3: Kommunikation außen -> innen:



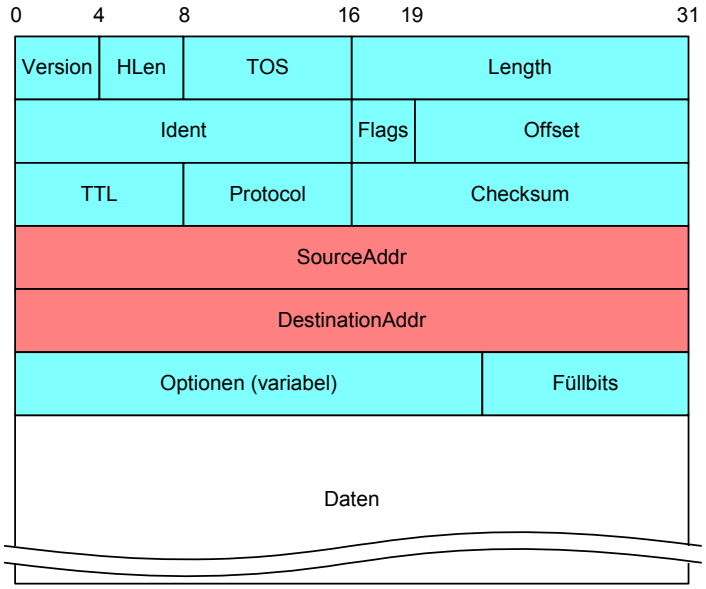
- Typischer Fall: **`192.168.1.34`** ist ein Web-Server.
- **`132.176.6.11`** sendet unaufgefordert ein Paket an **`84.56.65.2`**
- Der Router weiß nicht, dass es für **`192.168.1.34`** gedacht war
  - Vorkonfigurierte Liste im Router notwendig, z.B. alle Pakete, die zu einem Web-Protokoll gehören, werden an Adresse **`192.168.1.34`** weitergeleitet.
  - Protokolle werden anhand der *Port-Nummer* erkannt (siehe später).

NAT

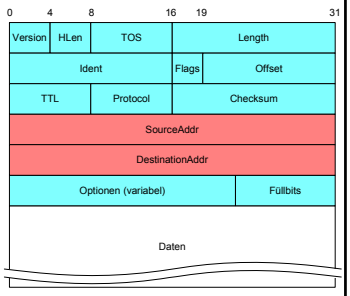
- Kritik an NAT:
  - Router kompliziert
  - Keine eindeutige Trennung mehr zwischen der Vermittlungs- und der Transportschicht
  - Eingebundene Rechner sind keine "vollen" IP-Knoten (Weiterleitung u.U. nicht eindeutig)
  - Ideal wäre, auf IPv6 zu wechseln (siehe später)
- NAT wird erfolgreich bei der Einbindung von Heim-Netzen über DSL verwendet.



Aufbau von IP-Paketen (IPv4)

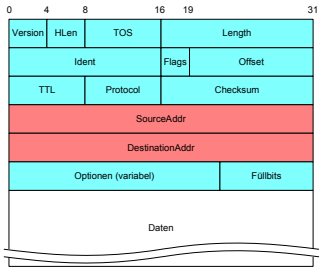


Aufbau von IP-Paketen (IPv4)



- *Version*:
  - 4: IPv4 (die aktuelle)
  - 5: experimentelles Protokoll ST-II
  - 6: IPv6 (die zukünftige)
- *HLen*:
  - Länge des Headers in 32 Bit-Wörtern
- *TOS* (*Type of Service*):
  - Service-Art – beeinflusst das Routing
  - wird nicht häufig verwendet, wurde häufig umdefiniert
- *Length*:
  - Länge des gesamten Pakets in Byte

Aufbau von IP-Paketen (IPv4)

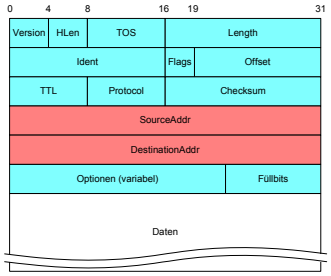


- *Ident*, *Flags*, *Offset*:
  - Fragmentierung und Reassemblierung
- *TTL* (*Time To Live*):
  - maximale Anzahl von Hops, die ein Paket noch weitergeleitet wird
  - verhindert "kreisende" Pakete
- *Protocol*:
  - welches Protokoll (z.B. Transportprotokoll) wird verwendet
  - Bedeutung einiger Nummern:

Protocol	Bedeutung
1	ICMP
4	IP in IP
6	TCP
17	UDP

Aufbau von IP-Paketen (IPv4)

- *Checksum*:
  - Checksumme *des Headers*
- *SourceAddr*:
  - die IP-Adresse des Senders
- *DestinationAddr*:
  - die IP-Adresse des Empfängers
- *Optionen*:
  - weitere Optionen
  - selten benutzt



Aufbau von IP-Paketen (IPv4)

Zur Prüfsumme:

- Ein Router ändert beim jeder Weiterleitung das Feld *TTL:* $TTL_{neu}=TTL_{alt}-1$
- Daher verändert sich auch das Feld *Checksum.*
- Aus der Art der Prüfsummenbildung ergibt sich jedoch: Wenn sich ein Feld des Headers um 1 verringert, erhöht sich die Prüfsumme um 1.
- Viele Router haben daher einfach $Checksum_{neu}=Checksum_{alt}+1$ ausgeführt, ohne die Prüfsumme wirklich zu prüfen.
- In IPv6 gibt es u.a. aus diesem Grund keine Prüfsummenfeld mehr.

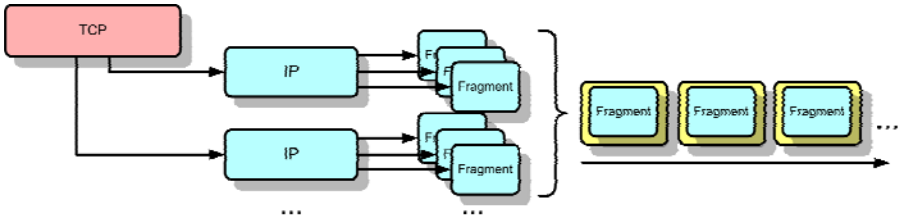
Fragmentierung und Reassemblierung

- IP-Pakete können bis zu 64 kByte groß werden.
- Üblicherweise können Schicht-2-Technologien wesentlich weniger transportieren
  - *MTU* (*Maximum Transmission Unit*) bei Ethernet 1500 Bytes
  - MTU bei Punkt-zu-Punkt-Netzwerk (PPP): 532 Bytes
- Ein Ausweg wäre, die Länge von Daten zu begrenzen, die zusammenhängend mit IP übertragen werden könnten:
  - auf dem Weg zum Ziel müsste man sich an der kleinsten MTU orientieren
  - dieser Wert ist häufig dem Sender nicht bekannt
  - Netzwerke, die auf dem Weg größere Einheiten transportieren könnten, würden durch viele kleine Pakete belastet

Segmentierung und Fragmentierung

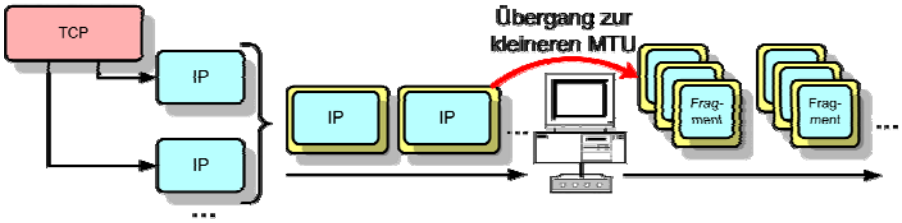
Lösung: Zerlegung der übertragenen Einheiten

- TCP → IP: *Segmentierung*
  - Beliebig große Nachrichten werden in IP-Pakete von max. 64 kByte zerlegt
- IP → Netzwerk: *Fragmentierung*
  - IP-Pakete werden so in *IP-Fragmente* zerlegt, dass sie in Rahmen mit entsprechender MTU passen



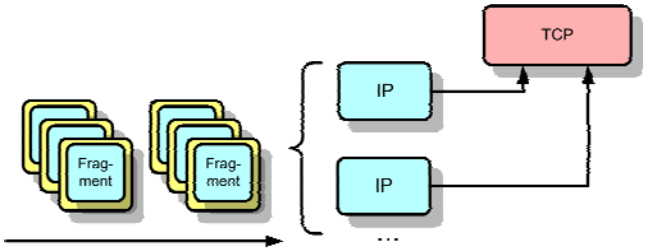
Fragmentierung

- Sinnvollerweise *segmentiert* TCP direkt so, dass im Heimnetzwerk des Senders nicht *fragmentiert* wird.
- Allerdings kennt der Sender nicht alle MTUs zum Ziel.
- Router, die zu Netzwerken mit kleineren MTUs weitervermitteln, müssen daher fragmentieren.

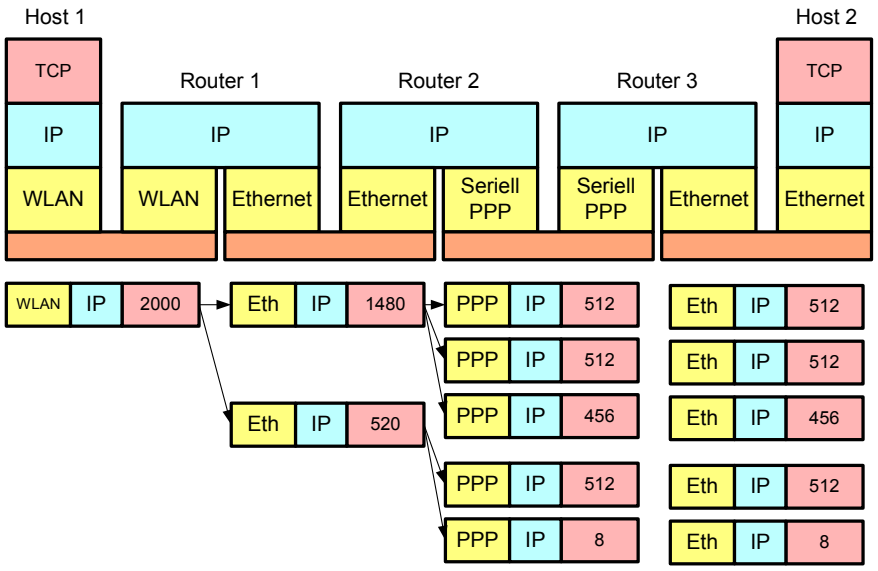


Reassemblierung

- Der Empfänger sammelt alle Fragmente ein und setzt das ursprüngliche IP-Paket wieder zusammen (*Reassemblierung*).
- Es findet keine Reassemblierung durch Router statt
  - Zu hoher Speicherbedarf
  - Fragmente müssen nicht über denselben Router gehen



Fragmentierung und Reassemblierung

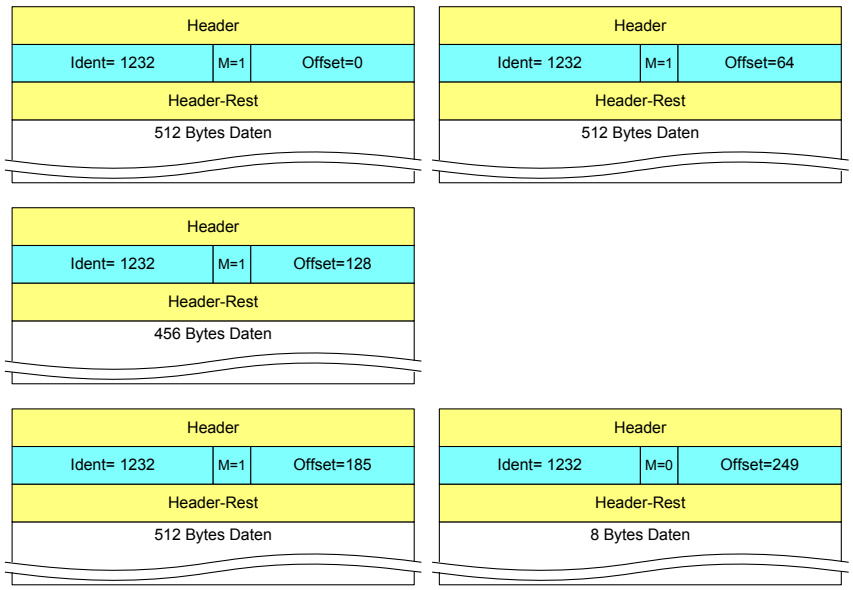


Fragmentierung und Reassemblierung

Kodieren der Fragmente in IP-Paketen:

- *Ident* (16 Bits):
  - eindeutige ID des Pakets (für alle Fragmente gleich)
  - wird vom Sender definiert und muss (für eine bestimmte, lange Dauer) eindeutig sein
- *Flags* (3 Bits):
  - 1. Das erste Flag-Bit ist reserviert
  - 2. *Fragmentierung*: 0: erlaubt, 1: verboten
  - 3. *More*: 1: es folgen weitere Fragmente, 0: letztes Fragment
- *Offset* (13 Bits):
  - Wo liegt dieses Fragment innerhalb der gesamten Nutzdaten? (ausgedrückt in 8-Byte-Blöcken)

Fragmentierung und Reassemblierung



Fragmentierung und Reassemblierung

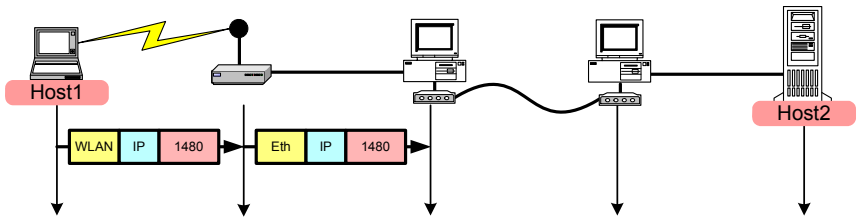
Der Empfänger sammelt zuerst *alle* Fragmente ein, bevor ein IP-Paket an TCP übergeben wird

- Fehlt ein Fragment, so werden alle anderen Fragmente verworfen. Insbesondere wird das Fragment nicht beim Sender nachgefordert.
- Grund: das Nachfordern findet auf TCP-Ebene statt; TCP kennt aber nur *Pakete*, keine *Fragmente.*
- Obwohl ein Fragment vielleicht schon verloren gegangen ist, belasten die existierenden Fragmente noch auf dem Übertragungsweg und beim Empfänger Ressourcen.
- Umgekehrt müssen erfolgreich übertragene Fragmente u.U. nochmals gesendet werden.

Fragmentierung und Reassemblierung

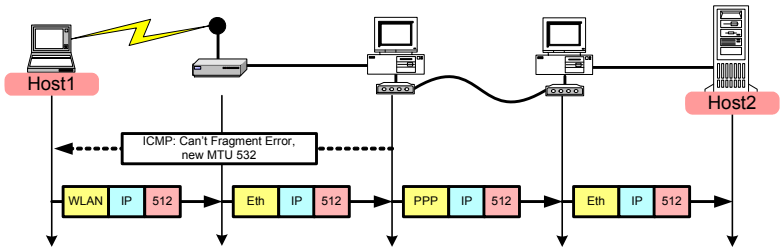
- Jeder Empfänger muss Speicher für Fragmente vorhalten
  - Fragmente können in unterschiedlicher Reihenfolge eintreffen.
  - Es können Fragmente mehrerer IP-Pakete eintreffen, bevor das erste IP-Paket vollständig ist.
- Deshalb wird die Fragmentierung im allgemeinen vermieden.
- Man verwendet statt dessen ein *MTU Path Discovery*: der Sender stellt die kleinste MTU auf dem gesamten Weg zum Empfänger fest und stellt entsprechend kleine IP-Pakete her.

MTU Path Discovery



- Der Sender fragt beim Öffnen einer TCP-Verbindung die MTU-Größe des Zielnetzwerks ab.
- Der Sender bildet die Minimum-MTU {fremde, eigene} und *segmentiert* auf diese Größe.
- IP-Pakete werden als *nicht-fragmentierbar* markiert.

MTU Path Discovery



- Ein Router zu einer kleineren MTU-Größe verwirft das IP-Paket und sendet ein Fehler-Paket (ICMP, siehe später) zum Sender mit der neuen Größenangabe.
- Der Sender segmentiert darauf Nachrichten an diesen Empfänger in kleinere IP-Pakete.
- Nach einer gewissen Zeit setzt er die Größe auf den ursprünglichen Wert zurück (möglich: bessere Route).

Das "Ende" von IPv4

IPv4-Adressen: Abschiedsgrüße, Mahnungen und Pappschilder
Mit einem symbolischen Akt und vielen salbungsvollen Worten wurden am Montag in Miami (US-Bundesstaat Florida) die letzten fünf IPv4-Adressblöcke an die regionalen Internetadressverwaltungen übergeben:
Die Chefs aller fünf Regional Internet Registries (RIR) nahmen bei der Zeremonie kamerataugliche Pappschilder entgegen, auf denen ihr letzter /8-IPv4-Block aus dem nun völlig leeren Pool notiert war. Diese Blöcke mit jeweils rund 16,7 Millionen IPv4-Adressen waren eigens für diese letzte, außerordentliche Zuteilung reserviert worden. Die Sonderzuteilung soll sicherstellen, dass am Ende auch die RIRs noch einmal einen Schwung Adressen bekommen, […]

Quelle: Heise Online, 03.02.2011

IPv6

Eine "richtige" Lösung: IPv6:

- Die IETF (Internet Engineering Task Force) begann sich im Jahr 1991 mit dem Problem des IP-Adressraums zu befassen:
  - 32 Bit sind zu wenig – selbst mit CIDR und NAT
  - erster Vorschlag: 64 Bit – man landete schließlich bei dem Vorschlag, 128 Bit für IP-Adressen vorzusehen
- Da sowieso bei der Umstellung drastische Änderungen zu erwarten sind, befasste man sich noch mit weiteren Themen:
  - Unterstützung von Echtzeit-Diensten
  - Sicherheit
  - Vereinfachte Konfiguration
  - Mobile Hosts

IPv6

- Das Resultat der Überlegungen wurde *IPv6* (*IP Version 6*) genannt, eine alte Bezeichnung lautet *IPnG* (*IP Next Generation*).
- Mittlerweile sind viele Konzepte für die neue IP-Version in die "alte" Version IPv4 eingeflossen (z.B. Mobile IP).
- Der Adressraum von 128 Bit ermöglicht
  - $3{,}4 \cdot 10^{38}$ Rechnerknoten insgesamt
  - $6{,}7 \cdot 10^{23}$ Rechner/Quadratmeter der Erdoberfläche
- Wichtig: es wurde ein Umstellungsplan für den Umstieg von IPv4 auf IPv6 erarbeitet
  - Umstellung kann nicht schlagartig erfolgen
  - Mehrere Jahre (Jahrzehnte?) Parallelbetrieb

IPv6 – Adressierung

- Notation: 8 vierstellige Hex-Zahlen mit ":" getrennt, z.B. **`47CD:1234:4422:AC02:0022:1234:A456:0124`**
- Vereinfachung
  - eine *einzelne* Reihe von **`0000`**-Blöcken kann durch "::" beschrieben werden
  - in jedem Block können führende **`0`**en eingespart werden:

`47CD:0000:0000:0000:0000:0000:A456:0124`
→ **`47CD::A456:124`**

- IPv4-Adressen im IPv6-Adressraum:
  - **`::FFFF:141.75.20.10`** für den IPv4-Host **`141.75.20.10`**
  - zu beachten: der IPv4-Anteil darf in der alten Notation beschrieben werden (d.h. mit 4 *dezimal*-Oktets)

IPv6 – Adressierung

Besondere Adressen/Adressblöcke:

- **`::1`** (**`0…01`**): *Local Loopback*
- **`fe80::/10`** (**`fe80`**… bis **`febf`**…): *Link Local Unicast*, verwendet für Autoconfiguration, keine Router-Weiterleitung
- **`fc00::/7`** (**`fc`**… und **`fd`**…): *Unique Local Unicast*, Private Adressen, keine Router-Weiterleitung
- **`ff00::/8`** (**`ff`**…): *Multicast*

IPv6 – Adressierung

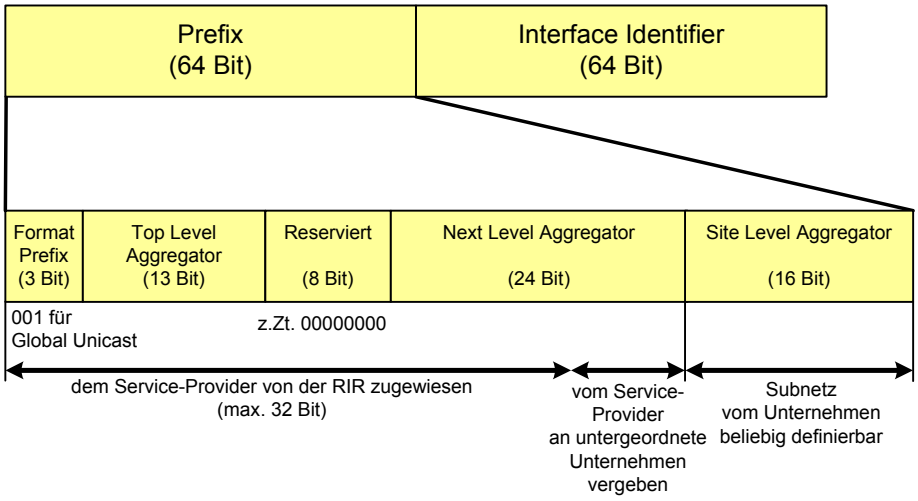
Aufbau von Unicast-Adressen

- 64 Bit *Prefix*: entspricht dem Netzwerk/Subnetz-Anteil aus IPv4
- 64 Bit *Interface Identifier*: entspricht dem Host-Anteil aus IPv4

Organisation von Präfixen:

- *RIRs* (*Regional Internet Registries*) bekommen Adressblöcke zentral durch die IANA zugeteilt.
- Diese verteilen Adressblöcke an die Service Provider.
- Service Provider geben Adressblöcke ggfs. an Unternehmen weiter.
- Unternehmen definieren ggfs. Subnetze.

IPv6 – Adressierung



Zuweisung einer Host-Adresse

Ein Host muss einen Interface Identifier erhalten.

Es gibt jetzt zwei Varianten:

- *Stateful*: über DHCPv6, Site-zentrale Zuweisung durch einen Server
- *Autoconfiguration*: ein Host kann sich selbst einen Interface Identifier zuweisen (*SLAAC, Stateless Address Auto Configuration*)

  Damit wird ein Host ohne DHCP-Server und ohne manuelle Konfiguration ein IPv6-Netzwerkteilnehmer.

Zuweisung einer Host-Adresse

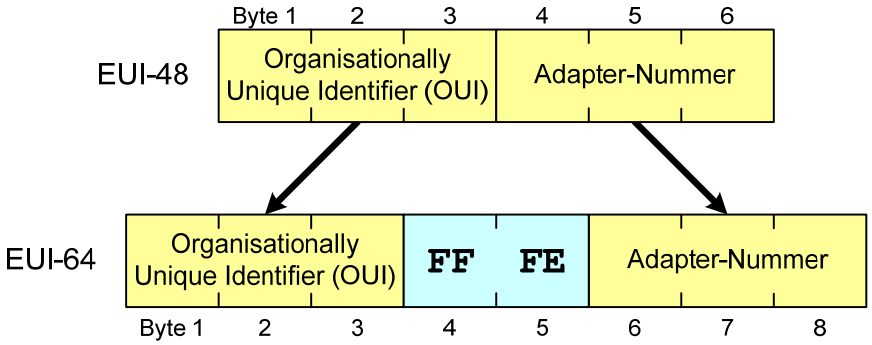
Autoconfiguration:

- Der Host berechnet sich einen Interface Identifier (siehe später).
- Der Host vergibt sich selbst eine Link Local Unicast-Adresse (**`fe80::/64`** + Interface Identifier). Diese wird nur zur Autoconfiguration verwendet.
- Der Host sucht einen Router über Multicast oder wertet Multicast-Ankündigungen des Routers aus.
- Der Router antwortet mit verfügbaren Präfixen.
- Der Host hängt seinen Interface Identifier an.
- Über *Duplicate Address Detection* (*DAD*) prüft der Host, ob die Adresse eindeutig.

Zuweisung einer Host-Adresse

Wie wird eine Interface Identifier berechnet:

- Abbildung der MAC-Adresse auf 64 Bits



Zuweisung einer Host-Adresse

- Das Einfügen von **FF FE** gilt genau genommen nur für EUI-48.
- Bei den "alten" MAC-48-Adressen wird **FF FF** eingefügt (mittlerweile veraltet).

Problem:

- Da die Mac-Adresse sich nie ändert, kann man auf einfache Weise einen Bezug von IPv6-Adresse zu Endgerät herstellen.
- Auch wenn ein Host über verschiedene Netze (d.h. unterschiedliche Präfixe) verbunden ist, bleibt der Bezug zum Host erhalten.

Zuweisung einer Host-Adresse

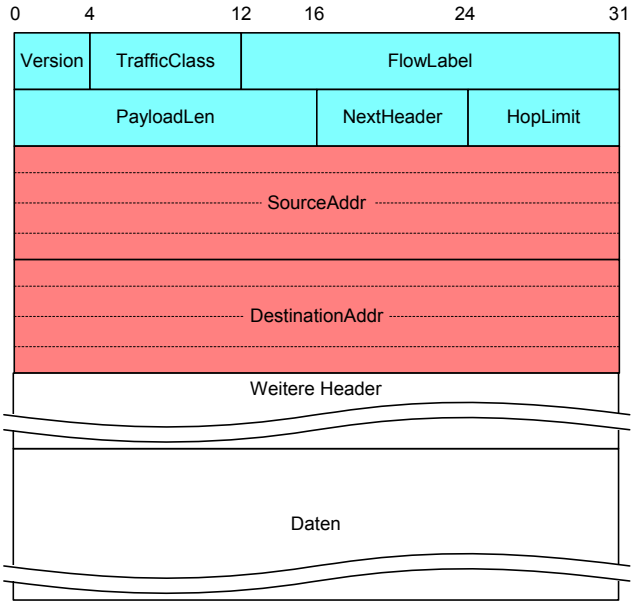
Lösungen:

- DHCPv6: Zuweisen einer neuen Adressen von Zeit zu Zeit.
- Privacy Extension (RFC 4941): Generieren von zufälligen Interface Identifiers.

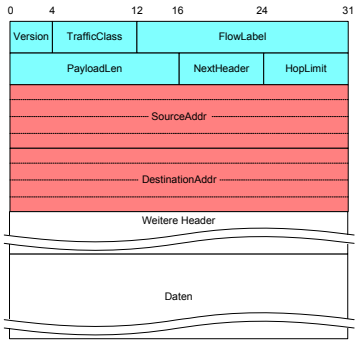
Möglichkeiten bei Privacy Extension:

- Wenn es einen persistenten Speicher im Host gibt: bilde einen neuen Interface Identifier, indem der alte auf einen Hashwert (z.B. MD5) abgebildet wird. Man startet mit einer Zufallszahl.
- Wenn es keinen persistenten Speicher gibt, wird jedes Mal eine "echte" Zufallszahl verwendet.

Aufbau von IP-Paketen (IPv6)



Aufbau von IP-Paketen (IPv6)



- *Version*:
  - bei IPv6: 6
- *TrafficClass*, *FlowLabel*:
  - Dienstgüteunterstützung
- *PayloadLen*:
  - Paketlänge *ohne* Header in Bytes
- *NextHeader*:
  - verwaltet so genannte *Erweiterungsheader*
- *HopLimit*:
  - entspricht dem TTL-Feld aus IPv4

IPv6 – Erweiterungsheader

Erweiterungs-Header in IPv6

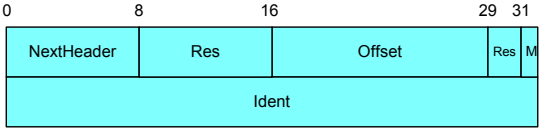
- Diese werden nicht mehr in die Header-Struktur integriert, sondern nach dem Header aufgeführt.
- Das Feld *NextHeader* entspricht entweder dem Feld *Protocol* aus IPv4 (keine weitere Header) oder trägt eine Erweiterungsheader-Nummer. Einige Werte von NextHeader:

NextHeader	Bedeutung
0	Hop-by-Hop Opt.
1	ICMPv4
4	IP in IP
6	TCP
17	UDP
43	Routing

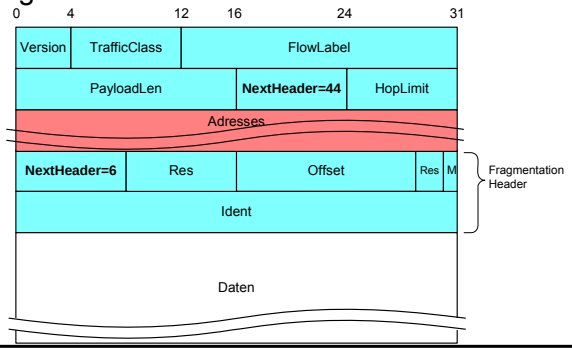
NextHeader	Bedeutung
44	Fragment
50	Security Payload
51	Authentication
58	ICMPv6
59	None
60	Destination Options

IPv6 – Erweiterungsheader

- Beispiel eines Erweiterungsheaders: Fragmentierung



  - Die Bedeutung entspricht weitgehend den IPv4-Feldern
  - Einbettung in das Paket



IPv6 – Fragmentierung

- Bei IPv6 dürfen die Router nicht mehr fragmentieren.
- Beim Übergang auf eine zu kleine MTU sendet ein Router statt dessen eine ICMP-Nachricht an den ursprünglichen Sender.
  - "Packet too big"
- Dieser kann dann
  - direkt kleinere Segmente senden (empfohlen)
  - Fragmente senden
- Die MTU darf in einem IPv6-Netzwerk 1280 Byte nicht unterschreiten.

IPv4-IPv6-Umstieg

Und inzwischen?

- Es wird keinen "Stichtag" geben, an dem die Umschaltung aller IPv4-Rechner nach IPv6 erfolgen kann.
- Die meisten Betriebssysteme unterstützen schon IPv6
- Die Hoffnung: DSL-Provider stellen bis 2013 um
- Parallelbetrieb durch zwei Konzepte:
  - *Dual-Stack*: Rechner können beide Protokollstapel unterstützen.
  - *Tunneling*: Verpacken eines Pakets in der Paketstruktur eines anderen Pakets: z.B. Verpacken eines IPv6-Pakets in einem IPv4-Paket als Nutzlast.
  - *Gateways*: Umwandeln IPv4 $\leftrightarrow$ IPv6 beim Routen (ähnlich NAT).

IPv4-IPv6-Umstieg

Was wird anders:

- Vieles wird einfacher Dank z.B. Autokonfiguration. Achtung: viele Betriebssysteme machen einen Host automatisch zum IPv6-Knoten, ohne dass man das weiß.
- NAT und private Adressen waren zwar nicht aus Gründen der Sicherheit konzipiert, sorgten aber für eine gewisse Abschottung – bei IPv6 ist jeder Host sichtbar.
- Erhebung von Nutzer-Profilen wird erleichtert, wenn Interface Identifier aus der MAC-Adresse gewonnen wird.

Rechnernetze

WS 2012/13

Kapitel 9

Vermittlung im Internet

– Weitere Protokolle der Schicht 3

Kapitel 9: Vermittlung im Internet – Weitere Protokolle der Schicht 3

Ziel: Darstellung weiterer Internet-Protokolle der Vermittlungsschicht.

- ICMP
- DHCP
- Multicast IP

ICMP

Internet Control Message Protocol (*ICMP*):

- IP-Router und -Host teilen dem Sender über ICMP mit, wenn Fehler aufgetreten sind
- ICMP Fehlernachrichten:
  - *Source Quench*: ein Router kann ein IP-Paket wegen Puffer-Überlauf nicht transportieren
  - *Destination Unreachable*: Der Zielhost ist nicht mehr erreichbar (z.B. ausgeschaltet)
  - *Time Exceeded*: TTL-Wert hat 0 erreicht oder Reassemblierung ist durch Timeout fehlgeschlagen
  - *Redirect*: Ein Router kann zwar prinzipiell das Ziel erreichen, es gibt aber eine bessere Route
  - *Can't Fragment*: Fragmentierung notwendig aber nicht erlaubt

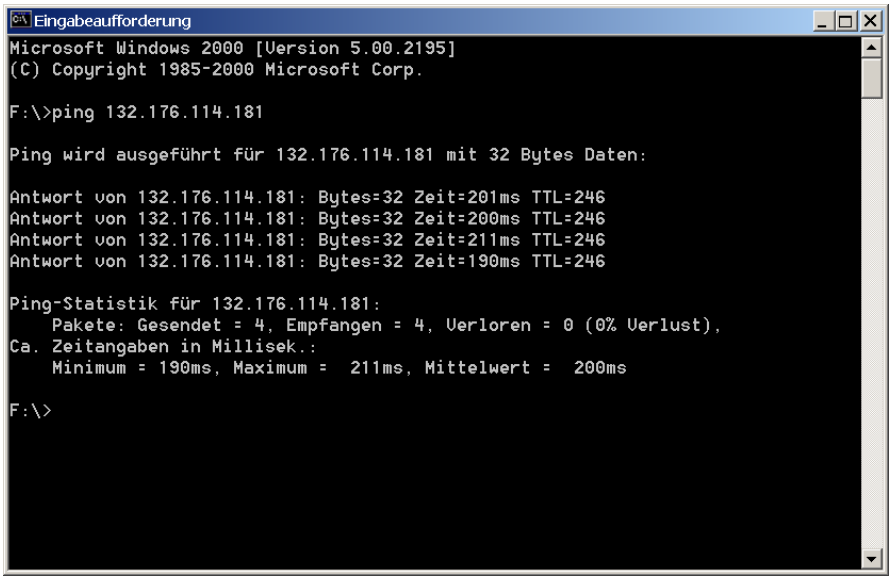
ICMP

- ICMP Informationsnachrichten
  - *Echo Request/Reply*: Jeder Computer, der IP "versteht", sendet auf ein Echo Request ein Echo Reply mit demselben Inhalt.
    (heutzutage oft aus Sicherheitsgründen deaktiviert)
  - *Address Mask Request/Reply*: Anfrage der Subnetzmaske, die durch den Router beantwortet wird.

Werkzeuge, die auf ICMP aufbauen

- *Ping*:
  - testen, ob ein Host erreichbar ist
  - rudimentäre Durchsatz-Statistik
  - basiert auf Echo Request/Reply

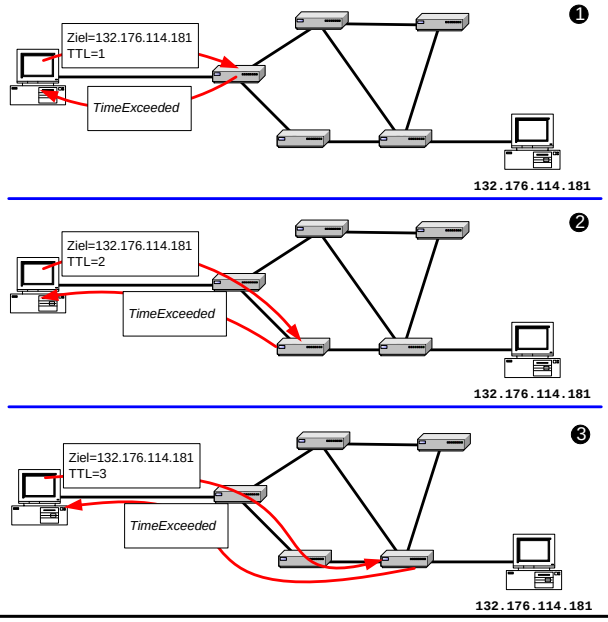
Ping



Traceroute

- *Traceroute*
  - Ermitteln der Route zu einem bestimmten Host.
  - Idee: man sendet nacheinander Pakete zum Zielhost mit den TTL-Werten 1, 2, 3 etc.
  - Der Router, der den TTL-Wert auf 0 dekrementiert, generiert einen ICMP-TimeExceeded-Meldung an den Sender.
  - Der initiierende Host kann aus den ICMP-Meldungen die Route zum Ziel zusammensetzen.

Traceroute



Traceroute

```
Eingabeaufforderung
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

F:\>tracert 132.176.114.181

Routenverfolgung zu cl-www.fernuni-hagen.de [132.176.114.181]  über maximal 30 A
bschnitte:

  1   200 ms   200 ms   210 ms  access1-lns-d.nas.tiscali.de [62.26.136.22]
  2   180 ms   181 ms   190 ms  ge-2-0-0.core0.d.tiscali.de [62.27.92.193]
  3   191 ms   190 ms   180 ms  so-6-0-0.core3.f.tiscali.de [62.27.95.2]
  4   180 ms   190 ms   180 ms  so-1-0-0.fra30.ip.tiscali.net [213.200.64.25]
  5   180 ms   180 ms   191 ms  ir-frankfurt2.g-win.dfn.de [213.200.64.90]
  6   191 ms   180 ms   180 ms  ir-frankfurt2.g-win.dfn.de [213.200.64.90]
  7   191 ms   200 ms   200 ms  cr-essen1-po0-0.g-win.dfn.de [188.1.18.90]
  8   200 ms   210 ms   191 ms  ar-essen1-ge0-0-0.g-win.dfn.de [188.1.86.2]
  9   190 ms   201 ms   190 ms  C65-GWIN.fernuni-hagen.de [132.176.100.1]
 10   191 ms   190 ms   200 ms  cl-www.fernuni-hagen.de [132.176.114.181]

Ablaufverfolgung beendet.

F:\>_
```

Problem der IP-Adressen bei mobilen Rechnern

Ein Teilnehmer in einem Subnetz muss die entsprechende Netz- und Subnetzadresse besitzen. Ansonsten können keine Pakete zugestellt werden.

- Dies gilt auch für mobile Rechner, die ständig zwischen verschiedenen Subnetzen wandern (z.B. Notebooks).
- Mögliche Lösungen:
  - Ein mobiler Rechner bekommt in jedem Subnetz automatisch eine neue Adresse zugewiesen.
    $\Rightarrow$ DHCP
  - Man ändert das Routing im Internet so, dass ein Rechner seine Adresse behalten darf.
    $\Rightarrow$ Mobile IP (wird hier nicht gezeigt)

Problem der IP-Adressen bei mobilen Rechnern

- Vergabe von IP-Adressen
  - Selbst bei kleinen Netzen erfordert die Verwaltung der IP-Adressen einen gewissen Aufwand
  - Doppelte Vergabe von Adressen $\Rightarrow$ Störungen im Netz
  - Unerkannte freie Adressen $\Rightarrow$ Nummernkreis schnell erschöpft

$\Rightarrow$ *Dynamic Host Configuration Protocol* (*DHCP*)

- In viele Betriebssysteme integriert (auch mobile)
- DHCP benutzt UDP $\Rightarrow$ eigentlich Anwendungsschicht, andererseits vergibt DHCP die IP-Adresse $\Rightarrow$ Vermittlungsschicht

DHCP

Funktionsweise von DHCP

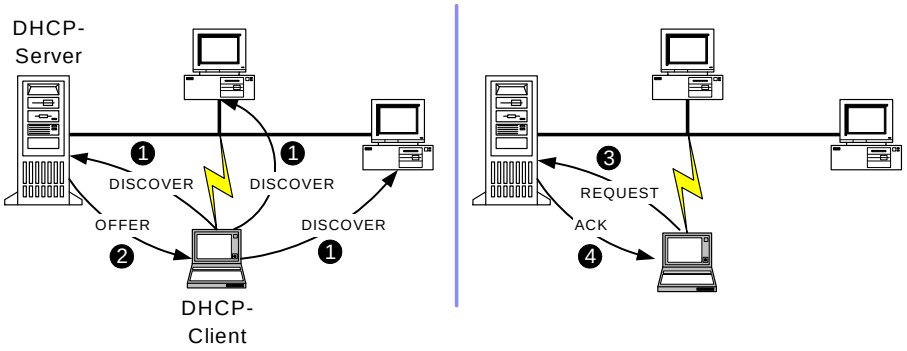
- Ein Rechner, der in ein Subnetz eingebunden wird, sucht selbständig nach einem DHCP-Server
- Dieser teilt ihm eine Netzwerkkonfiguration mit:
  - Eine *freie* IP-Adresse
  - Die *Subnetzmaske*
  - Adresse des *Domain Name Servers*
  - Adresse des *Standard-Gateway*
  - Weitere optionale Parameter (z.B. Adresse des Intranet-Webservers)

DHCP – Adressvergabe

Arten der Adressvergabe:

- *Manuell*: der Administrator hinterlegt vorher eine Liste MAC-Adresse → IP-Adresse
  (sinnvoll, wenn ein bestimmter Host immer dieselbe IP-Adresse erhalten soll).
- *Automatisch*: eine freie IP-Adresse wird automatisch für eine unbegrenzte Zeit vergeben.
- *Dynamisch*: eine freie IP-Adresse wird automatisch für eine bestimmte Zeit vergeben. Vor Ablauf dieser Zeit muss die Reservierung erneuert werden, ansonsten wird die Adresse zurückgezogen
  (ist für mobile Geräte sinnvoll).

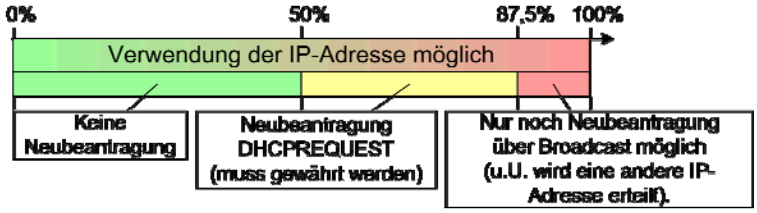
DHCP



DHCP – Prinzip

- Der Client sendet per Broadcast ein DHCPDISCOVER (Achtung: der Client dürfte eigentlich noch gar nicht kommunizieren, da er keine IP-Adresse hat).
- Ein DHCP-Server antwortet mit einem DHCPOFFER und sendet eine mögliche Netzwerkkonfiguration mit.
- Nimmt der Client die Konfiguration an, sendet er an den Server (dessen Adresse jetzt bekannt ist) ein DHCPREQUEST.
- Ist der Server einverstanden, bestätigt er mit DHCPACK und reserviert die Adresse.
- Der Client ist nun im Besitz eines *Leases*, d.h. er darf die IP-Adresse erst einmal nur für eine bestimmte Zeit benutzen.

DHCP – Leases



- Zwei relevante Zeiten
  - T1=50%, T2=87,5% der Lease-Zeit
- Nach T1 versucht der Client den Lease zu erneuern (mit DHCPREQUEST).
- Nach T2 muss ein Client sich per Broadcast an alle DHCP-Server des Netzes wenden.
- Nach der Lease-Zeit kann die Adresse ohne Vorwarnung wieder vergeben werden, d.h. der Client darf die Adresse nicht mehr verwenden.

DHCP – Problem des Clients

Client ohne Netzwerkparameter sind ein Problem:

sie müssen nach DHCP-Servern suchen, haben aber noch keine IP-Adresse

⇒ Wohin soll der Server eine Antwort schicken?

- TCP als zuverlässiges Protokoll steht damit nicht zur Verfügung, da es IP-Adressen benötigt.
- UDP kann verwendet werden, solange aber der Client noch keine IP-Adresse zugewiesen bekam, werden Anfragen und Antworten per Broadcast an alle versendet.
- Das kann genutzt werden: empfängt ein DHCP-Server ein REQUEST, das nicht für ihn gedacht ist, zieht er sein Angebot zurück.

DHCP – Relay Agents

- Oft ist es zu aufwändig, für jedes Subnetz einen DHCP-Server einzurichten.
- Die Suche per Broadcast ist üblicherweise auf ein Subnetz beschränkt.
- *Relay Agents* können Suchanfragen weiterleiten, damit kann der Suchraum für Clients erweitert werden.
- Moderne Router haben die Funktionsweise eines Relay Agents integriert, damit muss kein Rechner explizit eingerichtet werden.

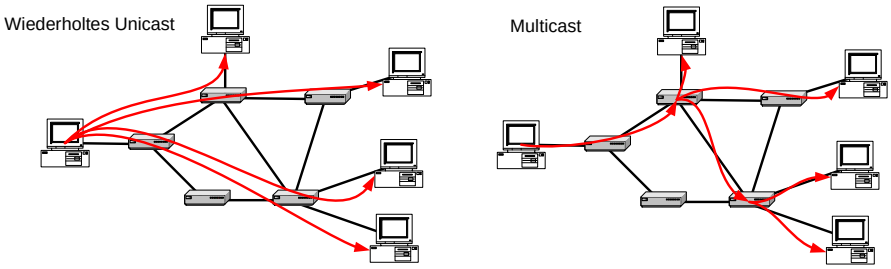
DHCP – Sicherheit

Sicherheitsprobleme

- Subnetze genießen Privilegien. Man möchte den Zugang ungern für jeden mobilen Rechner öffnen.
- Umgekehrt muss auch der mobile Rechner geschützt werden:
  - Falsche Informationen können den Client in seiner Funktionsweise beeinträchtigen.
  - Schlimmer: durch eine falsche Gateway-Adresse könnte der gesamte Nachrichtenverkehr abgehört werden.
- Es hat sich noch kein Sicherheitsstandard für DHCP durchgesetzt.
- Vorschlag: gegenseitige Authentifikation (aber nicht anhand der MAC-Adresse).

Multicast

- Häufig sollen dieselben Pakete an eine ganze Gruppe von Empfängern gesendet werden.
- Beispiele: Internet Radio, Video-Übertragung
- Möglich: man sendet dasselbe Paket an jeden Empfänger einzeln (in einer Schleife) – Nachteil: einige Transportwege müssen denselben Inhalt mehrfach transportieren.

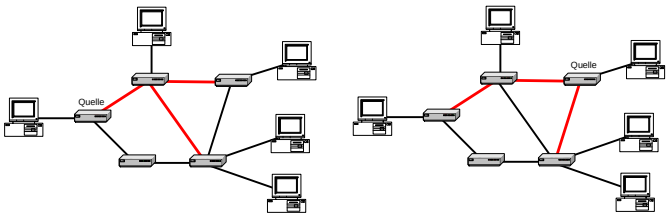


Multicast

- Lösung durch spezielles Multicast-Protokoll: *Multicast IP*
- Grundidee:
  - Hosts, die eine Multicast-Übertragung empfangen möchten, treten einer Multicast-Gruppe bei (*Join*). Der Beitritt wird über den lokalen Router abgewickelt.
  - Im Internet werden Multicast-Gruppen durch eine Klasse-D-Adresse repräsentiert.
  - Der Sender sendet ein Paket an eine Multicast-Gruppenadresse – die Routing-Infrastruktur sorgt dafür, dass alle teilnehmenden Hosts das Paket empfangen.
  - Ausnutzung von nativen Broadcast-Fähigkeiten des Schicht-2-Netzwerks, z.B. WLAN, Ethernet.

Multicast

- Problem: wie verwaltet man alle teilnehmenden Hosts?
  - dezentrale Verwaltung (kein zentraler Server)
  - dynamisch Gruppen: Hosts treten dauernd bei oder aus
- Beispiel: Link-State-Multicast
  - Wenn Router ihre Link-States im Netz verteilen, senden sie auch die Gruppen-Adressen von angemeldeten Hosts.
  - Ein Router kann für jede Gruppen-Adresse mit dem Algorithmus von Dijkstra einen Quellbaum generieren.



Multicast

- Weitere Möglichkeit: Distance-Vector-Multicast
  - Grundidee: man flutet das Netzwerk mit einer Multicast-Nachricht.
  - Man stellt sicher, dass nicht alle Netzwerke (weltweit) in das Fluten einbezogen werden.
- Begriff *MBone* (*Multicast Backbone*):
  - Zusammenschluss von Multicast-fähigen Netzwerken im Internet
  - Überbrücken von Teilstrecken ohne Multicast-fähige Router über IP-Tunneling (verpacken von IP-Paketen als Nutzlast anderer IP-Pakete)
  - Umfasst tausende von Netzwerken; wird zu Video-Konferenzen oder Video-Broadcasting verwendet

Rechnernetze

WS 2012/13

Kapitel 10

Transportprotokolle des Internets

Kapitel 10: Transportprotokolle des Internets

Ziel: Möglichkeiten von kommunizierenden Anwendungen auf der Transportschicht des Internets.

- Transportprotokolle UDP, TCP
- Fluss- und Überlastkontrolle
- Aufbau von TCP-Nachrichten
- TCP-Verbindungsaufbau

Neue Anforderungen

Was bisher geschah...

- Wir können IP-Pakete von einem Host über ein Vermittlungsnetzwerk zu einem weit entfernten anderen Host übertragen, ungeachtet der jeweiligen Schicht-2-Netzwerke zwischen den Hosts.
- Die Übertragung auf jedem einzelnen Hop ist jeweils durch ein Schicht-2-Protokoll gesichert, deshalb dürften eigentlich keine Pakete verloren gehen.

Dennoch:

- Pakete können durch Router-Überlastung verloren gehen.
- Die Reihenfolge der Pakete kann sich durch unterschiedliche Übertragungswege verändern.

Neue Anforderungen

Wichtige Anforderungen aus der Sicht einer Anwendung (bzw. eines Anwendungsentwicklers)

- Garantierte Übertragung von Nachrichten
- Gleiche Reihenfolge beim Empfänger wie beim Sender
- Unterstützung beliebig großer Nachrichten
- Unterstützung mehrerer Anwendungsprozesse
- Flusskontrolle, d.h. Sender stimmt Datenrate auf Empfänger ab
- Überlastkontrolle, d.h. Sender stimmt die Datenrate auf das Router-Netzwerk ab

Neue Anforderungen

Wie erfüllen die Internet Transportprotokolle diese Anforderungen:

Anforderung	UDP	TCP
Garantierte Übertragung	Nein	Ja
Reihenfolge	Nein	Ja
Beliebig große Nachrichten	Nein	Ja
Mehrere Anwendungsprozesse	Ja	Ja
Flusskontrolle	Nein	Ja
Überlastkontrolle	Nein	Ja

Anwendungsmultiplexing

Unterstützung mehrerer Anwendungsprozesse pro Host:

- Man erweitert IP-Pakete um die Funktionalität zu *demultiplexen,* d.h. um verschieden Prozesse auf einem Host zu adressieren.
- Denkbar: man identifiziert einen Empfängerprozess anhand seiner Prozess-ID, aber
  - nicht beim Sender bekannt
  - ändert sich beim Neustart der Anwendung
- Statt dessen: man verwendet eine 16-Bit lange *Portnummer.*
- Die empfangende Anwendung *bindet* sich an eine Portnummer und kann so die Nachrichten für diesen Port empfangen.

Ports

Problem: wie weiß der Sender, an welchem Port sich eine Anwendung befindet?

- *Well-known-Ports*: für wichtige Dienste gibt es bekannte Portnummern, z.B.

Port	Dienst	Port	Dienst	Port	Dienst
20,21/tcp	FTP	25/tcp	SMTP	80/tcp	HTTP
22/tcp	ssh	53/tcp/udp	DNS	119/tcp	NNTP
23/tcp	telnet	69/udp	tftp	513/udp	who

- Ansonsten verwendet die Anwendung einen freien Port und kodiert diesen fest
- oder die Anwendungen vereinbaren über einen festen Port, über welchen Port sie weiter arbeiten sollen
- oder die Anwendung testet einige Ports, bis sie verbunden wird

Ports

Mittlerweile sind Ports bis 50000 von der IANA definiert

```
gadugadu        8074/udp    Gadu-Gadu
#                           Marcin Gozdalik <gozdal@gadu-gadu.pl> May 2006
#               8075-8079   Unassigned
http-alt        UUUU/tcp    HTTP Alternate (see port UU)
http-alt        8080/udp    HTTP Alternate (see port 80)
#                           Stephen Casner <casner@precpt.com>
sunproxyadmin   8081/tcp    Sun Proxy Admin Service
sunproxyadmin   8081/udp    Sun Proxy Admin Service
#                           Arvind Srinivasan <arvind.srinivasan@sun.com> August 2005
us-cli          8082/tcp    Utilistor (Client)
us-cli          8082/udp    Utilistor (Client)
us-srv          8083/tcp    Utilistor (Server)
us-srv          8083/udp    Utilistor (Server)
#                           Andy Brewerton <andy.brewerton@otium.co.uk> August 2005
#               8084-8085   Unassigned
d-s-n           8086/tcp    Distributed SCADA Networking Rendezvous Port
d-s-n           8086/udp    Distributed SCADA Networking Rendezvous Port
#                           Gary Hampton <GNG2NTI.com> 27 February 2008
simplifymedia   8087/tcp    Simplify Media SPP Protocol
simplifymedia   8087/udp    Simplify Media SPP Protocol
#                           Emmanuel Saint-Loubert <emmanuel@simplifymedia.com> 08 August 2008
radan-http      8088/tcp    Radan HTTP
radan-http      8088/udp    Radan HTTP
#                           Previous contact: Steve May <Steve.May@uk.radan.com> April 2002
#                           Current contact: Steve May <SteveMay@planit.com> 13 June 2008
#               8089-8090   Unassigned
jamlink         8091/tcp    Jam Link Framework
#                           Evgeniy Filatov <evgeniyfilatov@yandex.ru> 25 November 2009
#               8091/udp    Reserved
#               UU92-UU9b   Unassigned
sac             8097/tcp    SAC Port Id
sac             8097/udp    SAC Port Id
#                           Girish Bhat <gbhat@cranitc.com> April 2006
#               8098-8099   Unassigned
xprint-server   8100/tcp    Xprint Server
xprint-server   8100/udp    Xprint Server
#                           John McKernan <John.McKernan@sun.com>
ldoms-migr      8101/tcp    Logical Domains Migration
#                           Liam Merwick <ldoms-iana-ports@sun.com> 16 January 2009
#               8101/udp    Reserved
#               8102-8114   Unassigned
mtl8000-matrix  8115/tcp    MTL8000 Matrix
mtlUUUU-matrix  U11b/udp    MTLUUUU Matrix
#                           David Pinch <dpinch@mtl-inst.com> April 2002
cp-cluster      8116/tcp    Check Point Clustering
cp cluster      8116/udp    Check Point Clustering
#                           Roni Moshitzky <rmosh@checkpoint.com>
```

http://www.iana.org/assignments/port-numbers

UDP

User Datagram Protocol (*UDP*):

- Telegramm-orientiert, d.h.
  - Übertragung nur in eine Richtung
    (eventuelle Rücksendung ist ein eigener Vorgang)
  - keine Zustellungsgarantie
  - keine Quittung – der Sender weiß nicht, ob ein Paket angekommen ist
- UDP ist geeignet für bestimmte Anwendungen:
  - verbindungslos
  - Nachrichten dürfen verloren gehen

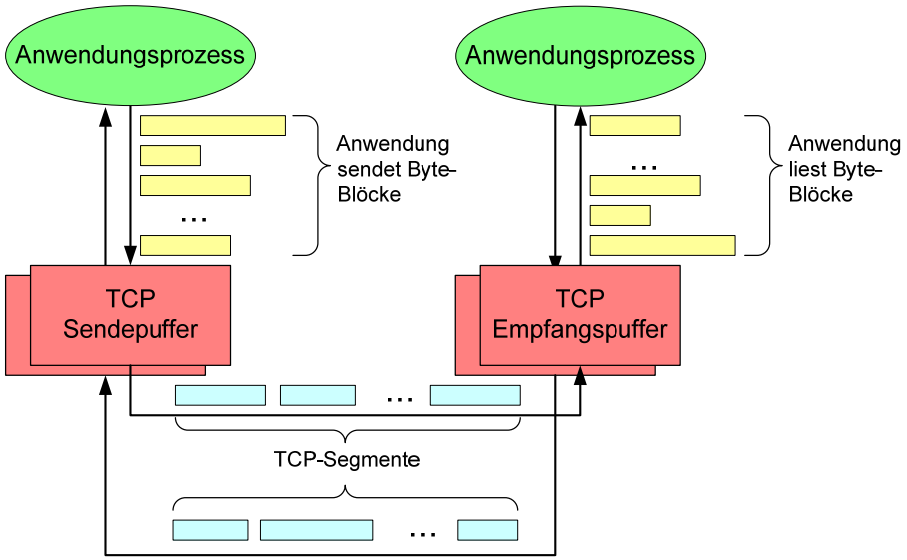
$\Rightarrow$ Z.B. Audio- oder Video-Übertragung, Suchanfragen im lokalen Netz

TCP

TCP (*Transmission Control Protocol*):

- Multiplexen verläuft analog zu UDP
- Zuverlässige, bidirektionale Byte-Streams
  - Kumulative Quittungen, Sliding Windows
  - Pakete in falscher Reihenfolge werden gepuffert, bis die Sendereihefolge hergestellt werden kann.
- Fluss- und Überlastkontrolle
  - Empfänger sendet auf dem Rückkanal die maximale Anzahl der Bytes bis zum Pufferüberlauf.
  - Sender tastet sich an die Überlastsituation (Pufferüberlauf der Router) heran.

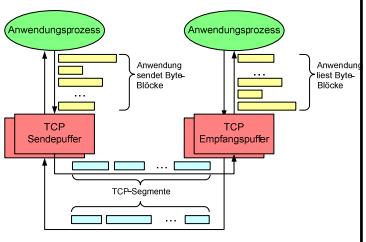
TCP



TCP

Wie werden TCP-Segmente aus dem Byte-Strom gebildet?

- Es wurden genug Bytes gepuffert, um ein TCP-Segment ohne IP-Fragmentierung zu versenden.
- Die Anwendung wünscht explizit, die bisher gepufferten Bytes zu versenden (*flush*).
- Periodischer Timer



TCP verwendet das Sliding-Window-Verfahren

- bekannt aus der Schicht-2 zur Sicherung der Punkt-zu-Punkt-Übertragung

TCP

Unterschiede von Sliding Window auf Schichten 2 und 4:

- Schicht 4 erfordert Verbindungsauf- und -abbau, während bei der Schicht-2-Verbindung zwischen zwei Computern die Verbindung implizit definiert ist.
- Die Round-Trip-Zeit schwankt bei Schicht 4 sehr stark, während sie auf Schicht 2 im Wesentlichen konstant ist.
- Auf Schicht 4 können Pakete umgeordnet werden – bei Schicht-2-Verbindungen ist das (meistens) nicht möglich.
- Ressourcen, insb. Puffergrößen, sind variabel auf Schicht 4 und können sich zur Laufzeit ändern.
- Überlastungen auf Schicht 4 betreffen Router – auf Schicht 2 betrifft die Überlastung lediglich eine Direktverbindung, wird somit auf Senderseite erkannt.

TCP

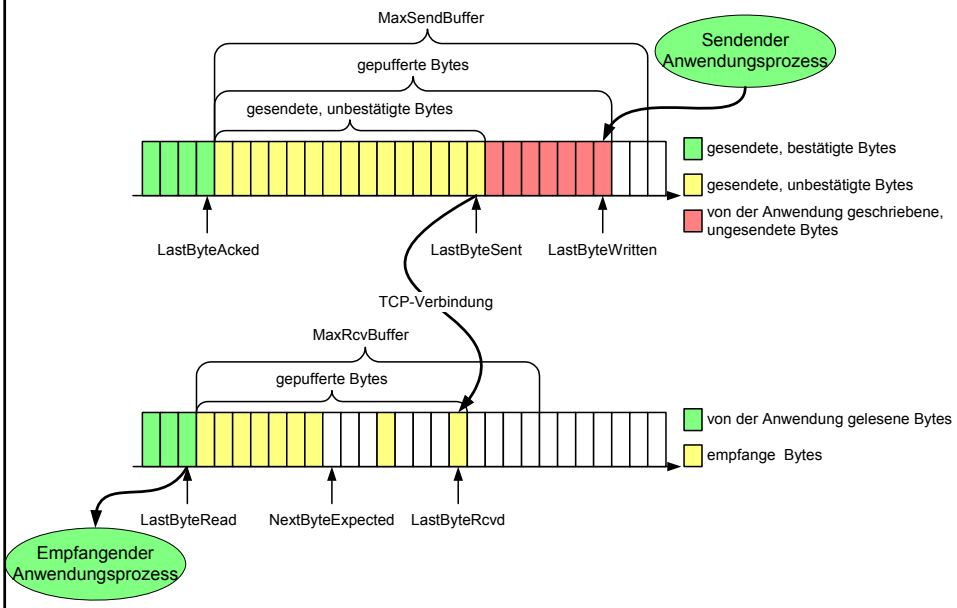
Konsequenz:

- TCP muss viel über die Verbindung *zur Laufzeit* lernen, während auf Schicht-2 die Optimierungen a priori vorgenommen werden können.

Realisierung des TCP-Sliding-Window-Verfahrens:

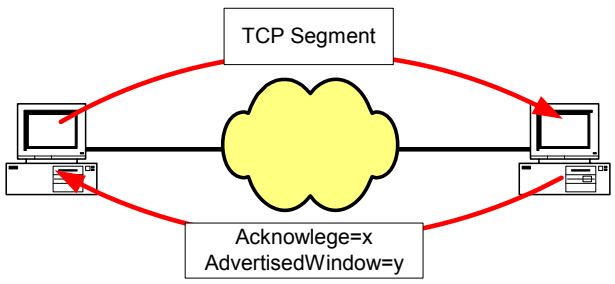
- TCP-Sender und -Empfänger verwenden Buffer.
- Die sendende Anwendung schreibt i.d.R. unblockierend in den Sendepuffer.
- Analog kann die empfangende Anwendung unblockierend aus dem Empfangspuffer lesen, wenn Daten vorliegen.

TCP – Sliding Window



TCP – Flusskontrolle

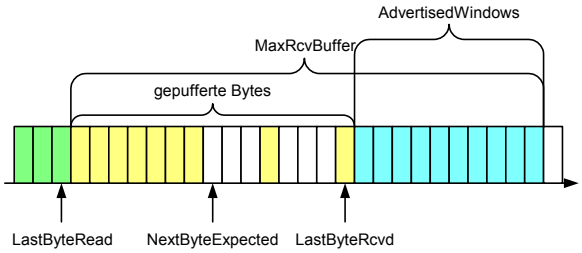
- Der Empfänger sendet bei jedem empfangenen TCP-Segment eine Antwort mit
  - *Acknowledge*: bis zu welchem Byte wurde eine ununterbrochene Folge empfangen (kumulatives ACK)
  - *AdvertisedWindow*: wie viele Bytes kann der Empfänger noch empfangen, bevor der Puffer überläuft



TCP – Flusskontrolle

- Der Empfänger berechnet nach jedem Empfang

AdvertisedWindow = MaxRcvBuffer – (LastByteRcvd–LastByteRead)



- So viele Bytes kann der Empfänger noch puffern, wenn die Anwendung nicht weitere Bytes konsumiert.

TCP – Flusskontrolle

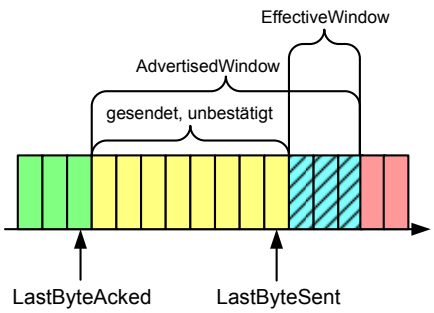
- Der Sender darf nur soviel senden, damit gilt

LastByteSent – LastByteAcked ≤ AdvertisedWindow

anders ausgedrückt: er kann noch

EffectiveWindow=AdvertisedWindow – (LastByteSent – LastByteAcked)

senden



TCP – Flusskontrolle

- EffectiveWindow kann auch 0 sein
  - Der Sender darf erstmal nichts senden.
  - Schreibt die sendende Anwendung ständig weiter in den Puffer, wird dieser irgendwann voll.
  - Ist der Sendepuffer voll, wird die Anwendung beim nächsten Schreiben solange blockiert, bis wieder Pufferplatz frei ist.
- Problem: neue AdvertisedWindow-Werte kommen nur vom Empfänger als Antwort auf ein TCP-Segment
  - Der Sender versucht periodisch, ein einziges Byte zu senden (auch wenn er eigentlich nicht dürfte), bis EffectiveWindow>0
  - Grundregel: *smart sender/dump receiver*

TCP – Neuübertragung

- Geht ein TCP-Segment verloren, sollte es möglichst schnell erneut gesendet werden.
- Grundlage zur Berechnung des Timeouts: die *Round-Trip-Zeit* (*RTT*) zwischen Sender und Empfänger. Hierzu wird die Zeit zwischen Senden eines TCP-Segments und Eintreffen des betreffenden ACKs gemessen.
  - Erster Ansatz: man berechnet einen gleitenden Durchschnitt vergangener RTT-Zeiten

    $RTT = RTT \cdot a + RTT_{last} \cdot (1 - a) \quad (a=0.8...0.9)$

    und setzt den Timeout auf 2·RTT
  - Problem: sendet man ein TCP-Segment zweimal, kann man das ACK nicht mehr der Sendung zuordnen
    $\Rightarrow$ in solchem Fall setzt man die Messung aus

TCP – Neuübertragung

- Weiteres Problem: die 2·RTT sind häufig zu großzügig – hat man eine weitgehend konstante RTT, so wird in der Regel zu lange gewartet bis ein Segment neu übertragen wird.
- Lösung von Jacobson/Karels
  - RTT wird wie bisher berechnet
  - Zusätzlich wird der gleitende Durchschnitt der *Abweichung* von RTT und letzter Messung berechnet

$$RTT = RTT \cdot a + RTT_{last} \cdot (1 - a) \quad (a=7/8)$$
$$Deviation = Deviation \cdot a + |RTT_{last} - RTT| \cdot (1 - a) \quad (a=7/8)$$
$$Timeout = RTT + 4 \cdot Deviation$$

TCP – Überlastkontrolle

- Bisher wurde nur die Flusskontrolle betrachtet – ein Empfänger kann die Datenrate beim Sender so drosseln, dass der Empfangspuffer nie überläuft.
- Notwendig ist jedoch auch eine *Überlastkontrolle* (*Congestion Control*): diese stellt sicher, dass die Übertragungswege, insb. die Router nicht überlastet werden.
- Problem: im Gegensatz zum Empfänger geben die Router eine drohende Überlastung nicht an.

$\Rightarrow$ Der Sender passt sich dynamisch an eine Überlastsituation an und berechnet ständig einen Wert *CongestionWindow*

TCP – Überlastkontrolle

- Anpassung der Formel für das *EffectiveWindow* (Anzahl Bytes, die noch unbestätigt versendet werden dürfen).

```
MaxWindow = min(CongestionWindow, AdvertisedWindow)
EffectiveWindow = MaxWindow – (LastByteSend–LastByteAcked)
```

- Der Wert CongestionWindow (Überlastfenster) gibt an, wie viele Bytes unbestätigt versendet werden dürfen, bis eine Überlastung droht.
- Berechnung von CongestionWindow über *Additive Increase/Multiplicative Decrease*
  - in den folgenden Ausführen nehmen wir vereinfachend die Anzahl der Pakete, statt die Anzahl der Bytes

TCP – Überlastkontrolle

- Start: `CongestionWindow = 1`
- Wenn alle Pakete im CongestionWindow positiv bestätigt wurden:

```
CongestionWindow = CongestionWindow + 1
```

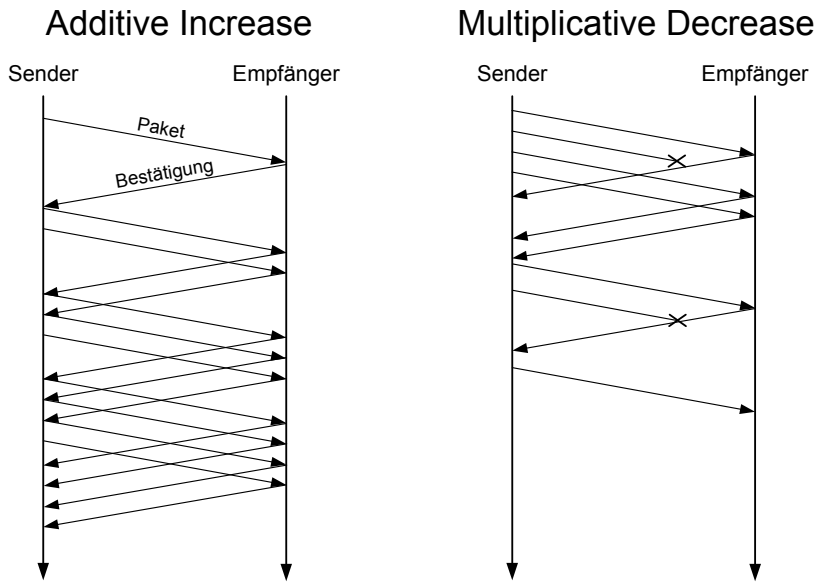
(*Additive Increase*)

- Wenn ein Paket im CongestionWindow verloren gegangen ist (erkannt durch Timeout):

```
CongestionWindow = CongestionWindow / 2
```

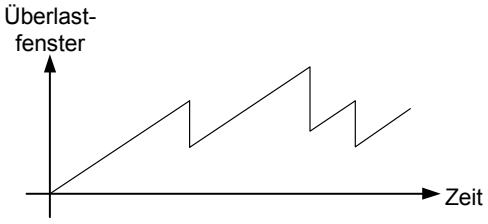
(*Multiplicative Decrease*)

TCP – Überlastkontrolle



TCP – Überlastkontrolle

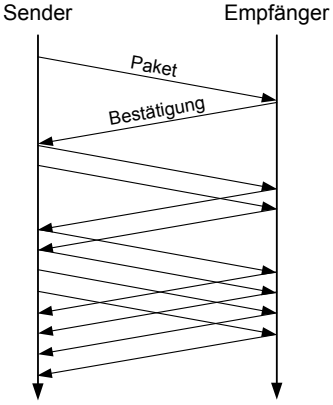
- Der Sender tastet sich langsam an die Überlastgrenze heran und reduziert dann drastisch das Überlastfenster bei Überlastung:
  - typische TCP-Verbindung



  - Warum die drastische Senkung? Überlast ist viel schlimmer als ein zu kleines EffectiveWindow. Würde man die Überlast durch *Additive Decrease* verringern, würde die Überlastsituation kaum beseitigt werden.

TCP – Überlastkontrolle, Slow Start

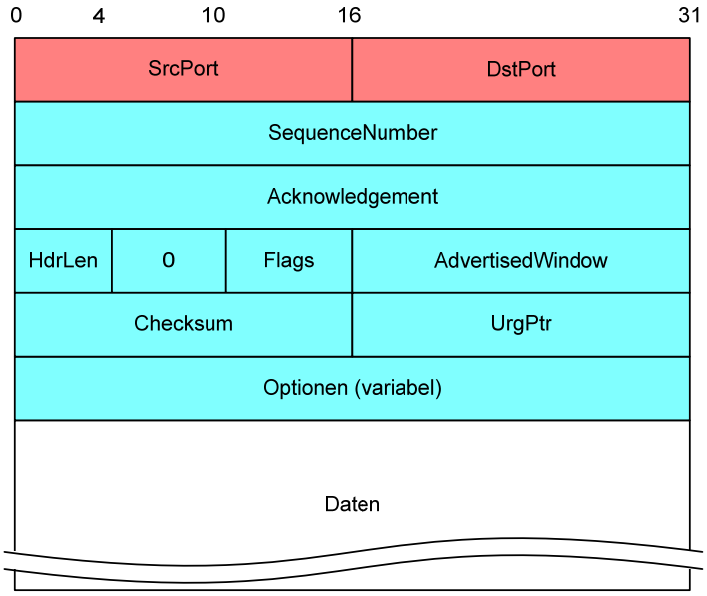
- Am Anfang einer Sitzung ist das Additive Increase zu langsam.
- *Slow Start* erhöht das Überlastfenster multiplikativ – bei Erfolg wird es verdoppelt.
- Slow Start wird angewendet
  - direkt nach dem Sitzungsaufbau
  - jedes Mal, nachdem eine Sitzung "abgestorben" ist, d.h. die Datenrate auf 0 gesunken ist
- Warum "slow"?
  - Historisch: Vorher verwendete man nur das AdvertisedWindow des Empfängers – Slow Start ist da viel langsamer



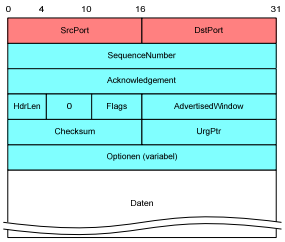
TCP – Überlastkontrolle – Fast Retransmission

- Der Sender kann durch Timeout feststellen, dass ein Segment nicht zugestellt wurde.
- Andere Möglichkeit: Auswerten der kumulativen ACKs zu *nachfolgenden* Segmenten.
- Beispiel: wird 1, 2, 3, 4, 5 gesendet und kommt ACK(1), ACK(2), ACK(2), ACK(2) zurück, so ging wahrscheinlich Segment 3 verloren
  - die ACK(2) heißen hier *Duplikat-ACKs*
  - TCP *Fast Retransmission* wartet drei Duplikat-ACKs ab, bis es das Segment erneut sendet
  - Warum nicht nur zwei Duplikat-ACKs? Das Segment könnte ja auf einem "langsamen Weg" unterwegs sein

Aufbau von TCP-Nachrichten

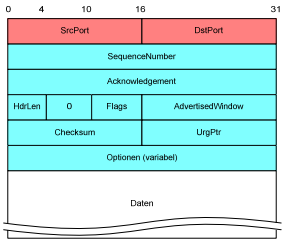


Aufbau von TCP-Nachrichten



- *SrcPort, DstPort*: Felder für das Multiplexen. Zusammen mit den IP-Adressen des IP-Pakets (*SourceAddr*, *DestinationAddr*) wird damit eindeutig eine Verbindung identifiziert.
- *SequenceNumber*: Byte-Nummer der Nutzdatenbytes im gesamten Datenstrom.
- *Acknowledgement*, *AdvertisedWindow*: Felder zur Flusskontrolle in die Rückrichtung .
- *HdrLen*: Länge des TCP-Headers in 32-Bit-Blöcken.
- *Checksum*: Prüfsumme von Header und Daten.

Aufbau von TCP-Nachrichten



- *Flags*:
  - SYN: Aufbau einer Verbindung
  - FIN: Normales Beenden einer Verbindung
  - RESET: Aktives Beenden der Verbindung wegen Problemen
  - PUSH: Daten sollen sofort unter Umgehung des Empfangspuffers an die Anwendung gesendet werden
  - URG: Selten verwendet: die Daten, auf die *UrgPtr* zeigt, müssen sofort von der Anwendung gelesen werden (analog zu einem Interrupt)
  - ACK: Das Feld *Acknowledgement* wird verwendet
- *UrgPtr*: Wenn URG in Flags gesetzt: dringende Daten
- *Optionen*: unbestimmt – Längen sind Vielfache von 32 Bit

TCP-Pseudo-Header

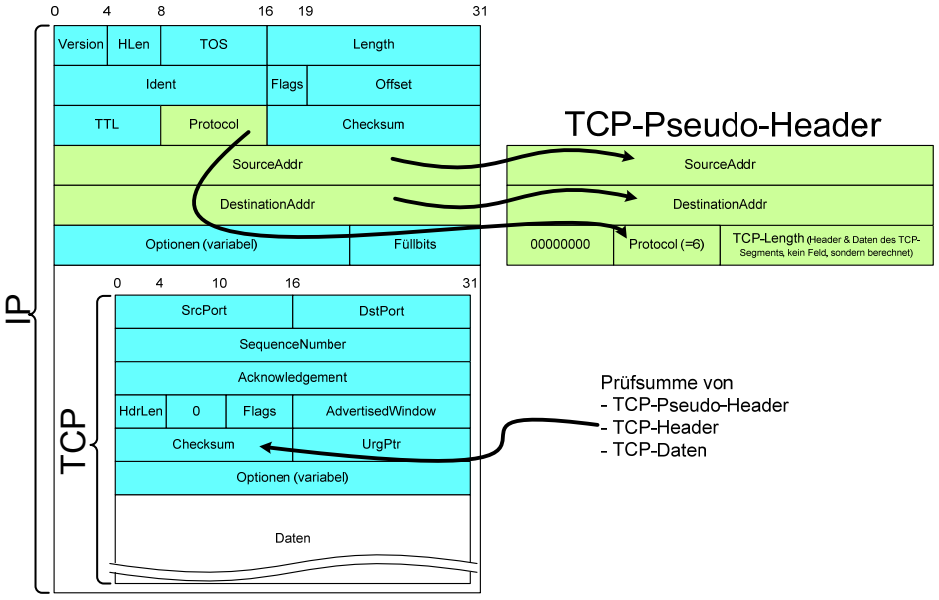
Prüfsumme der TCP-Nachricht:

- Aus Sicht der Anwendung bestehen Endpunkte neben den Ports auch aus den IP-Teilnehmern, repräsentiert durch deren IP-Adressen
  - IP-Adressen sind Bestandteil der IP-Header, also "außerhalb" der TCP-Nachricht
  - Änderungen der IP-Adressen beim Transport wären ein Problem für die TCP-Verbindung

Lösung: die TCP-Checksumme berücksichtigt auch einige Felder des IP-Pakets

- Eine Verletzung der Ebenen-Trennung
- Die Felder aus dem IP-Paket werden TCP-Pseudo-Header genannt

TCP-Pseudo-Header



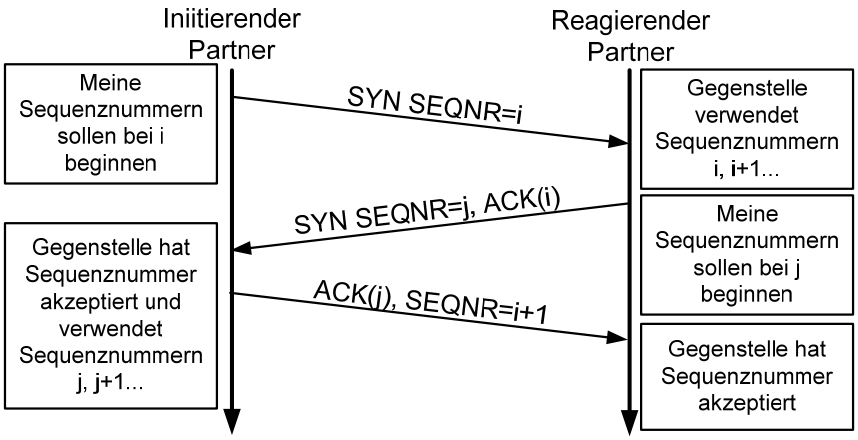
TCP-Verbindungsaufbau

Ziel des Verbindungsaufbaus:

- Beide Seiten sollen sich einige darüber sein, dass eine Verbindung besteht.
- Vereinbaren der Sequenznummern der Nachrichten:
  - Das Sliding-Window-Verfahren verlangt, dass die Reihenfolge der Nachrichten und die Vollständigkeit über Sequenznummern geregelt wird.
  - Diese Sequenznummern sind fortlaufend, der Beginn ist aber beliebig.
  - Während des Verbindungsaufbaus teilt jeder Partner die erste Sequenznummer mit.

TCP-Verbindungsaufbau

- Three-Way-Handshake:

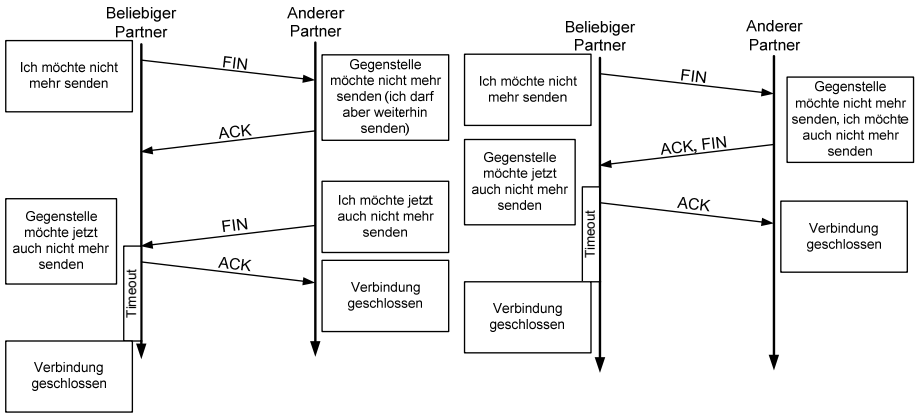


Ende der Verbindung

- Das Ende der Verbindung kann von jedem der beiden Partner separat angekündigt werden.
- Wenn ein Partner eine Verbindung beendet kann der andere weiter senden. Man spricht dann von einer *halboffenen* Verbindung.
- Möglichkeiten der Beendigung:
  - Four-Way-Handshake: jeder beendet die Verbindung separat (dazwischen: halboffene Verbindung)
  - Three-Way-Hanshake: der zweite Partner stimmt einer Beendingung zu, FIN und ACK werden in einer Nachricht versendet.

Ende der Verbindung

- Four- und Three-Way-Handshake:



Aufgabe 1 – Allgemeines über Netzwerke

a) Nennen Sie möglichst viele physikalische Medien, um Rechner miteinander zu verbinden.

b) Nennen Sie möglichst viele Computer-Anwendungen aus ihrem täglichen Leben, die eine Netzwerkverbindung erfordern.

c) Stellen Sie möglichst viele Netzwerk-Technologien auf unterschiedlichen Schichten zusammen. Versuchen Sie, diese zu Gruppen zusammenzufassen.

d) Warum kann man nicht einfach alle Rechner weltweit über ein einzelnes Medium miteinander verbinden?

e) Welche Geräte kennen Sie, die im Rahmen von Computer-Netzwerken eingesetzt werden?

Aufgabe 2 – Netzwerkfunktionen

a) Eine Netzwerkverbindung zwischen Rechnern muss viele verschiedene Funktionen ausführen. Versuchen Sie, möglichst viele zusammenzustellen.

b) Stellen Sie dar, wie diese Funktionen auf "Datenübertragungen" bei folgenden Alltagsbeispielen ausgeführt werden. In diesen Beispielen werden nicht immer alle oben genannte Funktionen benötigt.

- Versenden von Briefen
- Versenden von Rauchzeichen
- Gleichzeitige Unterhaltungen auf einer Party

Aufgabe 3 – Der optische Telegraf

Einführende Erklärungen

Eine der ältesten größeren Datenverbindungen in Deutschland war die optische Telegrafenlinie Berlin-Koblenz (1823-1849). Obwohl uns diese Art der Datenübertragung mittlerweile sehr antiquiert vorkommen dürfte, gibt es dennoch einige Parallelen zu heutigen Netzwerken. Die Eigenschaften der Telegrafenlinie:

- Es gab 61 Stationen im Abstand von ca. 10 km, die eine Kette bildeten.
- Jede Station war im Sichtkontakt zu den beiden Nachbarstationen.
- Pro Station standen 6 Flügel mit je 4 Positionen zur Kodierung zur Verfügung.

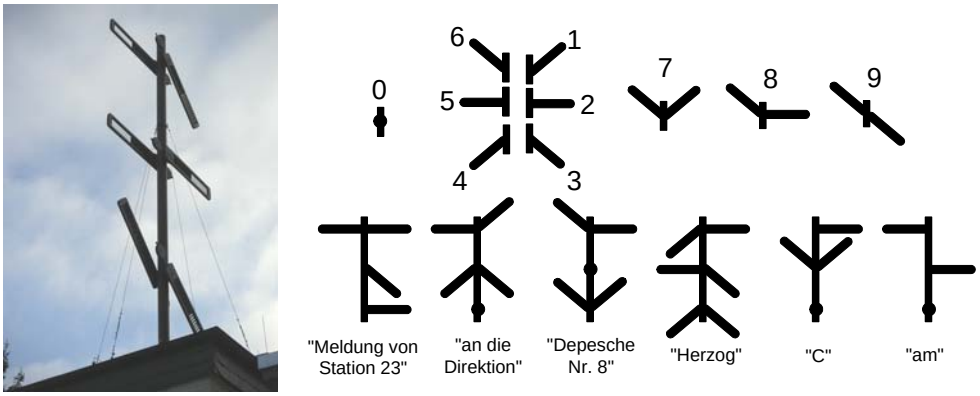


Abbildung 1: Historischer optischer Telegraf

Die dargestellten Kombinationen erlauben z.B. die Darstellung der Ziffern 0-9 mit einem Flügelpaar. Für alle mögliche Kombinationen (auch solche, die keine Einzelziffer beschreiben), gab es eine bestimmte Bedeutung, die aber nur den Telegrafen-Beamten der Endpunkte bekannt war.

Aufgabe

a) Angenommen, man kann alle 10 s eine neue Flügeleinstellung vornehmen. Wie viele Bits pro Sekunde können übertragen werden?

b) Angenommen, jede Station benötigt 1 min für die Weiterleitung. Wie groß ist die Ende-zu-Ende-Verzögerung?

c) Wie wurden die in Aufgabe 2 (Seite 1) gefundenen Funktionen realisiert?

Aufgabe 4 – Broadcast-Systeme

Die Datenübertragung kann auch in einer Richtung erfolgen. Broadcast-Systeme versenden unidirektional Daten an eine große Menge von Empfängern, ohne dass diese auf Transmissionen antworten können.

a) Zählen Sie möglichst viele Systeme auf, die nach diesem Prinzip arbeiten.

b) Welche Funktion ist bei Broadcast-Systemen, welche bei herkömmlichen Netzwerken einfacher.

Aufgabe 5 – Drahtlose Netzwerke

a) Tragen Sie möglichst viele Unterschiede zwischen kabelbasierten und drahtlosen Netzwerken zusammen.

b) Tragen Sie möglichst viele Unterschiede zwischen der Infrarot- und Funk-Übertragung zusammen.

Aufgabe 6 – Referenzmodelle

Beantworten Sie die folgenden Fragen stichwortartig:

a) Nennen Sie die sieben Schichten des OSI-Referenzmodells.

b) Nennen Sie Beispiele dafür, was durch die Bitübertragungsschicht festgelegt wird.

c) In der Sicherungsschicht des OSI-Referenzmodells: Auf welche *prinzipielle* Arten können Übertragungen gesichert werden?

d) Welche OSI-Schichten werden von IEEE 802 abgedeckt?

e) Wie werden die Schichten von IEEE 802 genannt?

f) Welche Schichten umfasst das TCP/IP-Referenzmodell?

g) Nennen Sie drei Internetprotokolle der Anwendungsschicht.

h) Nennen Sie Beispiele für die unterschiedlichen Adressen auf den vier Schichten des Internets.

Aufgabe 7 – Netzwerkschichten

Ordnen Sie die Begriffe in der ersten Spalte der folgenden Tabelle den richtigen Kommunikationsschichten (Spalten 2-6) zu. In einigen Fällen kann ein Begriff mehreren Schichten zugeordnet werden. Als Schichten stehen die OSI-Schichten Bitübertragungsschicht, Sicherungsschicht, Vermittlungsschicht, Transportschicht und die Anwendungsschicht des TCP/IP-Referenzmodells zur Verfügung.

Tabelle 1: Netzwerkfunktionen und Kommunikationsschichten

Begriff	Bitüber-tragungs-schicht	Siche-rungs-schicht	Vermitt-lungs-schicht	Trans-port-schicht	Anwen-dungs-schicht
TCP					
IP					
Modulation					
Infrarotlicht					
Routing					
Ping					
WLAN 802.11					
SMTP					
Network-Layer (TCP/ IP-Referenzmodell)					
Lichtwellenleiter					
Sentinel-Methode					
Sliding Window					
CSMA/CD					
Login/Logout					
Portnummern					

Aufgabe 8 – Bandbreiten

Einführende Erklärungen

Um die maximale Bandbreite für Übertragungskanäle zu berechnen gibt es zwei Formeln. Für einen *ungestörten Übertragungskanal* mit Bandbreite B (der *Nyquist-Frequenz*), der L diskrete Signalstufen darstellen kann, ergibt sich die maximale Datenrate D_{max} wie folgt:

$$D_{max} = 2 \cdot B \cdot \log_2(L)$$

Das *Shannon-Hartley-Gesetz* behandelt einen durch Rauschen *gestörten* Übertragungskanal. Hier gilt:

$$D_{max} = B \cdot \log_2(1 + \frac{S}{N})$$

S/N ist dabei der *Signal-Rauschabstand* (*Signal-Noise-Ratio*, SNR), der typischerweise in dB angegeben wird. Hierbei gilt:

$$SNR[dB] = 10 \cdot \log_{10}(\frac{S}{N})$$

Aufgabe

a) Wie viele Bit/s kann man maximal über einen ungestörten Übertragungskanal mit Bandbreite 1000 Hz und 6 Signalstufen übertragen?

b) Die 6 Signalstufen lassen nicht auf einfache Weise auf eine ganze Zahl von Bits abbilden (für 2 Bits benötigt man 4, für 3 Bits 8 Signalstufen). Wie kommt man dennoch möglichst nahe an die theoretische Obergrenze der maximalen Datenrate laut der Formel heran? Beschreiben Sie die Idee für eine konkrete Kodierung.

c) Wie viele Signalstufen benötigt man bei ungestörtem Kanal und 1000 Hz für eine Datenrate von 10000 Bit/s?

d) Ein Übertragungskanal mit Bandbreite 1000 Hz habe jetzt einen Signal-Rausch-Abstand von 25 dB. Wie hoch ist die maximale Datenrate?

e) Welches Signal-Rausch-Verhältnis muss ein gestörter Übertragungskanal einhalten, damit bei einer Bandbreite von 1000 Hz eine Datenrate von 5000 Bit/s erreicht werden kann?

Aufgabe 9 – Begriffe

Beantworten Sie die folgenden Fragen stichwortartig:

a) Nennen Sie die Hauptkategorien von Modulationsverfahren.

b) Erklären Sie die Betriebsweisen *synchron* und *asynchron*.

c) Erklären Sie die Betriebsweisen *simplex*, *halbduplex* und *vollduplex*.

d) Welche fünf Hauptprobleme müssen zur Sicherung von Daten auf der Sicherungsschicht gelöst werden?

Aufgabe 10 – Kodierung von Bitfolgen (1)

Einführende Erklärungen

Zur Kodierung von Bits über Signalpegel gibt es verschiedene Verfahren, z.B.

- *Non-Return to Zero* (*NRZ*): 0-Bit: low, 1-Bit: high;
- *Non-Return to Zero Inverted* (*NRZI*): 0-Bit: Halten des letzten Signalpegels, 1-Bit: Wechsel des Signals (high → low oder umgekehrt, in der Taktmitte);
- *Manchester*: 0-Bit: Wechsel von low → high, 1-Bit: Wechsel von high → low (jeweils in der Takt-Mitte).

Dieser Kodierungen unterscheiden sich im Umgang mit langen 0- oder 1-Ketten. Ohne zusätzliches Taktsignal kann sich ein Empfänger nur durch einen Signalwechsel mit dem Sendertakt synchronisieren. Während bei NRZ sowohl lange 0- als auch 1-Ketten problematisch sind, erzwingt NRZI zumindest bei 1-Ketten einen

ständigen Wechsel. Bei Manchester wird schließlich in jedem Takt mindestens einmal gewechselt, so dass sich der Empfänger ständig synchronisieren kann.

Aufgabe

Kodieren Sie die Bitfolge `0010 1001 0100 1110`

a) mit NRZ, b) mit NRZI, c) mit Manchester.

Stellen Sie jeweils den Signalverlauf grafisch dar.

Aufgabe 11 – Kodierung von Bitfolgen (2)

Einführende Erklärungen

Die *4B/5B-Kodierung* bildet 4 Datenbits auf 5 Code-Bits ab, die mit NRZI übertragen werden. Die Abbildung der Daten auf Codes wird folgendermaßen durchgeführt. Durch diese Kombination werden sowohl lange 0- als auch 1-Ketten vermieden.

Tabelle 2: 4b/5B-Kodierung

Daten	Code	Daten	Code	Daten	Code	Daten	Code
0000	11110	0100	01010	1000	10010	1100	11010
0001	01001	0101	01011	1001	10011	1101	11011
0010	10100	0110	01110	1010	10110	1110	11100
0011	10101	0111	01111	1011	10111	1111	11101

Aufgabe

a) Kodieren Sie die Bitfolgen mit 4B/5B und NRZI und stellen Sie den Signalverlauf dar.

- `0010 1111 0001`
- `1101 0000 1001`

b) Sie erhalten folgende Signalverläufe, die mit NRZI/4B5B kodiert wurden. Wie lauten die Nutzdaten?

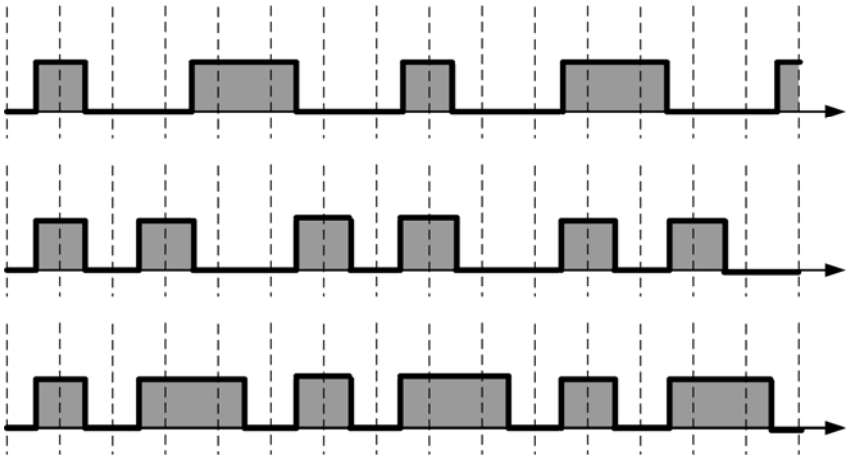


Abbildung 2: Signalverläufe vor der 4B/5B-Kodierung

Aufgabe 12 – Sentinel-Zeichen

Einführende Erklärungen

Ein älteres zeichenorientiertes Rahmen-Protokoll ist *BISYNC*. BISYNC-Rahmen sind wie folgt aufgebaut:

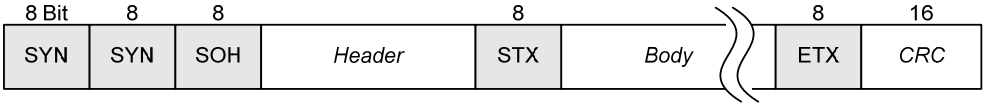


Abbildung 3: Aufbau eines BISYNC-Rahmens

Die Struktur eines Rahmens wird durch so genannte *Sentinel-Zeichen* vorgegeben. Für die BISYNC-Sentinel-Zeichen gilt folgende Festlegung:

Tabelle 3: BISYNC-Sentinel-Zeichen

Sentinel-Zeichen	Hex-Wert
SOH	01
STX	02
ETX	03
DLE	10
SYN	16

Sentinel-Zeichen im Nutzdaten-Teil werden durch das DLE–Zeichen vor der Interpretation geschützt (*Zeichenstopfen*).

Aufgabe

Sie erhalten folgende BISYNC-Rahmen (hexadezimal dargestellt). Wie lauten die Nutzdaten im Body-Teil?

a) `16 16 01 99 98 97 96 95 02 A1 A2 A3 A4 A5 03 A0 B7`

b) `16 16 01 99 98 97 96 95 02 01 02 10 03 04 05 03 76 35`

c) `16 16 01 99 98 97 96 95 02 10 03 10 10 10 03 03 92 55`

Aufgabe 13 – Die Zyklische Blocksicherung (1)

Einführende Erklärungen

Die Prüfsummenberechnung mittels *zyklischer Blocksicherung* (*Cyclic Redundancy Check*, CRC) funktioniert wie folgt:

- Man interpretiert die Nutzdatenbits als Koeffizienten eines binären Polynoms I(x), genannt das *Informationspolynom*.
- Man definiert ein *Generatorpolynom* G(x) vom Grad k. Dieses ist Sender und Empfänger bekannt. Es haben sich bestimmte Generatorpolynome etabliert:

Tabelle 4: Übersicht über etablierte CRC-Generatorpolynome

Name	Generatorpolynom
CRC-8 (ATM HEC)	$x^8+x^2+x^1+1$
CRC-10	$x^{10}+x^9+x^5+x^4+x+1$
CRC-12	$x^{12}+x^{11}+x^3+x^2+x+1$
CRC-16	$x^{16}+x^{15}+x^2+1$
CRC-CCITT	$x^{16}+x^{12}+x^5+1$
CRC-32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

- Man teilt $I(x)\cdot x^k$ durch das Generatorpolynom (mod 2) und erhält ein Restpolynom R(x) gemäß dem Zusammenhang

$$I(x)\cdot x^k = P(x)\cdot G(x)+R(x)$$

- Das Restpolynom R(x) ist die zu übertragende Prüfsumme. Darüber hinaus ist das *Codepolynom* $C(x) = I(x)\cdot x^k+R(x)$ ohne Rest durch das Generatorpolynom teilbar. Hierdurch erhält man eine Testmöglichkeit, ob eine Übertragung fehlerfrei war.

Aufgabe

a) Berechnen Sie zur Bitfolge `111` die CRC-Prüfsumme mit dem Generatorpolynom CRC-8.

b) Berechnen Sie zur Bitfolge `11011` die CRC-Prüfsumme mit dem Generatorpolynom CRC-16.

c) Sie erhalten eine Nachricht mit Prüfsumme durch die Bitfolge `1110111110` (das entspricht dem Codepolynom). Ist sie korrekt übertragen worden, wenn das Generatorpolynom x^3+x^2+1 lautete?

Aufgabe 14 – Die Zyklische Blocksicherung (2)

a) Konstruieren Sie eine Schaltung bestehend aus D-Flipflops und XOR-Gattern, die die CRC-Prüfsumme mit dem Generatorpolynom x^3+x^2+1 berechnet. Zeigen Sie schrittweise die Überprüfung des Codepolynoms `1110111110`. Vergleichen Sie das Ergebnis mit dem von Aufgabe 13 Teil c.

b) Verwenden Sie die Schaltung, um die CRC-Prüfsumme zu `110101` zu berechnen.

Aufgabe 15 – Die Parität (1)

Ein Paritätsbit wird so gesetzt, dass die Anzahl der Bits in einem Block *gerade* (*even parity*) oder *ungerade* (*odd parity*) ist. Sichert man eine Matrix von Bits sowohl reihen- als auch spaltenweise, spricht man von der *zweidimensionalen Parität*.

Sichern Sie die folgenden Reihen von 7-Bit-Blöcken durch eine zweidimensionale Parität. Die Reihen sind hexadezimal dargestellt, wobei das höchstwertige Bit jeweils 0 lautet. Geben Sie die abgesicherte Reihe von Hex-Zahlen an – setzen Sie dazu das höchstwertige Bit gemäß der Parität und berechnen Sie zusätzlich die Gesamt-Parität. Es soll die *gerade* Parität verwendet werden.

a) `2C 0E 5A 1A` b) `00 1C 76 38`

Aufgabe 16 – Die Parität (2)

a) Zeigen Sie, dass man mit der zweidimensionalen Parität 1-, 2- und 3-Bit-Fehler sicher erkennen kann.

b) Unter welchen Umständen werden 4-Bit-Fehler nicht erkannt?

Aufgabe 17 – Sliding Window (1)

Einführende Erklärungen

Beim *Sliding-Window-Verfahren* werden mehrere Rahmen in einem Sendefenster ausgesendet, bevor die erste Bestätigung erwartet wird. Bei Eintreffen einer Bestätigung wird das Sendefenster verschoben und der Sender kann weitere Rahmen aussenden. Die Größe des Sendefensters wird *Send Window Size* (SWS) genannt.

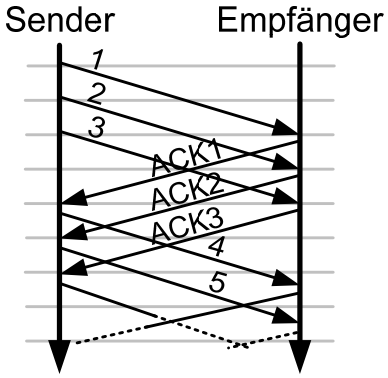


Abbildung 4: Raumzeit-Diagramm für ein Sliding-Window-Verfahren (SWS = 3)

Auf Empfängerseite wird ein Empfangspuffer der Größe *Receive Window Size* (RWS) verwaltet. Treten in der Empfangsfolge Lücken auf, kann der Empfänger einige Rahmen in diesem Puffer speichern, bis die fehlenden Rahmen nachgeliefert wurden.

Ein Sender kann einen fehlerhaften Rahmentransport in der Regel nur durch das Ausbleiben einer positiven Bestätigung (*ACK*) erkennen. Hierzu wird ein Timeout verwendet, der für jede Rahmenaussendung geprüft wird.

Aufgabe

In einem speziellen Sliding-Window-Protokoll gelten folgende Festlegungen:

- Der Sender sendet 1 Rahmen/Zeiteinheit aus, wenn dies durch das Sendefenster möglich ist.
- Die Übertragungszeit für Rahmen und ACKs beträgt jeweils 2 Zeiteinheiten.
- Die Verarbeitungszeit beim Empfänger beträgt 1 Zeiteinheit (Rahmen-Empfang bis ACK-Aussendung). Daraus ergibt sich eine Round-Trip-Zeit von 5 Zeiteinheiten.
- Die Verarbeitungszeit beim Sender beträgt 1 Zeiteinheit (ACK-Empfang bis zur potentiellen neuen Rahmen-Aussendung).

- Es sollen *kumulative ACKs* verwendet werden, d.h. eine Bestätigung erfolgt nur über eine lückenlos empfangene Folge von Rahmen.
- Der Timeout sei 7 Zeiteinheiten.
- Die Fenstergrößen sind SWS = RWS = 4.

a) Wie sieht das Raumzeit-Diagramm der Rahmen 1...8 aus, wenn jede Übertragung erfolgreich ist?

b) Wie sieht das Raumzeit-Diagramm der Rahmen 1...8 aus, wenn jede Übertragung außer der ersten Übertragung von Rahmen 2 erfolgreich ist?

Aufgabe 18 – Sliding Window (2)

Bei einem weiteren Sliding-Window-System sendet der Sender Rahmen mit den Nummern 0 bis 8 an einen Empfänger. Die Zustellung der Rahmen ist bis auf die Rahmen 1 und 4 sofort erfolgreich. Bei den Rahmen 1 und 4 geht die jeweils erste Kopie beim Sendevorgang verloren, die zweite Zustellung ist jeweils erfolgreich. Die ACKs des Empfängers werden in jedem Fall erfolgreich zugestellt.

Der Empfänger verwendet kumulative ACKs. Nehmen Sie folgende Zeiten an:

- Die Transportzeit für Rahmen und ACKs beträgt 10 ms.
- Der Timeout beträgt 30 ms.
- Eine Station benötigt 1 ms um eine Reaktion auf einen Rahmen oder ACK durchzuführen (die Round-Trip-Zeit ist also insgesamt 21 ms).
- Darf der Sender gemäß dem Sendefenster weitere Rahmen aussenden, so geschieht dies mit der Rate von 1 Rahmen/2 ms.

a) Zeichnen Sie Raumzeit-Diagramme, die den Rahmen- und ACK-Transport darstellen und zwar für die Konfigurationen SWS = RWS = 3 sowie SWS = RWS = 6. Stellen Sie auch jeweils die Sende- und Empfangsfenster dar.

b) Bisher wird der Rahmen-Verlust nur durch einen Timeout festgestellt. Der Sender könnte aber auch den wiederholten Empfang *desselben* ACKs auswerten (so genannte *duplicate ACKs*). Diskutieren Sie Vor- und Nachteile.

Aufgabe 19 – Sliding Window (3)

Führen Sie analog zu Aufgabe 18 (Seite 11) das Sliding-Window-Protokoll für SWS = RWS = 3 mit *selektiven ACKs* durch. Selektive ACKs bestätigen separat *alle* erfolgreich empfangenen Rahmen, also nicht nur den letzten in einer ununterbrochenen Folge erfolgreich empfangener Rahmen.

Diskutieren Sie Vor- und Nachteile von selektiven ACKs.

Aufgabe 20 – Die Bridge (1)

Einführende Erklärungen

Eine Bridge verbindet zwei LAN-Segmente:

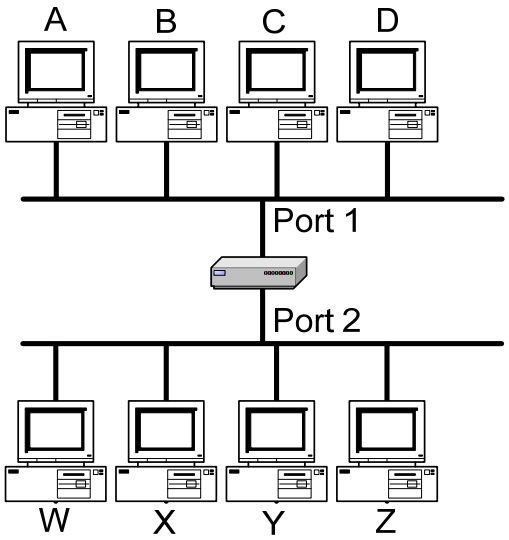


Abbildung 5: Durch eine Bridge verbundene LAN-Segmente

Die Bridge soll *selbstlernend* sein, d.h. immer wenn sie einen Rahmen empfängt, merkt sie sich den Sender und die Portnummer des Empfangs. Hierzu verwaltet sie eine Tabelle, die die Zuordnung

Host-Adresse → Port, um den Host zu erreichen

ermöglicht.

Aufgabe

Die Bridge kennt zunächst keine Zuordnung. Es findet jetzt die Kommunikation statt, wie in der nächsten Tabelle dargestellt. Angegeben sind jeweils der Sender und der Empfänger eines Rahmens. Die Bridge sendet nun je nach Lernstatus den Rahmen an einen oder keinen der Ausgabeports. Geben Sie bei jeder Sendung an, zu welchem Port die Übertragung kopiert wird (Beispiel in der ersten Zeile).

Tabelle 5: Zuordnung der Transmissionen zu Ports

Sender	Empfänger	Port 1	Port 2
W	Y	✓	
A	Y		
X	W		
A	C		
D	B		
B	D		
X	D		
Z	W		
C	B		

Aufgabe 21 – Die Bridge (2)

Gegeben sei folgendes Netzwerk mit zwei selbstlernenden Bridges.

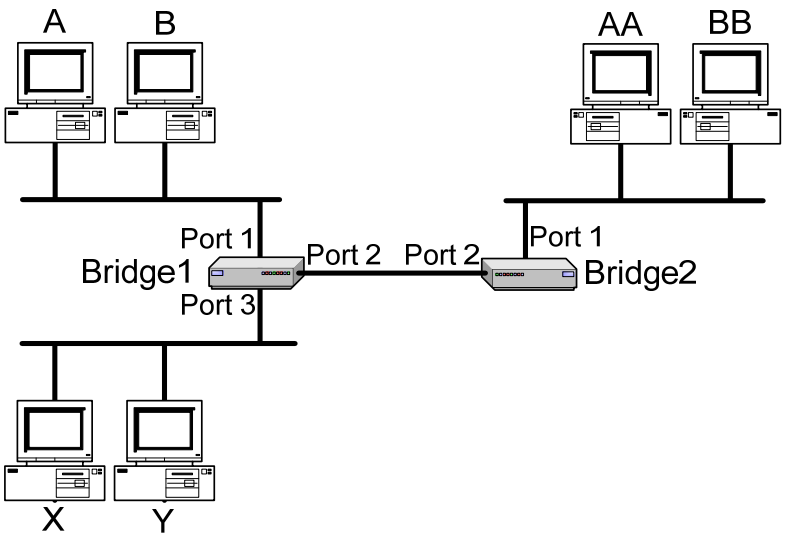


Abbildung 6: Durch zwei Bridges verbundene LAN-Segmente

Die folgende Tabelle gibt an, in welcher Reihenfolge die Rahmen im Netzwerk versendet werden. Markieren Sie in den Spalten "Bridge$_i$ Port$_j$", ob der entsprechende Rahmen über den Port ausgegeben wird.

Tabelle 6: Zuordnung der Transmissionen zu Ports von zwei Bridges

Sender	Empfänger	Bridge 1 Port 1	Bridge 1 Port 2	Bridge 1 Port 3	Bridge 2 Port 1	Bridge 2 Port 2
A	B					
AA	B					
X	A					
Y	BB					
BB	AA					
B	X					
AA	B					

Aufgabe 22 – Der Ethernet-Medienzugriff

In einem Ethernet-Netzwerk nach dem CSMA/CD-Verfahren müssen die *Sender* einer Übertragung eine Kollision sicher erkennen können. Hierzu vergleichen sie ihre eigene Übertragung mit dem Signal auf dem Netzwerk. Ist der Vergleich negativ, so wurde das Signal durch eine andere Übertragung gestört.

Sind zwei Sender zu weit voneinander entfernt, kann es sein, dass diese die Kollision nicht mehr erkennen können, da das kollidierende Signal erst *nach* der kompletten eigenen Aussendung eintrifft. In dieser Aufgabe berechnen wir, wie weit zwei Stationen in einem 10-MBit-Ethernet maximal voneinander entfernt sein dürfen, um Kollisionen sicher erkennen zu können. Wir machen folgende Annahmen:

- Es wird mit 10 MBit/s (= $10 \cdot 10^6$ Bit/s) gesendet.
- Ein Rahmen hat eine Mindestlänge von 512 Bits.
- Erkennt eine Station eine Kollision, so sendet sie noch ein so genanntes *Stausignal*, bevor sie abbricht. Die Länge des Signals ist hierbei unwichtig – relevant ist nur, dass über das Stausignal andere Sender eine Kollision erkennen können.
- Die Signale breiten sich mit Lichtgeschwindigkeit aus – hier vereinfacht mit 300 000 km/s angenähert.

a) Berechnen Sie den maximalen Abstand zweier Stationen unter der Annahme, dass keine Verzögerung außer der Signalausbreitung auf dem Verbindungsweg vorliegt.

b) Berechnen Sie den maximalen Abstand, wenn zwischen zwei Stationen maximal vier Repeater mit je einer Verzögerung von 4 μs geschaltet werden können.

Aufgabe 23 – Der WLAN-Medienzugriff (1)

Einführende Erklärungen

Wireless LAN nach IEEE 802.11 verwendet das CSMA/CA-Verfahren zum Medienzugriff:

- Ist das Medium frei, sendet ein Sender sofort.
- Ist es belegt, wartet er bis zum Ende der Übertragung. Danach wartet er zusätzlich eine Wartezeit DIFS (*DCF Interframe Space*) sowie eine zufallszahlabhängige Wartezeit ab.
- Tritt während der Wartezeit eine Medienbelegung durch eine andere Station auf, wird der Warte-Vorgang wiederholt. Die zufallszahlabhängige Wartezeit wird beim letzten Stand eingefroren.

Die maximale Wartezeit nach dem DIFS wird *Contention Window* genannt.

Aufgabe

Gegeben sei ein Wireless LAN mit vier Stationen S1 bis S4. Während Station S1 einen Rahmen1 sendet, bewerben sich die Stationen S2, S3 und S4 um den Zugriff auf das Funkmedium.

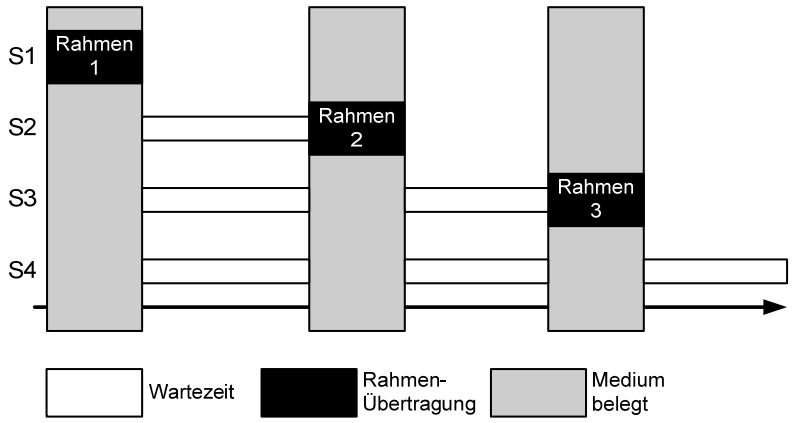


Abbildung 7: Medienbelegung im WLAN

Es wird das einfache CSMA/CA-Verfahren ohne Bestätigung eingesetzt, um den gemeinsamen Zugriff zu regeln. Dabei wählen die Stationen die folgenden Zufallszahlen für die Berechnung der Wartezeiten:

S2: 2, S3: 5, S4: 6.

Berechnen Sie für die Stationen S2, S3 und S4 jeweils die Zeiten, die zwischen dem Ende von Rahmen1 und dem Start des eigenen Rahmens vergehen. Nehmen Sie dazu folgende Vereinfachung an:

- die zufallsabhängigen Wartezeiten bilden sich aus Einheiten der Zeit T
- ein DCF Interframe Space hat eine Länge von 2,5T
- jeder Rahmen hat eine Länge von 10T

Aufgabe 24 – Der WLAN-Medienzugriff (2)

In Aufgabe 23 (Seite 15) sind wir im Wireless LAN davon ausgegangen, dass jede Station beim Medienzugriff jede andere empfangen kann.

Sei durch (X, Y) beschrieben, dass sich die Stationen X und Y gegenseitig empfangen können, -(X, Y) beschreibt, dass kein direkter Empfang möglich ist.

a) Bei drei Stationen S1, S2, S3: was für ein Problem könnte durch die Kombination (S1, S2), (S2, S3) aber -(S1, S3) entstehen? Wie könnte man es lösen?

b) Bei vier Stationen S1, S2, S3, S4: was für ein Problem könnte durch die Kombination (S1, S2), (S2, S3), (S3, S4) aber -(S1, S4), -(S1, S3), -(S2, S4) entstehen? Wie könnte man es lösen?

Untersuchen Sie diese Konfigurationen nur auf eventuelle Probleme beim Medienzugriff. Lassen Sie Probleme der prinzipiellen Erreichbarkeit einiger Knoten außer Acht (diese werden ggf. von der OSI-Schicht 3 gelöst).

Aufgabe 25 – Der WLAN-Medienzugriff (3)

a) In einem WLAN bewerben sich zwei Stationen um das Medium. Das Contention Window sei 7. Wie groß ist die Wahrscheinlichkeit für eine Kollision?

b) Wie groß ist die Kollisionswahrscheinlichkeit bei drei Stationen und einem Contention Window von 7?

c) Leiten Sie eine allgemeine Formel für die Kollisionswahrscheinlichkeit p_i für i Stationen in Abhängigkeit des Contention Windows cw her.

d) Bei welcher der WLAN-Stufen für Contention Windows ergibt sich eine Kollisionswahrscheinlichkeit für drei Stationen von weniger als 5%?

Aufgabe 26 – Begriffe

Grenzen Sie den Begriff *Circuit Switching* gegenüber *Packet Switching* ab.

Aufgabe 27 – ATM

Beantworten Sie die folgenden Fragen stichwortartig:

a) Warum benutzt ATM *kleine* Zellen mit *fester* Länge?

b) ATM sichert nur den Header, nicht aber die Nutzdaten mit der Prüfsumme ab. Nennen Sie Gründe dafür?

c) ATM unterstützt Dienstgüte. Was für Nachteile könnten entstehen, wenn höhere Protokolle *ohne* Dienstgüteunterstützung ATM benutzen?

Aufgabe 28 – ATM-Dienstgüte

Erläutern Sie die vier ATM-Dienstgüte-Klassen an selbstgewählten Szenarien.

Aufgabe 29 – Routing (1)

Tragen Sie möglichst viele Unterschiede von *Distance-Vector-* und *Link-State-Routing* zusammen.

Aufgabe 30 – Dijkstra

Einführende Erklärungen

Der Algorithmus von *Dijkstra* zur Bestimmung von Laufweg-Quellbäumen arbeitet wie folgt. Sei

- N_q der Quellknoten,
- M die Menge der schon betrachteten Knoten,
- T der sukzessive aufgebaute Quellbaum und
- d_i der bisher berechnete Abstand von N_q zu N_i.

Initialisierung:

- $M \leftarrow \{N_q\}$
- $T \leftarrow \{N_q\}$
- Für alle Nachbarn N_i von N_q: $d_i \leftarrow$ Abstand(N_q, N_i)
- Für alle anderen Knoten: $d_i \leftarrow \infty$

Solange noch nicht alle Knoten in M erfasst sind:

- Bestimme den Knoten N_w mit dem geringsten Abstand zu N_q, genauer: wähle N_w mit $d_w = \min\{d_i \mid N_i \notin M\}$.
- Bestimme die Kante k mit dem kleinsten Gewicht, die N_w und ein Knoten aus T verbindet.
- $M \leftarrow M \cup \{N_w\}$, $T \leftarrow T \cup \{k\}$
- $d_i \leftarrow \min \{d_i, d_w + \text{Abstand}(N_w, N_i)\}$ für alle $N_i \notin M$

Danach gilt:

- In T steht der gesuchte Quellbaum.
- In d_i steht der kürzeste Abstand vom Quellknoten zu N_i.

Aufgabe

Gegeben sei folgendes Netzwerk:

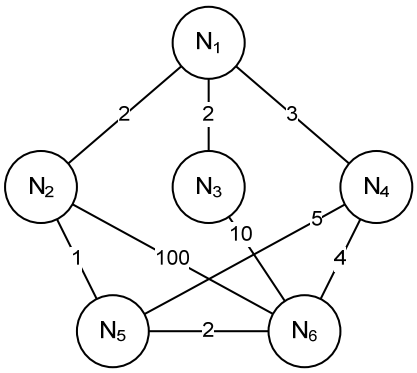


Abbildung 8: Netzwerk zur Berechnung des Quellbaums

Wenden Sie Dijkstras Algorithmus zur Bestimmung von Laufweg-Quellbäumen für Knoten N_5 an.

Aufgabe 31 – Distributed Bellman-Ford (1)

Einführende Erklärungen

Das Routing-Verfahren *Distributed Bellman-Ford* (DBF) arbeitet wie folgt:

- Jeder Knoten verwaltet eine Routing-Tabelle mit den Spalten *Ziel* (an welchen Zielknoten soll ein Paket gesendet werden), *Hop* (über welchen direkten Nachbarn wird gesendet), *Metrik* (welche Distanz hat das Ziel). Ein Knoten hat für jeden Knoten im Netzwerk eine separate Zeile in dieser Tabelle.
- Für eine Routing-Tabelle eines Knotens N_a bezeichne H_{ab} den Eintrag der Spalte Hop für einen Zielknoten N_b und M_{ab} den Eintrag der Spalte Metrik für einen Zielknoten N_b.
- Eine Tabelle wird mit $H_{ij} \leftarrow ?$, $M_{ij} \leftarrow \infty$ für $i \neq j$ sowie mit $H_{ij} \leftarrow N_i$, $M_{ij} \leftarrow 0$ für $i = j$ initialisiert.
- Für jeden direkten Nachbarn N_j von N_i wird eingetragen: $H_{ij} \leftarrow N_j$, $M_{ij} \leftarrow$ Abstand(N_i, N_j). Der Abstand wird für die so genannte *Hop-Metrik* auf 1 gesetzt.
- Jeder direkte Nachbar N_j von N_i sendet seine Routing-Tabelle an N_i. Für einen Tabelleneintrag zu N_k wird überprüft, ob $M_{ij}+M_{jk}<M_{ik}$. Wenn dies gilt, erfolgen die Zuweisungen:

- $H_{ik} \leftarrow N_j$,
- $M_{ik} \leftarrow M_{ij}+M_{jk}$.

Für die Hop-Metrik vereinfacht sich die letzte Zuweisung zu $M_{ik} \leftarrow 1+M_{jk}$.

Aufgabe

Gegeben sei folgendes Netzwerk:

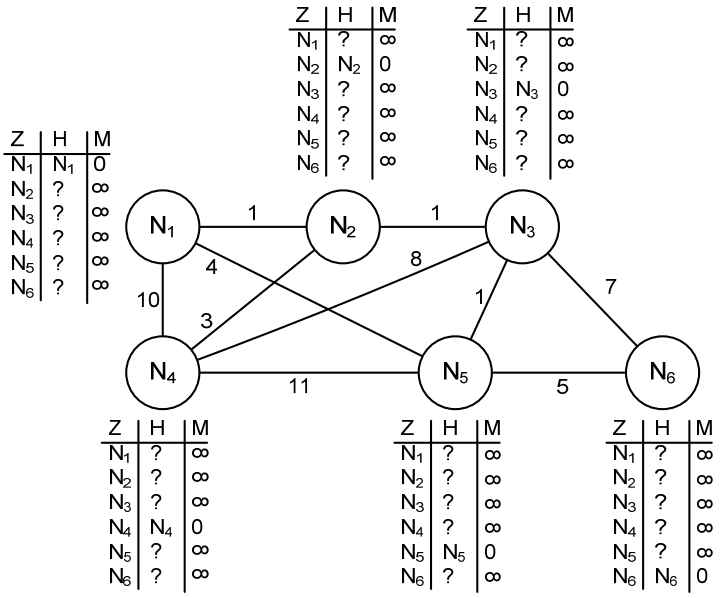


Abbildung 9: Netzwerk mit DBF-Tabellen

Führen Sie das DBF-Verfahren aus und geben Sie nach jedem Schritt die Routing-Tabellen an. Ein Schritt gilt dabei als durchgeführt, wenn jeder Knoten die Routing-Tabellen mit all seinen Nachbarn einmalig abgeglichen hat.

Aufgabe 32 – Distributed Bellman-Ford (2)

In einem Netzwerk mit sechs Knoten wird das Routing nach Distributed Bellman-Ford durchgeführt. Für die Knoten N_1, N_2 und N_3 sind die aktuellen Routing-Tabellen dargestellt.

Hinweis: in diesem Netzwerk wird *nicht* die Hop-Metrik zugrunde gelegt.

Tabelle 7: Routing-Tabellen im DBF-Verfahren

N_1		
Ziel	Hop	Metrik
N_1	N_1	0
N_2	N_2	1
N_3	?	∞
N_4	N_4	6
N_5	N_4	8
N_6	N_4	7

N_2		
Ziel	Hop	Metrik
N_1	N_1	1
N_2	N_2	0
N_3	N_3	2
N_4	?	∞
N_5	N_5	6
N_6	?	∞

N_3		
Ziel	Hop	Metrik
N_1	?	∞
N_2	N_2	2
N_3	N_3	0
N_4	N_4	1
N_5	N_4	3
N_6	N_4	2

Die benachbarten Knoten gleichen nun ihre Routing-Tabellen ab. Bei einem Abgleich aktualisieren immer beide beteiligten Knoten ihre Routing-Tabellen. Die Reihenfolge der Abgleiche ist:

- N_1 mit N_2
- N_2 mit N_3
- N_1 noch einmal mit N_2

Stellen Sie die Routing-Tabellen der beteiligten zwei Knoten nach jedem Abgleich in folgenden Tabellen dar.

Tabelle 8: Routing-Tabellen nach Abgleich N_1 mit N_2

N_1		
Ziel	Hop	Metrik
N_1		
N_2		
N_3		
N_4		
N_5		
N_6		

N_2		
Ziel	Hop	Metrik
N_1		
N_2		
N_3		
N_4		
N_5		
N_6		

N_3
bleibt unverändert

Tabelle 9: Routing-Tabellen nach Abgleich N_2 mit N_3

N_1
bleibt unverändert

N_2		
Ziel	Hop	Metrik
N_1		
N_2		
N_3		
N_4		
N_5		
N_6		

N_3		
Ziel	Hop	Metrik
N_1		
N_2		
N_3		
N_4		
N_5		
N_6		

Tabelle 10: Routing-Tabellen nach zweitem Abgleich N_1 mit N_2

N_1		
Ziel	Hop	Metrik
N_1		
N_2		
N_3		
N_4		
N_5		
N_6		

N_2		
Ziel	Hop	Metrik
N_1		
N_2		
N_3		
N_4		
N_5		
N_6		

N_3
bleibt unverändert

Aufgabe 33 – DSDV

Einführende Erklärungen

Das Routing-Verfahren *Destination-Sequenced Distance Vector* (DSDV) geht auf das DBF-Verfahren zurück und verhindert das so genannte *Count-to-Infinity-Problem*. Dieses Problem tritt dann auf, wenn ein Netzwerk-Segment abgetrennt wird und im abgetrennten Bereich mindestens zwei Knoten verbleiben: obwohl ein Knoten explizit den Verbindungswegfall wahrnimmt, erhält er vom zweiten Knoten noch einen gültigen Eintrag über den nicht erreichbaren Bereich und verwendet diese Route. In der Folge wird der Metrik-Wert pro Abgleich zwar immer größer, der Wegfall spiegelt sich aber nicht explizit in den Tabellen wider.

DSDV führt zur Lösung des Problems noch eine Spalte *Sequenznummer* ein (im Folgenden mit S_{ij} gekennzeichnet). Das DBF-Verfahren wird wie folgt abgeändert:

- Eine Tabelle wird mit $H_{ij} \leftarrow ?$, $M_{ij} \leftarrow \infty$, $S_{ij} \leftarrow -1$ für $i \neq j$ sowie mit $H_{ij} \leftarrow N_i$, $M_{ij} \leftarrow 0$, $S_{ij} \leftarrow 0$ für $i = j$ initialisiert.
- Jeder direkte Nachbar N_j von N_i sendet seine Routing-Tabelle an N_i. Für einen Tabelleneintrag zu N_k wird überprüft, ob $S_{ik}<S_{jk}$ oder ($M_{ij}+M_{jk}<M_{ik}$ und $S_{ik} = S_{jk}$). Wenn dies gilt, erfolgen die Zuweisungen:

- $H_{ik} \leftarrow N_j$,
- $M_{ik} \leftarrow M_{ij}+M_{jk}$,
- $S_{ik} \leftarrow S_{jk}$

Für die Hop-Metrik vereinfacht sich die zweite Zuweisung zu $M_{ik} \leftarrow 1+M_{jk}$.

– Hat ein Knoten N_j den Eintrag an alle Nachbarn gesendet, erhöht er die Sequenznummer: $S_{jj} \leftarrow S_{jj}+2$.
– Entdeckt ein Knoten Ni eine Unterbrechung zu N_j, sucht N_i alle Zeilen für die gilt: $H_{ik} = N_j$: für diese wird ausgeführt: $H_{ik} \leftarrow ?$, $M_{ik} \leftarrow \infty$, $S_{ik} \leftarrow S_{ik}+1$.

Aufgabe

a) In folgendem Netzwerk mit drei Knoten wird Routing nach dem DSDV-Verfahren angewendet.

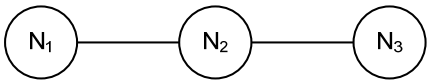


Abbildung 10: Netzwerk für das DSDV-Verfahren

Die Routing-Tabellen sehen wie folgt aus:

Tabelle 11: Routing-Tabellen im DSDV-Verfahren

N_1			
Ziel	Hop	Met	Seq
N_1	N_1	0	70
N_2	N_2	1	64
N_3	N_2	2	88

N_2			
Ziel	Hop	Met	Seq
N_1	N_1	1	70
N_2	N_2	0	64
N_3	N_3	1	88

N_3			
Ziel	Hop	Met	Seq
N_1	N_2	2	70
N_2	N_2	1	64
N_3	N_3	0	88

Die Verbindung zwischen N_2 und N_3 wird nun unterbrochen. Wie sehen die Routing-Tabellen aus, nachdem *alle* Knoten die Information über die Unterbrechung verarbeitet haben und wieder ein stabiler Zustand entstanden ist?

Tabelle 12: Routing-Tabellen nach der Unterbrechung zwischen N_2 und N_3

N_1			
Ziel	Hop	Met	Seq
N_1			
N_2			
N_3			

N_2			
Ziel	Hop	Met	Seq
N_1			
N_2			
N_3			

N_3			
Ziel	Hop	Met	Seq
N_1			
N_2			
N_3			

b) N_3 wird jetzt mit N_1 verbunden (die Unterbrechung N_2 und N_3 bleibt weiterhin bestehen). *Alle* Knoten senden eine Aktualisierung aus. Wie sehen die Routing-Tabellen aus, nachdem alle Knoten die Aktualisierungen verarbeitet haben und wieder ein stabiler Zustand entstanden ist?

Tabelle 13: Routing-Tabellen nach der Verbindung zwischen N_3 und N_1

N_1			
Ziel	Hop	Met	Seq
N_1			
N_2			
N_3			

N_2			
Ziel	Hop	Met	Seq
N_1			
N_2			
N_3			

N_3			
Ziel	Hop	Met	Seq
N_1			
N_2			
N_3			

Aufgabe 34 – OLSR

Einführende Erklärungen

Optimized Link State Routing (OLSR) erweitert den generellen Link-State-Ansatz um zwei Optimierungen: Kontrollpakete werden verkleinert und sie werden nicht mehr über alle Nachbarn geflutet. Erreicht werden die Optimierungen durch die so genannten *Multipoint Relays* – das sind diejenigen Nachbarn eines Knotens, über die man alle Nachbarn zweiter Stufe erreichen kann. Zur Beschreibung des Verfahrens führen wir folgende Bezeichnungen ein:

- $N(N_i)$ bezeichne die Menge der Nachbarn von N_i.
- $N2(N_i)$ bezeichne die Menge der Knoten, die exakt 2 Schritte von N_i entfernt sind.
- $MPR(N_i)$ bezeichne die Multipoint Relays von N_i.
- $MPRsel(N_i)$ bezeichne alle Knoten, die N_i als Multipoint Relay ausgewählt haben. Anders ausgedrückt: $N_j \in MPRsel(N_i) \Leftrightarrow N_i \in MPR(N_j)$.

Beim Fluten gibt ein Knoten nur ein Paket weiter, wenn er es von einem Knoten aus MPRsel erhalten hat. Die gefluteten Kontrollpakete enthalten darüber hinaus nicht alle Nachbarschaftsinformationen, sondern nur die Mengen MPRsel.

Es gibt verschiedene Möglichkeiten, MPR zu berechnen. OLSR funktioniert dabei auch mit suboptimalen MPR-Mengen. Ein Vorschlag für die Berechnung lautet:

- Beginne mit einer leeren Menge MPR.
- Füge alle Knoten aus N hinzu, die eine einzige Verbindung zu Knoten aus N2 darstellen.
- Solange es noch Knoten in N2 gibt, die nicht über Knoten aus MPR erreichbar sind:
    - Wähle den Knoten aus N, der die meisten noch nicht erreichbaren Knoten aus N2 erreicht.

- Gibt es mehrere Knoten mit der gleichen Zahl, wähle den Knoten mit den meisten Nachbarn.
- Füge diesen Knoten zu MPR hinzu.

– Wenn es Knoten N_j aus MPR gibt, so dass MPR\{N_j} immer noch alle Knoten aus N2 erreicht, dann lösche N_j.

Basierend auf den gefluteten Kontrollpaketen können nun kürzeste Wege berechnet werden. Der OLSR-Algorithmus basiert dabei auf dem Dijkstra-Verfahren. Es werden zwei Tabellen angelegt: eine Topologie-Tabelle sammelt alle gefluteten Kontrollpakete. Als Beispiel: wenn N_2 die Menge MPRsel = {N_1, N_3} flutet, macht jeder andere Knoten folgende Einträge in der Topologie-Tabelle:

Tabelle 14: OLSR-Topologie-Tabelle

Knoten (Sender des Kontrollpakets)	Selektor (Element aus MPRsel)
…	…
N_2	N_1
N_2	N_3
…	…

Mit einer zweiten Tabelle, der *Routing-Tabelle*, werden die kürzesten Wege berechnet. Einträge haben die Form (Ziel, Hop, Metrik). Die Berechnung geht wie folgt:

– Beginne mit einer leeren Routing-Tabelle.
  - Füge einmalig (N_i, N_i, 0) ein.
  - Für jedes $N_j \in N(N_i)$ füge (N_j, N_j, 1) ein.
– Zum Zählen der Hops wird eine Variable h benutzt, die mit h ← 1 initialisiert wird.
– *Schleifenanfang*: Durchlaufe die Routing-Tabelle zeilenweise.
  - Teste dabei für alle Zeilen (z_r, h_r, m_r) der Routing-Tabelle, ob es in der Topologie-Tabelle einen Eintrag (k_t, s_t) gibt mit $z_r = s_t$.
  - Wenn ja, lösche diese Zeile aus der Topologie-Tabelle.
– Ist die Topologie-Tabelle leer, ist die Berechnung beendet.
– Durchlaufe die Topologie-Tabelle zeilenweise.
  - Teste dabei für alle Zeilen (k_t, s_t), ob es einen Eintrag (z_r, h_r, m_r) der Routing-Tabelle gibt mit $z_r = k_t$ und $m_r = h$.
  - Wenn ja, füge die Zeile (s_t, h_r, h+1) in die Routing-Tabelle ein.
– Setze h ← h+1 und fahre mit dem Schleifenanfang fort.

Aufgabe

a) Gegeben ist folgendes Netzwerk:

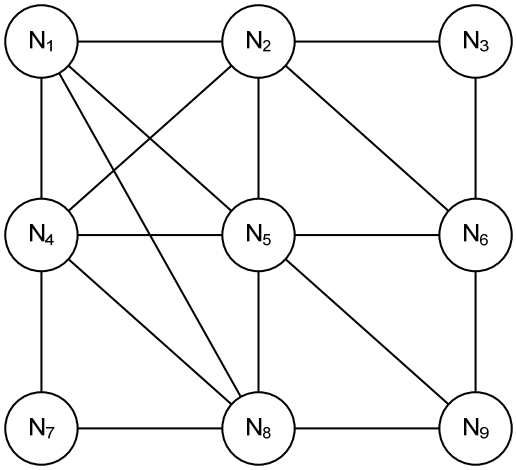


Abbildung 11: Netzwerk zur Berechnung von MPR(N_1)

Berechnen Sie die Menge MPR(N_1) nach dem vorgestellten Verfahren.

b) Gegeben sei nun folgendes Netzwerk:

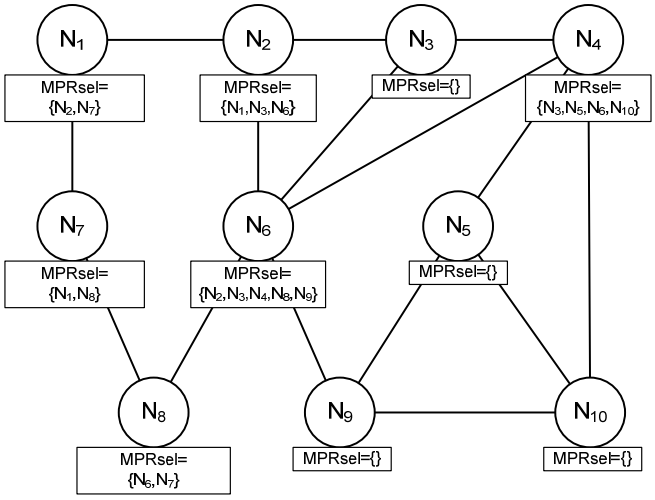


Abbildung 12: OLSR-Netzwerk

Bei den Knoten sind die Mengen MPRsel angegeben. Der Knoten N_1 soll nun anhand der zugesandten Kontrollpakete eine Routing-Tabelle berechnen.

Stellen Sie dazu zuerst die Topologietabelle für N_1 auf und führen Sie den OLSR-Algorithmus zur Berechnung der kürzesten Wege für N_1 durch.

Aufgabe 35 – Routing (2)

Beantworten Sie die folgenden Fragen stichwortartig:

a) Was versteht man unter dem Count-to-Infinity-Problem und wie kann es gelöst werden?

b) Was versteht man unter Source-Routing?

c) Was versteht man unter Fluten? Nennen Sie Details.

d) Der in Aufgabe 31 (Seite 18) dargestellte Algorithmus von DBF erlaubt nur, dass Distanzen kleiner werden. In welchen Fällen muss die Distanz jedoch vergrößert werden? Erweitern Sie den Algorithmus entsprechend.

Aufgabe 36 – IP-Adressen (1)

a) Sie wollen ein Netz mit 5000 Rechnern an das weltweite Internet anschließen. Wenn Sie eine IPv4-Klassenadresse (ohne CIDR) verwenden wollen – welche IP-Adress-Klasse beantragen Sie? Wie viele Prozent der für Sie reservierten Adressen nutzen Sie aus?

b) Die 5000 Rechner werden gleichmäßig auf 50 Abteilungen aufgeteilt. Setzen Sie eine Subnetz-Maske unter der Annahme, dass die Anzahl der Abteilungen in der Zukunft nicht mehr signifikant anwachsen wird. Die Anzahl der Rechner pro Abteilung ist jedoch schwer für die Zukunft abzuschätzen. Man erwartet hier noch einen erhöhten Rechnerbedarf und möchte möglichst flexibel bleiben.

c) Ihrem Antrag auf eine volle Klassen-Adresse wird nicht entsprochen. Stattdessen sollen klassenlose Adressen mit *CIDR* (*Classless Inter-Domain Routing*) vergeben werden. Um Kosten zu sparen, geben Sie die Subnetz-Struktur der Abteilungen auf. Wie viele Klasse-C-Adressen benötigen Sie? Welchen Block von Adressen bekommen Sie zugeteilt? Geben Sie ein Beispiel für einen gültigen Adress-Bereich.

Aufgabe 37 – Fragmentierung (1)

Es soll ein IP-Paket mit 2200 Bytes Nutzdaten übertragen werden. Das Paket muss fragmentiert werden, da es über Schicht-2-Netzwerke (LAN1-LAN4) transportiert wird, die nur kleinere Rahmen erlauben. Angegeben ist jeweils die Anzahl der Nutzdaten-Bytes, die ein IP-Paket haben darf:

Tabelle 15: Maximale Nutzlastbytes in verschiedenen LANs

LAN	LAN1	LAN2	LAN3	LAN4
max. Bytes Nutzlast	1400	512	1400	2400

Geben Sie die Nutzlast-Größe der Fragmente an, die beim Empfänger ankommen.

Aufgabe 38 – IP-Adressen (2)

a) Ordnen Sie die folgenden IP-Adressen in die zugehörigen IP-Adress-Klassen ein. Welche der Adressen haben eine besondere Bedeutung?

Tabelle 16: Beispiele für IP-Adressen

IP-Adresse	Adress-Klasse	Besonderheit
200.200.200.200		
85.6.200.1		
249.101.5.1		
192.168.5.3		
178.11.11.99		
127.0.0.1		
32.88.9.24		
172.20.111.23		
10.254.55.6		
225.3.3.64		

b) Gegeben folgende Netzwerkkonfigurationen bestehend aus Adresse und Subnetz-Maske. Der entsprechende Rechner schickt ein Paket an die angegebene Adresse. Geben Sie jeweils an, ob das Paket das Subnetz verlässt oder nicht.

Tabelle 17: IP-Paket-Übertragungen innerhalb und außerhalb des Subnetzes

IP-Adresse	Subnetz-Maske	Ziel-Adresse	Verlässt Subnetz?
132.176.67.44	255.255.255.0	132.176.67.200	ja/nein?
132.176.67.44	255.255.255.0	132.176.68.44	
201.20.222.13	255.255.255.240	201.20.222.17	
15.200.99.23	255.192.0.0	15.239.1.1	
172.21.23.14	255.255.255.0	172.21.24.14	
210.5.16.199	255.255.255.252	210.5.16.196	
210.5.16.199	255.255.255.252	210.5.16.195	
5.5.5.5	255.254.0.0	5.6.6.6	

Aufgabe 39 – Fragmentierung (2)

Ein IP-Paket sei folgendermaßen notiert: `Ident=n`, `M=b`, `Offset=m`, `Daten=d [g]`, wobei

- `n` die ID des Pakets
- `b` der Inhalt des More-Bits (0 oder 1)
- `m` der Offset des Dateninhalts im Paket
- `d [g]` der Datenteil des Pakets mit Länge `g`

a) Sie empfangen folgende IP-Fragmente. Wie lauten die entsprechenden nicht-fragmentierten IP-Pakete?

```
Ident=102, M=0, Offset=32, Daten="2222..." [256]
Ident=101, M=1, Offset=64, Daten="cccc..." [256]
Ident=101, M=0, Offset=196, Daten="aaaa..." [800]
Ident=103, M=1, Offset=100, Daten="ZZZZ..." [200]
Ident=101, M=1, Offset=0, Daten="dddd..." [512]
Ident=102, M=1, Offset=0, Daten="1111..." [256]
Ident=103, M=1, Offset=0, Daten="YYYY..." [800]
Ident=104, M=0, Offset=0, Daten="0000..." [32]
Ident=103, M=0, Offset=125, Daten="bbbb..." [200]
Ident=101, M=1, Offset=96, Daten="bbbb..." [800]
```

b) Sie empfangen folgende IP-Fragmente. Welche IP-Pakete fordern Sie nach?

```
Ident=105, M=0, Offset=196, Daten="aaaa..." [800]
Ident=102, M=1, Offset=0, Daten="1111..." [256]
Ident=101, M=1, Offset=0, Daten="XXXX..." [400]
Ident=103, M=1, Offset=0, Daten="3333..." [800]
Ident=105, M=1, Offset=0, Daten="dddd..." [512]
Ident=105, M=1, Offset=96, Daten="bbbb..." [800]
Ident=101, M=1, Offset=50, Daten="YYYY..." [400]
Ident=103, M=1, Offset=110, Daten="4444..." [120]
Ident=103, M=0, Offset=125, Daten="5555..." [200]
Ident=102, M=0, Offset=32, Daten="2222..." [256]
Ident=101, M=0, Offset=100, Daten="ZZZZ..." [200]
Ident=104, M=0, Offset=0, Daten="0000..." [32]
```

Aufgabe 40 – Die Internet-Schicht (1)

Beantworten Sie die folgenden Fragen stichwortartig:

a) Warum verwendet IP nicht MAC-Adressen als Ziel-Adressen?

b) Gibt es Rechner mit mehr als einer IP-Adresse?

c) Welche IP-Adressen haben besondere Bedeutungen?

d) Erklären Sie die Funktionsweise von ARP (*Address Resolution Protocol*).

e) Erklären Sie die Funktionsweise von NAT (*Network Address Translation*).

f) Welche Kritik besteht gegenüber NAT?

Aufgabe 41 – Die Internet-Schicht (2)

Beantworten Sie die folgenden Fragen stichwortartig:

a) Wieso könnte es passieren, dass *Traceroute* eine Liste von Zwischenschritten zu einem Ziel angibt, die gar nicht miteinander verbunden sind?

b) Wieso können mobile Rechner nicht ohne weitere Vorkehrungen ihre IP-Adresse im Fremdnetz behalten?

c) Wieso ist das DHCP (*Dynamic Host Configuration Protocol*) nicht geeignet für mobile Rechner, die selbst Dienste anbieten?

d) Wieso wird die Zuweisung von Adressen über DHCP in zwei Schritte `DISCOVER/OFFER` und `REQUEST/ACK` geteilt? Der DHCP-Server könnte doch direkt nach Erhalt des `DISCOVER`s die Adresse zuweisen und mit ACK quittieren.

e) Welche Sicherheitsprobleme entstehen durch DHCP?

f) Welches Hauptproblem von IPv4 soll mit IPv6 gelöst werden?

g) Beschreiben Sie die Funktionsweise von BGP (*Border Gateway Routing*).

Aufgabe 42 – IP-Adressen (3)

a) Geben Sie für folgende Netzwerkkonfigurationen die entsprechenden Subnetz-Masken an:

- Maximal viele Subnetze mit je 5 Rechnern in einem Klasse-B-Netz
- 50 Subnetze mit je 999 Rechnern in einem Klasse-B-Netz
- 12 Subnetze mit je 12 Rechnern in einem Klasse-C-Netz

b) Geben Sie für eine vorgegebene Adresse und Subnetz-Maske den Bereich von gültigen Rechneradressen an, die sich im selben Subnetz befinden. Den Adress-Bereich geben bitte an, indem Sie die kleinste und größte mögliche *Rechner*adresse des Subnetzes angeben.

Tabelle 18: Rechneradressen in Subnetzen

Rechneradresse	Subnetz-Maske	Kleinste Rechneradresse	Größte Rechneradresse
151.175.31.100	255.255.254.0		
151.175.31.100	255.255.255.240		
151.175.31.100	255.255.255.128		

Aufgabe 43 – Noch einmal Sliding Window

Beantworten Sie die folgenden Fragen stichwortartig:

a) Was sind die Hauptunterschiede der OSI-Schichten 2 und 4, die beim Einsatz von Sliding Window zu berücksichtigen sind?

b) Was ist der Unterschied zwischen Fluss- und Überlastkontrolle?

c) Was verbirgt sich hinter Slow Start und wozu wird es benötigt?

Aufgabe 44 – TCP allgemein

Beantworten Sie die folgenden Fragen stichwortartig:

a) TCP ist auf der Transportschicht angesiedelt. Die Sicherungsschicht stellt schon sicher, dass keine Rahmen verloren gehen. Wieso muss TCP dennoch mit Paketverlusten rechnen?

b) Wozu würde man typischerweise UDP einsetzen, wozu TCP?

c) Welche Dienstleistungen erbringt TCP für höhere Schichten?

d) Wann sendet TCP ein Segment aus?

e) Was sind duplicate ACKs und welchen Vorteil kann man durch ihre Auswertung erlangen?

Aufgabe 45 – TCP-Flusskontrolle

Einführende Erklärungen

Die TCP-Flusskontrolle arbeitet wie folgt:

- Ein Sender verwaltet einen Sendepuffer der Größe `MaxSendBuffer`. Er merkt sich in `LastByteAcked` das letzte Byte, das vom Empfänger bestätigt wurde (kumulative ACKs) und in `LastByteSent` das letzte gesendete (u.U. noch unbestätigte) Byte.
- Ein Empfänger verwaltet einen Empfangspuffer der Größe `MaxRcvBuffer`. Er merkt sich in `LastByteRead` das Byte, das von der lesenden Anwendung zuletzt gelesen wurde. `NextByteExpected` gibt die erste Lücke nach einer ununterbrochenen Folge empfangener Bytes an. `LastByteRcvd` gibt das letzte empfangene Byte an.
- Der Empfänger sendet mit jeder positiv empfangenen Nachricht ein kumulatives ACK mit einem `AdvertisedWindow` zurück. Dieses berechnet sich wie folgt:

  $$\texttt{AdvertisedWindow} = \texttt{MaxRcvBuffer} - (\texttt{LastByteRcvd} - \texttt{LastByteRead})$$

- Der Sender berechnet darauf

  $$\texttt{EffectiveWindow} = \texttt{AdvertisedWindow} - (\texttt{LastByteSent} - \texttt{LastByteAcked})$$

- Der Sender kann `EffectiveWindow` weitere Bytes senden, bevor der Empfänger im schlimmsten Fall einen Pufferüberlauf hat.

Aufgabe

In einem konkreten Fall seien die Puffer wie folgt konfiguriert:

- `MaxSendBuffer = 1000`
- `MaxRcvBuffer = 200`

Der Sendepuffer sei ständig durch die sendende Anwendung hinreichend gefüllt. Es gehen keine Pakete verloren und es findet keine Router-Überlastung statt. Es wird ausschließlich eine Flusskontrolle (ohne Überlastkontrolle) durchgeführt.

Zusätzlich soll folgende vereinfachende Änderung am TCP-Protokoll angenommen werden: pro Block von Paketen (d.h. pro dargestellter Zeile in der folgenden Tabelle) wird exakt eine Bestätigung vom Empfänger zurückgesendet. In der Realität würde TCP für jedes Paket im Advertised-Window eine *eigene* Bestätigung senden.

Vervollständigen Sie unter diesen Annahmen die folgende Tabelle.

Tabelle 19: Beispiel einer TCP-Sitzung

	Last Byte Acked	Effective Win	Send	Last Byte Sent	Last Byte Read	Last Byte Rcvd	Next Byte Exp	Advertised Win	ACK (kum; win)
1			(1..50)	50	0	50	51	150	(50;150)
2	50	150	(51..200)	?	10	?	201	?	?
3					20		211		
4					30	220	216(!)		
5					40		226		
6					50		241		
7					50(!)		251		
8					50		251		
9					60		252		
10					70		261		

Aufgabe 46 – TCP-Timeouts

Für die Berechnung der TCP-Timeouts gibt es zwei Formeln:

- Formel 1:

$$RTT \leftarrow RTT \cdot a + RTT_{last} \cdot (1-a)$$
$$Timeout_1 \leftarrow 2 \cdot RTT$$

- Formel 2 (Jacobson/Karels):

$$RTT \leftarrow RTT \cdot a + RTT_{last} \cdot (1-a)$$
$$Deviation \leftarrow Deviation \cdot a + |RTT_{last} - RTT| \cdot (1-a)$$
$$Timeout_2 \leftarrow RTT + 4 \cdot Deviation$$

a) Es sei a = 0,75. Gegeben sei eine zeitliche Abfolge von gemessenen Round-Trip-Zeiten RTT_{last}. Berechnen Sie die jeweiligen Timeouts nach den beiden Formeln.

Tabelle 20: Entwicklung der TCP-Timeouts

	RTT_{last}	RTT	$Timeout_1$	Deviation	$Timeout_2$
		200	400	10	240
1	200				
2	190				
3	180				
4	210				
5	200				
6	190				
7	210				
8	190				
9	210				
10	100				
11	90				
12	80				
13	90				
14	110				
15	100				
16	100				
17	90				
18	110				
19	100				
20	90				

b) Führen Sie die Berechnungen nun mit folgender Abfolge durch.

Tabelle 21: Entwicklung der TCP-Timeouts

	RTT_{last}	RTT	$Timeout_1$	Deviation	$Timeout_2$
		100	200	10	140
1	100				
2	50				
3	140				
4	60				
5	160				
6	100				
7	60				
8	50				
9	150				
10	100				
11	160				

12	100				
13	105				
14	90				
15	100				
16	105				
17	100				
18	95				
19	105				
20	100				

c) Stellen Sie die Ergebnisse von a) und b) jeweils grafisch dar. Interpretieren Sie die Kurvenverläufe.

Aufgabe 47 – Kryptographische Komponenten

Zeichnen Sie Architekturbilder für verschiedene Anwendungsfälle. Es stehen die folgenden kryptographischen Komponenten zur Verfügung:

- Verschlüsselungs-, Entschlüsselungsalgorithmus, Hashfunktion;
- Klartext, verschlüsselter Text, digitale Signatur;
- privater, öffentlicher, geheimer Schlüssel.

Dabei wird jeweils nur ein Teil der oben genannten Komponenten eingesetzt. Die Anwendungsfälle sind:

a) Ein Sender verschlüsselt symmetrisch, ein Empfänger entschlüsselt.

b) Ein Sender verschlüsselt asymmetrisch, ein Empfänger entschlüsselt.

c) Ein Sender signiert ein Dokument, ein Empfänger überprüft die Signatur.

Aufgabe 48 – Allgemeines über Sicherheit

Beantworten Sie die folgenden Fragen stichwortartig:

a) Nennen Sie mindestens drei Eigenschaften oder Zielsetzungen, die unter dem Begriff "Sicherheit" verstanden werden.

b) Was versteht man unter "Security through obscurity"?

c) Nennen Sie symmetrische und asymmetrische Verschlüsselungsverfahren.

d) Wie bezeichnet man die Schlüssel bei symmetrischen Verfahren, wie bei asymmetrischen Verfahren?

e) Bei welchem Angriff ist die Schlüssellänge von entscheidender Bedeutung?

f) Welche Funktionen werden für digitale Unterschriften verwendet?

g) Was verbirgt sich hinter dem Begriff "Nonce"?

Aufgabe 49 – Schlüssellänge

Ein Superrechner "Deep Thought" kann vielleicht einmal im Jahre 2050 pro Sekunde 10^{15} Schlüssel austesten. Ein neues symmetrisches Verschlüsselungsverfahren soll einem Brute-Force-Angriff von Deep Thought mindestens einen Tag widerstehen. Aus wie vielen Bits muss ein Schlüssel bestehen?

Aufgabe 50 – Wireless Equivalent Privacy

Einführende Erklärungen

Der erste Versuch ein Wireless-LAN-Netz zu verschlüsseln basierte auf dem Verfahren *Wireless Equivalent Privacy* (WEP). Dieses Verfahren gilt mittlerweile als sehr unsicher. Mit dieser Aufgabe soll ein bestimmter Angriff behandelt werden. Im Gegensatz zu den meisten Angriffen gegen die Vertraulichkeit soll hier ein Angriff gegen die Integrität von Rahmen durchgeführt werden.

Die Architektur der WEP-Verschlüsselung ist in der folgenden Abbildung dargestellt.

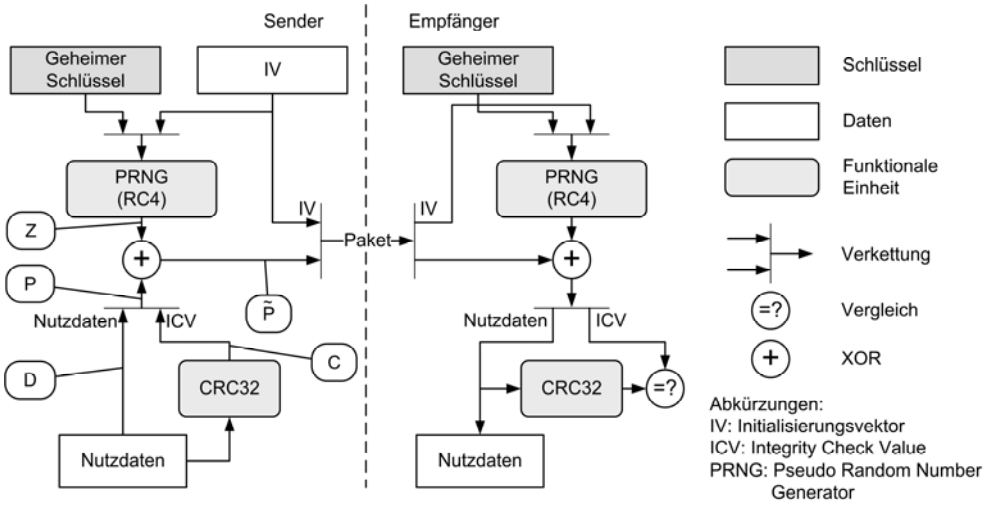


Abbildung 13: Architektur der WEP-Verschlüsselung

Aufgabe

Ein Angreifer fängt ein WEP-verschlüsseltes Paket ab und möchte unerkannt einige Bits des Pakets verändern. Die Bits an den Positionen, die verändert werden sollen, sind dem Angreifer im Klartext bekannt. Wie muss der Angreifer vorgehen, damit der Betrug unentdeckt bleibt, also insbesondere nicht durch eine falsche CRC-Prüfsumme auffällt?

Benutzen Sie folgende Notation:

- Sei $\oplus$ der XOR-Operator und $\|$ die Verkettung von Bitfolgen.
- Sei $P = D\|C$ das unverschlüsselte Paket mit D als Daten und $C = CRC(D)$ die CRC-Prüfsumme der Daten.
- Sei $\tilde{P} = \tilde{D}\|\tilde{C}$ das verschlüsselte Paket.
- Sei $\tilde{P}_2 = \tilde{D}_2\|\tilde{C}_2$ das gesuchte verschlüsselte Paket, das statt P unerkannt eingeschleust werden soll.
- Sei $Z = Z_D\|Z_C$ die Pseudozufallsfolge mit den Anteilen Z_D für die Daten und Z_C für die Prüfsumme.

Nutzen Sie die Eigenschaft der *Linearität* der CRC-Prüfsumme:

$$CRC(X \oplus Y) = CRC(X) \oplus CRC(Y) \oplus CRC(0)$$

Der Zusatz $\oplus CRC(0)$ sieht erst einmal überflüssig aus, da bei der CRC-Prüfsummenberechnung, so wie wir sie in Aufgabe 13 (Seite 8) eingeführt haben, $CRC(0) = 0$ gilt. Allerdings wird durch diese Formel berücksichtigt, dass ein CRC-Startwert 1…1 verwendet wird (siehe Seite 8). Diese Formel ist somit allgemeiner als die häufig dargestellte Formel $CRC(X \oplus Y) = CRC(X) \oplus CRC(Y)$. WEP verwendet übrigens das Generatorpolynom CRC-32 (Seite 8), was allerdings zur Lösung dieser Aufgabe nicht von Bedeutung ist.

Aufgabe 51 – Die RSA-Verschlüsselung

Einführende Erklärungen

Die Schlüssel für RSA werden folgendermaßen berechnet:

- Der Empfänger wählt zwei beliebige Primzahlen p, q. Diese müssen geheim bleiben.
- Man berechnet $n = p \cdot q$, $\Phi(n) = (p-1) \cdot (q-1)$.
- Man wählt ein e mit $ggT(e, \Phi(n)) = 1$, $1 < e < \Phi(n)$.
- Man berechnet d mit $d = e^{-1} \bmod \Phi(n)$, d.h., d ist das multiplikativ Inverse von e, also $d \cdot e \bmod \Phi(n) = 1$.
- e, n werden an den Sender veröffentlicht.

Eine Klartextnachricht M (aufgefasst als natürliche Zahl) wird zu einer Nachricht C verschlüsselt über

$$C = M^e \bmod n$$

Der Empfänger entschlüsselt C zum Klartext M durch

$$M = C^d \bmod n$$

Aufgabe

a) Geben Sie für die Primzahlen p = 3, q = 11 ein RSA-Schlüsselpaar an. Wählen Sie dazu ein e mit $e \leq 6$.

b) Ein Sender, dem Sie den Schlüssel aus Teil a) gegeben haben, möchte die Nachricht M = 15 an Sie versenden. Welche verschlüsselte Nachricht C erhalten Sie?

c) Sie erhalten eine verschlüsselte Nachricht C = 19. Wie lautete die ursprüngliche Nachricht M?

d) Sie fangen eine Übertragung C = 4 ab und wissen, dass der öffentliche Schlüssel (e, n) = (3, 15) ist. Versuchen Sie die Nachricht zu "knacken" und den Wert M des Klartextes zu ermitteln.

Aufgabe 52 – Domain Name System (1)

Einführende Erklärungen

Durch das *Domain Name System* des Internets wird zu einem symbolischen Rechnernamen eine IP-Adresse ermittelt. Die Nameserver bilden dabei ein verteiltes Verzeichnis. Die folgende Abbildung zeigt beispielhaft den Ablauf bei einer Anfrage.

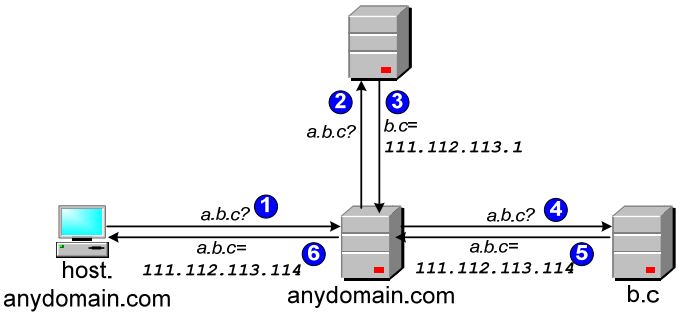


Abbildung 14: Notation der DNS-Anfragen und -Antworten

Dieses Beispiel dient nur zur Darstellung der Notation: Zu einer Anfrage (z.B. `a.b.c`?) gibt es jeweils eine Antwort der Form `Domain = IP-Adresse` (z.B. `b.c = 111.112.113.1`). Die Nummern an den Pfeilen geben zusätzlich die Reihenfolge der Nachrichten an. Man unterscheidet drei Antworttypen:

- *autoritativ* (Antwort 5): Diese werden gegeben, wenn ein Nameserver nach einem Namen gefragt wird, der in der lokalen Zonendatei aufgelistet ist.
- *iterativ* (Antwort 3): sind nicht die gewünschten Antworten, es wird aber ein "besserer" Nameserver vorgeschlagen (*forwarding*). Diese werden gegeben, wenn der Nameserver den gesuchten Namen nicht in der lokalen Zonendatei hat. Es wird nach dem größten Teilnamen (von rechts gelesen) gesucht, für den ein Nameserver-Eintrag lokal vorliegt.
- *rekursiv* (Antwort 6): ist die gewünschte Antwort, der Nameserver hat diese aber unter Zuhilfenahme anderer Nameserver generiert. Aus der Sicht des

Anfragers wird die Antwort direkt komplett geliefert (er sieht die weiteren Anfragen im Hintergrund nicht).

Lokale Nameserver sollten immer rekursive Antworten geben, da die Anfrager in der Regel iterative Antworten nicht verstehen. Root-Server geben keine rekursiven Antworten – das würde sie schnell überlasten.

Aufgabe

a) In der folgenden Situation möchte ein Host `host.pqr.de` die IP-Adresse des Hosts `www.xy.ab.eu` ermitteln. Es gibt 5 Nameserver, die die Antworttypen gemäß der folgenden Tabelle geben.

Tabelle 22: Antworttypen für die Nameserver

DNS für	pqr.de	root	eu	ab.eu	xy.ab.eu
Antworttyp	rekursiv	iterativ	rekursiv	iterativ	autoritativ

Alle Caches sind zunächst leer. Zeichnen Sie in die folgende Abbildung die Anfragen und Antworten gemäß der im Beispiel oben eingeführten Notation ein.

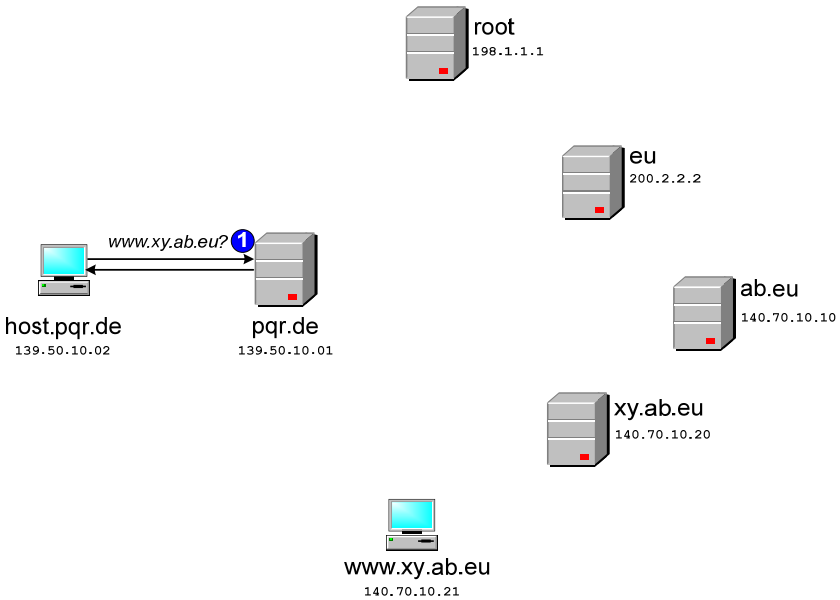


Abbildung 15: Ablauf eine DNS-Anfrage

b) Jetzt wird eine zweite Anfrage nach dem Rechner `mail.xy.ab.eu` durchgeführt. Gehen Sie davon aus, dass die Nameserver dieselben Antworttypen wie in Teil a)

geben. Gehen Sie weiter davon aus, dass die Caches durch die in Teil a) durchgeführten Anfragen gefüllt wurden und diese Cache-Werte jetzt berücksichtigt werden. Zeichnen Sie in die folgende Abbildung die Anfragen und Antworten ein.

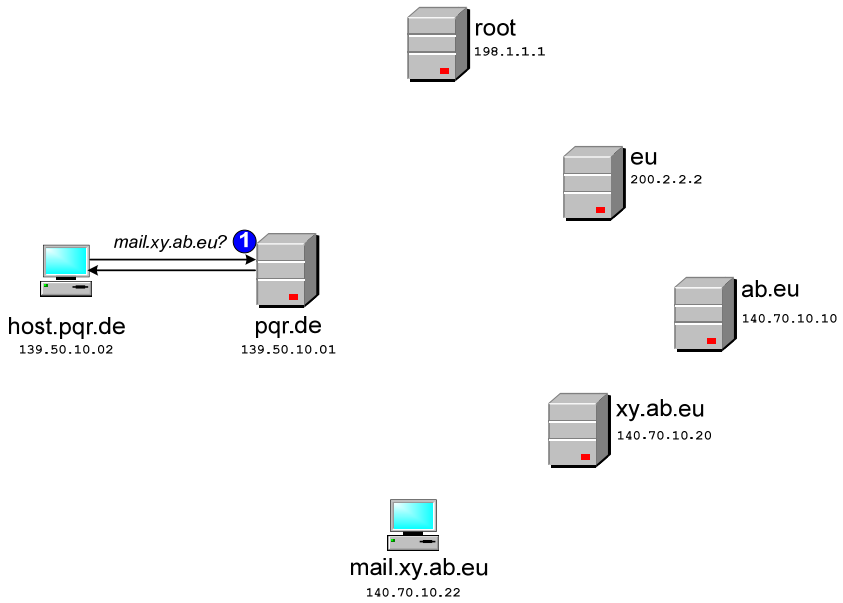


Abbildung 16: Ablauf einer weiteren DNS-Anfrage

Aufgabe 53 – Punycode

Einführende Erklärungen

Der *Punycode* zur Einbettung von internationalen Sonderzeichen in Domainnamen arbeitet wie folgt:

- Jedes Label (also jeder Namensbestandteil, zwischen den ".") wird separat kodiert.
- Labels *ohne* Sonderzeichen werden nicht modifiziert, damit man abwärtskompatibel zum traditionellen Verfahren bleibt.
- Labels *mit* Sonderzeichen werden durch "`xn--`" eingeleitet, gefolgt von dem Label, aus dem die Sonderzeichen entfernt wurden; danach folgt ein "-". Die Zeichenkette endet mit einer Kodierung, die Auskunft über Position und Art der Sonderzeichen gibt.

Als Beispiel:

`www.nürnberg.de` → `www.xn--nrnberg-n2a.de`

Die Übersetzung findet ausschließlich auf dem anfragenden Host statt (üblicherweise im Web-Browser). Das Domain Name System bleibt unverändert und arbeitet ausschließlich mit den übersetzten Domainnamen (ohne Sonderzeichen).

Für die Kodierung wird jedem erlaubten Sonderzeichen ein Zeichencode zugeordnet (beginnend bei 128). Wir beschränken uns hier auf die deutschen Sonderzeichen:

Tabelle 23: Kodierung von Sonderzeichen

Zeichen	Code
Ä	196
Ö	214
Ü	220
ä	228
ö	246
ü	252
ß	223

Wir stellen die Umformung an einem Beispiel dar. Wir beschränken uns dabei auf ein einzelnes Label: `nürnberg-wöhrd`. Erst wird eine Zeichenkette ohne diese Sonderzeichen gebildet, also `nrnberg-whrd`. Diese Zeichenkette hat 13 Einfügepositionen für Sonderzeichen (von 0 - 12):

	n		**r**		**n**	**b**	**e**	**r**	**g**	**-**	**w**	**h**		**r**	**d**	
0		1		2		3	4	5	6	7	8	9	10		11	12

Abbildung 17: Mögliche Einfügepositionen für Sonderzeichen

Wir wählen das einzufügende Sonderzeichen mit dem *kleinsten Code*. Gibt es mehrere Zeichen mit demselben kleinsten Code, wählen wir das Zeichen, das am weitesten links steht. In unserem Beispiel ist das erste Zeichen das "`ö`" an Position 9.

Aus dem Zeichen-Code kombiniert mit der Einfügeposition ermitteln wir eine Zahl gemäß dem folgenden Schema:

Tabelle 24: Kombinierte Definition von Einfügeposition und Sonderzeichen

Code	n	r	n	b	e	r	g	-	w	h	r	d	
128	0	1	2	3	4	5	6	7	8	9	10	11	12
129	13	14	15	16	17	18	19	20	21	22	23	24	25
…	…												
245	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533
246:ö	1534	1535	1536	1537	1538	1539	1540	1541	1542	**1543**			

Die Tabellezeilen beginnen bei 128 (dem kleinsten Code für Sonderzeichen) und enden mit dem Code des fraglichen Sonderzeichens (hier 246 für "ö").

Wir führen jetzt gedanklich diese Einfügung aus (d.h. wir haben danach 14 Einfügepositionen). Wir wählen das nächste Sonderzeichen (bei uns das "ü") und ermitteln wiederum einen Code. Allerdings beginnen wir die Zählung jetzt nicht mehr links oben (d.h. beim Code 128 und Position 0) sondern bei der letzten Einfügeposition. Aus diesem Grund müssen die einzufügenden Sonderzeichen auch gemäß Code und Position sortiert werden. Es ergibt sich folgendes Schema.

Tabelle 25: Kombinierte Definition von Einfügeposition und zweites Sonderzeichen

Code	n	r	n	b	e	r	g	-	w	ö	h	r	d	
246:ö											0	1	2	3
247	4	5	6	7	8	9	10	11	12	13	14	15	16	17
…	…													
251	60	61	62	63	64	65	66	67	68	69	70	71	72	73
252:ü	74	**75**												

Wären noch mehr Sonderzeichen übrig, würden wir den letzten Punkt solange wiederholen, bis alle abgearbeitet sind. In unserem Beispiel sind wir jetzt fertig.

Wir haben jetzt eine Reihe von Zahlen erhalten (in unserem Fall die zwei Zahlen 1543, 75); allgemein gibt es pro einzufügendem Sonderzeichen eine Zahl. Diese Zahlen werden durch die Zeichen "a"-"z" und "0"-"9" repräsentiert, die folgendermaßen kodiert sind:

Tabelle 26: Punycode-Kodierung der Zahlenwerte

a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
s	**t**	**u**	**v**	**w**	**x**	**y**	**z**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Zur Darstellung könnte prinzipiell eine 36-adische Entwicklung verwendet werden, also Wert = $\Sigma a_i \cdot 36^i$ für eine Folge von Ziffern a_i. Allerdings müsste man bei aufeinander folgenden Zahlen ein Trennzeichen einfügen (z.B. das "-"), d.h. für n kodierte Zahlen würde man n-1 zusätzliche Trennzeichen benötigen.

Um diese Trennzeichen zu vermeiden, wird ein anderes Verfahren gewählt: Für jede Stelle i wird eine *Wertigkeit* w_i und ein *Schwellwert* t_i definiert. Der Gesamtwert einer Teilfolge von Ziffern a_i ergibt sich analog zur b-adischen Entwicklung dann zu $\Sigma a_i \cdot w_i$, wobei nur für die letzte Ziffer $a_i < t_i$ gilt. Arbeitet man also eine Zeichenkette ab, kann man anhand von $a_i < t_i$ erkennen, dass eine Zahl zu Ende ist und mit dem nächsten Zeichen die neue Zahl beginnt. Dadurch wird das Trennzeichen vermieden. Die w_i sind jetzt nicht 36^i, sondern können auch kleinere Werte annehmen. Man kann zeigen, dass man jeden Zahlenwert auch unter diesen Bedingungen darstellen kann und in der Regel weniger Zeichen benötigt als mit der 36-adischen Entwicklung mit Trennzeichen.

Die w_i und t_i werden in dem Punycode-Verfahren nach einem komplizierten Verfahren pro Schritt (d.h. sowohl bei jeder Ziffer als auch bei jeder Zahl) geändert. Für Domainnamen mit einem *einzelnen Sonderzeichen* ergibt sich allerdings folgende Vereinfachung (die aber nur in den *meisten* Fällen zutrifft): $w_i = 35^i$, $t_i = 1$, d.h.

- der Wert der Zahl ergibt sich durch eine 35-adische Entwicklung (wobei die 1er-Stelle links und die höchstwertige Stelle rechts erscheinen).
- die Zahl ist zu Ende, wenn das Zeichen "a" erscheint.

Für unser Beispiel mit zwei Sonderzeichen gilt diese Vereinfachung nicht. Es ergibt sich (ohne Herleitung) die Zeichenfolge `djb9e`. Hierbei gilt:

- `djb` hat die Ziffernwerte (3, 9, 1) und den Gesamtwert $3+9 \cdot 35+1 \cdot 35^2 = 1543$
- `9e` hat die Ziffernwerte (35, 4) und den Gesamtwert $35+4 \cdot 10 = 75$ (d.h. die Basis wurde hier für die zweite Zahl von 35 auf 10 umgestellt).

Aufgabe

a) Geben Sie die Punycode-Darstellung der Domäne `www.wöhrd.de` an.

b) Welche Umlaut-Domäne verbirgt sich hinter `www.xn--netzwrk-e1a.com`?

Aufgabe 54 – Domain Name System (2)

Beantworten Sie die folgenden Fragen stichwortartig:

a) Woraus besteht das Domain Name System des Internets?

b) Nennen Sie die syntaktischen Regeln für den Aufbau gültiger Domainnamen.

c) Nennen Sie Kategorien, nach denen man Toplevel-Domains einteilen kann.

d) Nennen Sie einige Toplevel-Domains.

e) Nennen Sie einige inhaltliche Konventionen für Domainnamen.

f) Beschreiben Sie den DNS-Auflösungsmechanismus.

g) Nennen Sie die wichtigsten DNS-Recordtypen.

h) Nennen Sie die Antwort-Typen von Nameservern.

Aufgabe 55 – URIs, URNs, URLs

Beantworten Sie die folgenden Fragen stichwortartig:

a) Beschreiben Sie, was URIs, URNs und URLs sind.

b) Beschreiben Sie den Aufbau von URLs.

c) Geben Sie die wichtigsten Protokoll-Einträge von URLs an.

d) Geben Sie für die wichtigsten Protokolle Beispiele für komplette URLs an. Versuchen Sie dabei die maximale Anzahl unterstützter Felder für ein Protokoll abzudecken.

Aufgabe 56 – Peer-to-Peer vs. Client-Server

Tragen Sie möglichst viele Unterschiede zwischen dem Client-Server- und dem Peer-to-Peer-Paradigma zusammen.

Aufgabe 57 – Chord (1)

Einführende Erklärungen

Chord ist ein prominenter Vertreter eines Peer-to-Peer-Protokolls, das auf der Idee der *Distributed Hashtable* (DHT) basiert. Die Idee von DHTs ist wie folgt:

- Eine Menge von Ressourcen (z.B. Dateien) ist verteilt gespeichert, d.h. jeder Rechner des Netzwerk speichert nur einen kleinen Anteil aller Ressourcen.
- Jeder Ressource i ist ein Schlüssel Key_i zugeordnet, z.B. indem eine kryptographische Hashfunktion auf den Dateinamen angewendet wird.
- Eine Distributed Hashtable bildet jeden Schlüssel auf die Rechneradresse ab, unter der diese Ressource gespeichert ist.
- Jeder Rechner speichert nur einen Teil der gesamten Hashtable.

Möchte ein Rechner auf eine bestimmte Ressource zugreifen, so muss er erst den Rechner finden, der den entsprechenden Ausschnitt der Hashtable speichert. Erst dann erfährt er die Rechneradresse unter der die Ressource zugreifbar ist.

Chord setzt die Idee der Distributed Hashtable wie folgt um:

- Es gibt im Wesentlichen die Operationen
  - `insert(key, adr)`: Generieren eines Eintrags
  - `lookup(key)` → `adr`: Suchen nach Eintrag
  - `join(adr), leave(adr)`: Eintreten, Verlassen

- Schlüssel und Rechneradressen werden über eine Hashfunktion auf eine Menge $[0,2^m-1]$ abgebildet.
- Jeder Rechner ist für eine zusammenhängende Teilmenge [start, end] $\subseteq [0,2^m-1]$ zuständig (mod 2^m). Hierbei bildet sich end aus dem Hashwert der eigenen Rechneradresse. Der start-Wert ist end+1 des Nachbarn mit dem nächstkleineren Bereich.
- Jeder Rechner verwaltet eine *Fingertabelle*, die einen Bereich finger.start … finger.end auf eine Rechneradresse node abbildet. Diese Tabelle wird für Werte $k \in \{1,\ldots m\}$ wie folgt belegt:
  - finger[k].start = end+2^{k-1} mod 2^m
  - finger[k].end = end+2^k-1 mod 2^m
  - finger[k].node = erstes n mit finger[k].start≤n.end

Die Verweise sind in der folgenden Abbildung dargestellt.

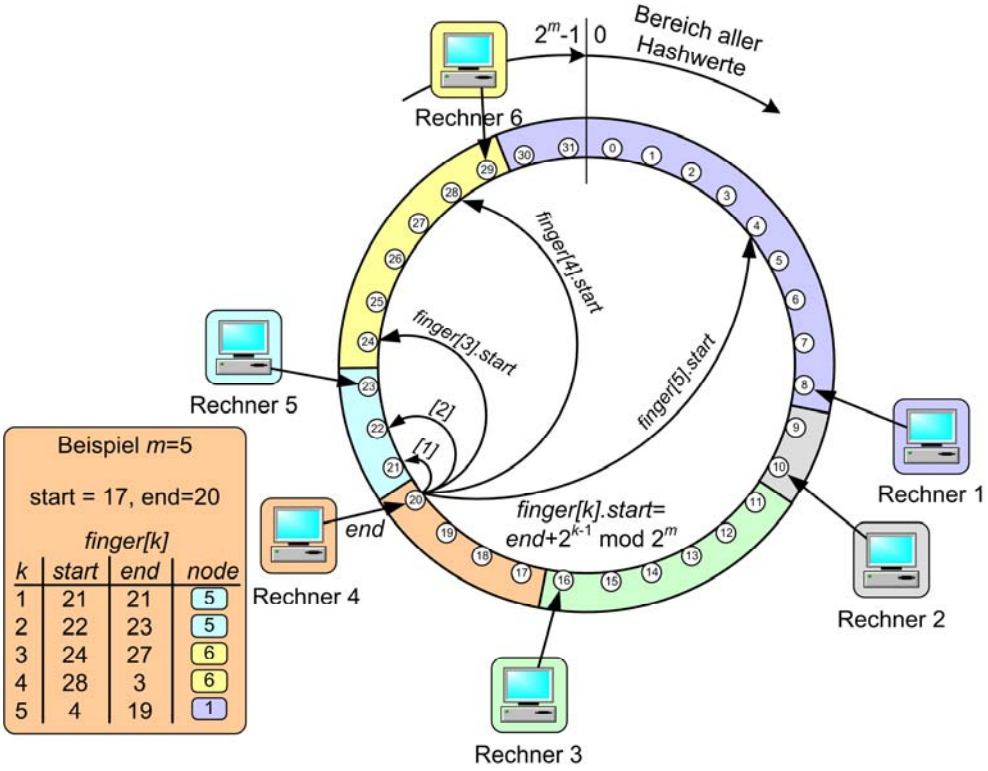


Abbildung 18: Aufbau der Chord-Finger-Tabelle

Die Operation `lookup(key)` → `adr` wird wie folgt realisiert:

- Setze adr auf die Rechneradresse des Rechners, der die Suche ausführt.
- *Schleifenanfang*: Wenn key ∈ [adr.start, adr.end]: fertig (Rechner adr hat den gesuchten Hasheintrag).
- Suche das k mit key ∈ [adr.finger[k].start, adr.finger[k].end] – hierzu muss man Kontakt zu Rechner adr aufnehmen.
- Setze adr ← adr.finger[k].node, gehe zum Schleifenanfang.

Tritt ein neuer Rechner dem Netzwerk bei (`join`), belegt er einen Bereich [start, end], der bisher von einem anderen Rechner verwaltet wurde. Die entsprechenden Einträge müssen von diesem kopiert werden. Darüber hinaus müssen alle Finger-Einträge anderer Rechner, die auf diesen Bereich verweisen, aktualisiert werden.

Aufgabe

In einem Chord-P2P-Netzwerk sei m = 5, d.h. die Hashwerte sind aus [0,...31]. Es nehmen 4 Rechner an dem Netzwerk teil. Die Hashwerte der Rechneradressen sind:

- Rechner A: 8
- Rechner B: 24
- Rechner C: 12
- Rechner D: 29

a) Vervollständigen Sie die folgende Tabelle, indem Sie die Einträge zur Spalte "start" festlegen.

Tabelle 27: Zugeordnete Index-Bereiche

Rechner	start	end
A		8
B		24
C		12
D		29

b) Bei der Suche nach einem Hashwert konsultiert ein Rechner seine Finger-Tabelle. Dort findet er zu Hashwerten einen geeigneten Rechner, der bei der Suche als nächstes gefragt werden soll. Ein Tabelleneintrag besteht aus

- k: der Index des Tabelleneintrags;
- start, end: der Teilbereich aller Hashwerte, für die ein anderer Rechner abgefragt wird;
- node: die Rechneradresse, die bei Hashwerten aus dem Bereich start…end als nächstes abgefragt wird.

Vervollständigen Sie die Finger-Tabellen für die Rechner A und C.

Tabelle 28: Finger-Tabellen von Rechner A und C

A			
k	**start**	**end**	**node (Rechner)**
1	9	9	
2			
3			
4			
5			

C			
k	**start**	**end**	**node (Rechner)**
1			
2			
3			
4			
5			

Aufgabe 58 – Die Peer-to-Peer-Idee

Beantworten Sie die folgenden Fragen stichwortartig.

a) Nennen Sie mögliche Anwendungsbeispiele für Peer-to-Peer-Netze.

b) Beschreiben Sie die Idee der Distributed Hashtables.

c) Geben Sie die Laufzeit-Komplexität der Chord-Operationen `join` und `lookup` an.

Aufgabe 59 – Chord (2)

In einem Chord-P2P-Netzwerk sei m = 5, d.h. die Hashwerte sind aus [0,...31]. Es nehmen 4 Rechner an dem Netzwerk teil. Die Hashwerte der Rechneradressen sind:

- Rechner 1: 10
- Rechner 2: 23
- Rechner 3: 12
- Rechner 4: 29

a) Stellen Sie die Finger-Tabellen für alle 4 Rechner auf.

b) Rechner 2 möchte gerne die Datei mit dem Hashwert 11 laden. Stellen Sie die Anfragen dar, die zur Suche der entsprechenden Rechneradresse notwendig sind.

c) Führen Sie eine analoge Suche von Rechner 3 nach Hashwert 30 aus.

Aufgabe 60 – Base64

Einführende Erklärungen

Mit *Base64* können beliebige Binärdaten als Zeichenketten kodiert werden. Da druckbare Zeichen verwendet werden, kann man so Binärdaten als Texte z.B. in Emails oder über HTTP übertragen.

Der Eingabestrom wird in 24-Bit-Blöcke zerlegt, die durch jeweils vier 6-Bit-Worte dargestellt werden. Ein 6-Bit-Wort wird durch ein Zeichen der folgenden Tabelle kodiert.

Tabelle 29: Darstellung von 6-Bit-Blöcken in Base64

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
Q	R	S	T	U	V	W	X	Y	Z	a	b	c	d	e	f
32	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
48	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
w	x	y	z	0	1	2	3	4	5	6	7	8	9	+	/

Bezüglich der Länge der Binärdaten ergeben sich folgende Fälle:

- Es fehlt ein Byte, um auf eine durch drei teilbare Anzahl zu kommen: es werden zwei 0-Bits angefügt; der letzte Block hat damit 18 Bits, die über 3 Zeichen kodiert werden. Zusätzlich fügt man ein "=" an das Resultat an.
- Es fehlen zwei Bytes, um auf eine durch drei teilbare Anzahl zu kommen: es werden vier 0-Bits angefügt; der letzte Block hat damit 12 Bits, die über 2 Zeichen kodiert werden. Zusätzlich fügt man "==" an das Resultat an.
- Die Zahl der Bytes ist durch drei teilbar: die Kodierung erfolgt ohne Zusatz.

Aufgabe

a) Kodieren Sie folgende Strings mit Base64. Die einzelnen Zeichen sind ASCII-7-Bit kodiert, ohne 0-Terminierung, abgelegt mit 1 Byte/Zeichen:

- `ABCD`
- `abcde`

b) Dekodieren Sie die folgenden Base-64-Nachrichten:

- `QW50d29ydA==` (das 4. Zeichen ist eine Null)
- `UmVjaG5lcm5ldHpl` (das 8., 5.-letzte und letzte Zeichen sind ein kleines "`L`")

Zur Abbildung der ASCII-Codes auf die Zeichen verwenden Sie folgende Tabelle.

Tabelle 30: ASCII-Tabelle der druckbaren Zeichen

Nr.	Zchn	Nr.	Zchn	Nr.	Zchn	Nr.	Zchn	Nr.	Zchn	Nr.	Zchn
32	blank	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o		

Aufgabe 61 – Darstellung von Daten

Beantworten Sie die folgenden Fragen stichwortartig.

a) Nennen Sie einige Probleme bei der Darstellung von Anwendungsdaten auf Übertragungswegen.

b) Nennen Sie Vorteile von dem Verpacken von Daten als Text. Was sind die Nachteile?

c) Wie werden MIME-Typen klassifiziert. Welche Basistypen kennen Sie? Nennen Sie Beispiele für MIME-Typen.

d) Erläutern Sie das Request-Response-Paradigma am Beispiel von SMTP.

Aufgabe 62 – V-Formate

Einführende Erklärungen

Die so genannten *V-Formate* sind entwickelt worden, um strukturierte Daten im mobilen Umfeld zu übertragen. Die populärsten V-Formate sind:

- *vCard*: elektronische Visitenkarten, Adressen, Telefonnummern
- *vCalendar*: Kalendereinträge

Die Formate sind wie folgt aufgebaut:

- Alle Objekte werden als Text dargestellt.
- Objekte sind in `BEGIN:`*xxx* und `END:`*xxx* eingeschlossen (*xxx* ist z.B. `VCARD`).
- In einem Objekt befinden sich Einträge der Form

  Eigenschaftsname;Param$_1$*...;Param*$_n$*:Wert*

  z.B. `TEL;PREF;WORK;VOICE:+49 911 5880 1169`
- Jeder Eintrag umfasst eine Zeile; sind mehrere Zeilen erforderlich, muss der Zeilentrenner `\` oder `=0D=0A` verwendet werden.

Ein Beispiel einer vCard sieht folgendermaßen aus:

```
BEGIN:VCARD
VERSION:2.1
N;CHARSET=ISO-8859-1:Roth;Jörg
ADR:;; Computer Science Department\
Kesslerplatz 12;Nuremberg;;90489;Germany
ORG:University of Nuremberg
TEL;PREF;WORK;VOICE:+49 911 5880 1169
TEL;WORK;FAX:+49 911 5880 5666
EMAIL;WORK;INTERNET:Joerg.Roth@Ohm-hochschule.de
UID:1900581
END:VCARD
```

Beim vCalendar-Format sollten insbesondere Wiederholungseinträge möglich sein, um regelmäßig wiederkehrende Ereignisse ausdrücken zu können. Feste Zeitpunkte werden im Format

<Jahr><Monat><Tag>T<Stunde><Minute><Sekunde>

angegeben. Sollen nur Tage ohne Uhrzeit beschrieben werden, wird der Bereich ab "`T`" weggelassen. Wiederholungseinträge sind beispielhaft in der folgenden Tabelle dargestellt:

Tabelle 31: Kodierung der Wiederholungseinträge im vCalendar-Format

Zeichenkette	Bedeutung
D1 #10	täglich für 10 Tage
D1 20101231	täglich bis zum 31.12.2010
D2 #0	jeden 2. Tag, für immer
W1 #10	jede Woche, 10 Wochen lang
W2 MO FR 20101231	jede 2. Woche montags und freitags, bis zum 31.12.2010
MP1 1+ FR #0	jeden 1. Freitag im Monat, für immer
MP1 1- MO #0	jeden letzten Montag im Monat, für immer
MD1 4+ #0	jeder 4. im Monat, für immer
MD1 1- #0	jeder Monatsletzte im Monat, für immer
YD3 2 #10	alle drei Jahre am 2. Jan. (2. Tag im Jahr), für 10 Jahre
YM1 3 #5	jedes Jahr im März (3. Monat), für 5 Jahre

Einzelne Termine, regelmäßige Wiederholungen sowie Listen von Ausnahmen können kombiniert werden. Hierzu bietet vCalendar folgende Notation an.

Tabelle 32: vCalendar-Notation für die Kombination von Termineinträgen

vCalendar Eigenschafts-name	Bedeutung
DTSTART	Erster Zeitpunkt des Ereignisses
DTEND	Letzter Zeitpunkt des Ereignisses
RDATE	Explizite Liste von Tagen, an denen ein Ereignis eintritt
RRULE	Wiederholungseinträge gemäß einer Wiederholungsregel
XDATE	Tage, die Ausnahmen von der Wiederholungsregel darstellen
XRULE	Ausnahmeregeln für Wiederholungsregeln

Ein Beispiel eines vCalendar-Eintrags sieht folgendermaßen aus:

```
BEGIN:VCALENDAR
VERSION:1.0
BEGIN:VEVENT
DTSTART:20101231
RRULE:MD12 #0
DESCRIPTION:Silvester
UID:6201394
END:VEVENT
END:VCALENDAR
```

Aufgabe

a) Geben Sie für folgende Person eine vCard an:

- Fritz Mustermann, Netzweg 1a, 99999 Nirwana
- arbeitet bei NetComp
- Telefon (dienstlich): 0123/4567; Telefon (privat): 0123/890, Fax: 0123/111
- E-Mail: fritz.mustermann@netcomp.com

b) Geben Sie für folgendes Ereignis ein vCalendar-Objekt an:

Jeden 2. Freitag im Monat, vom 14. Mai 2010, die folgenden 10 Monate, außer am 11. Juni, jeweils von 18:00-19:00 Uhr: "Tennis".

Aufgabe 63 – Javas Object Serialization

Einführende Erklärungen

Die Programmiersprache Java bietet ein komfortables Verfahren genannt *Object Serialization*, um beliebige Objekte über ein Netzwerk transportieren zu können. Java kennt dabei die Struktur von Objekten und kann sequentiell alle Variablen eines Objektes serialisieren. Hierbei geht Java wie folgt vor:

- Durchlaufe alle Variablen des Objektes.
- Eine Variable von einem elementaren Datentyp (z.B. `int`, `boolean`) wird anhand eines festen Schemas serialisiert, das Typ und Inhalt kodiert.
- Ist die Variable wieder ein Objekt, so wird die Serialisierung rekursiv auf dieses Objekt angewendet. Hierbei wird die Klasse des Objektes notiert.
- Wird beim rekursiven Abstieg ein Objekt gefunden, das schon einmal serialisiert wurde, wird dieses nicht noch einmal serialisiert, sondern nur die Objektkennung gesendet. Dadurch können auch zirkular verzeigerte Strukturen serialisiert werden.

Wir erläutern das Vorgehen an einem Beispiel:

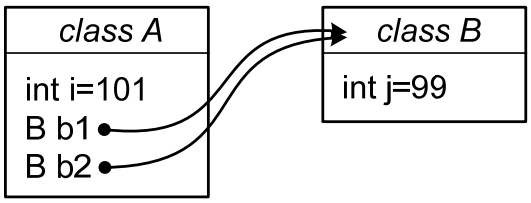


Abbildung 19: Beispiel für die Objekt-Serialisierung

Ein Objekt der Klasse A soll serialisiert werden. Es hat drei Variabeln, wovon zwei mit derselben Instanz der Klasse B belegt sind. Die Übertragung wird in der folgenden Tabelle dargestellt.

Tabelle 33: Übertragung eines serialisierten Objektes

Daten	Bedeutung
AC ED	Magiccode für Object Stream
00 05	Versionskennung des Formats
73	TC_OBJECT
72	TC_CLASSDESC
00 01 41('A')	Klassenname "A"
EB 95 02 49 8E FC C3 93	Handle
02	Flags: SC_SERIALIZABLE
00 03	Anzahl der Variablen im Objekt
49('I')	Basistyp int
00 01 69('i')	Variablenname "i"
4C('L')	Kein Basistyp sondern Objekt
00 02 62('b') 31('1')	Variablenname "b1"
74	TC_STRING
00 03 4C('L') 42('B') 3B(';')	Klassendeskriptor: "L": non-array, "B": Klassenname, ";" wird immer angefügt
4C('L')	Kein Basistyp sondern Objekt
00 02 62('b') 32('2')	Variablenname "b2"
71	TC_REFERENCE
00 7E 00 01	Referenz auf Klassenname "B"
78	TC_ENDBLOCKDATA
70	TC_NULL
00 00 00 65(101dez)	integer 101
73	TC_OBJECT
72	TC_CLASSDESC
00 01 42('B')	Klassenname "B"
99 36 81 B8 06 0C E9 B8	Handle
02	Flags: SC_SERIALIZABLE
00 01	Anzahl der Variablen im Objekt
49(I)	Basistyp int
00 01 6A('j')	Variablenname "j"
78	TC_ENDBLOCKDATA
70	TC_NULL
00 00 00 63(99dez)	integer 99
71	TC_REFERENCE
00 7E 00 04	Referenz auf Objekt

Dieses Beispiel soll nur einen groben Eindruck der Funktionsweise der Objekt-Serialisierung vermitteln und nicht das Format komplett beschreiben.

Aufgabe

Sie empfangen folgende Bytes über einen ObjectInputStream von Java:

Tabelle 34: Übertragung eines serialisierten Objektes (Aufgabe)

Daten	Bedeutung
AC ED	
00 05	
73	
72	
00 06 41('A') 43('C') 6C('l') 61('a') 73('s') 73('s')	
8B 30 8E B8 FD 86 20 55	
02	
00 01	
49('I')	
00 05 61('a') 6E('n') 49('I') 6E('n') 74('t')	
78	
70	
00 00 00 16	

Wie könnte die Klasse der Instanz aussehen? Welche Belegung hatten die Variablen dieser Instanz?

Aufgabe 64 – XML (1)

a) Definieren Sie eine XML-Datei, die Informationen über Kinofilme speichert. Sie soll die folgenden Informationen enthalten:

- Titel,
- Jahr der Produktion,
- Hauptdarsteller,
- Fortsetzung, wenn vorhanden.

Die XML-Datei soll die folgenden Einträge speichern:

- *Die dunkle Bedrohung* (1999), Liam Neeson, Ewan McGregor, Natalie Portman
- *Angriff der Klonkrieger* (2001), Ewan McGregor, Natalie Portman, Hayden Christensen
- *Die Rache der Sith* (2005), Ewan McGregor, Natalie Portman, Hayden Christensen

Der zweite und dritte Film sind Fortsetzungen des ersten bzw. zweiten Films.

b) Die XML-Datei hat u.U. noch viele doppelte Einträge von z.B. Schauspielern. Wie könnte man die XML-Datei so formulieren, dass möglichst keine Redundanzen vorhanden sind.

Aufgabe 65 – XML (2)

Erzeugen Sie die passenden XML-Schema-Dateien zu den XML-Dateien aus Aufgabe 64 (Seite **Fehler! Textmarke nicht definiert.**).

Aufgabe 66 – Sockets

Schreiben Sie zwei Java-Programme, die über Sockets kommunizieren:

- Ein Programm `Server.java` öffnet einen Server-Socket, nimmt zwei int-Werte entgegen, multipliziert diese und sendet einen int-Wert mit dem Produkt zurück.
- Ein Programm `Client.java` verbindet zu dem Server-Socket, sendet zwei int-Werte und empfängt als Antwort das Produkt beider Zahlen.

Aufgabe 67 – XML-RPC

Ein fernes Objekt obj definiert die Methode

```
int doAction(int i, String s, int[] ar)
```

a) Über XML-RPC wird ein Aufruf mit den Parametern `i=50`, `s="hallo"` und `ar=[5, 6, 99]` gemacht. Wie sieht die entsprechende HTTP-Anforderung aus?

b) Wie sieht eine HTTP-Antwort mit dem Rückgabewert 123 aus.

Aufgabe 68 – Webservices (1)

Beantworten Sie die folgenden Fragen stichwortartig.

a) Was ist die Motivation für Webservices?

b) Welche Typen von Webservices kann man unterscheiden?

c) Nennen Sie die Schritte bei der Generierung von Laufzeitkomponenten von Webservices mit SOAP/WDSL.

d) Beschreiben Sie den REST-Ansatz.

Aufgabe 69 – CORBA

Schreiben Sie zwei Java-Programme, die über CORBA kommunizieren:

- Ein Server bietet eine entfernte Methode `public int mult(int a, int b)`, die zwei Zahlen multipliziert.
- Ein Client nutzt diese Methode.
- Stellen Sie neben den Programmen auch die Interface-Beschreibung (IDL-Datei) dar.
- Trennen Sie beim Server die angebotene Methode von dem Teil, der die CORBA-Umgebung aufbaut.

Aufgabe 70 – Webservices (2)

Schreiben Sie zwei Java-Programme, die über Webservice kommunizieren:

- Ein Server bietet eine entfernte Methode `public int mult(int a, int b)`, die zwei Zahlen multipliziert.
- Ein Client nutzt diese Methode.
- Nutzen Sie die Java Annotations `@WebService` und `@WebServiceRef`, um den Code möglichst kompakt zu gestalten.

Aufgabe 71 – Webservices (3)

Eine Webservice-Anwendung soll die Bestellung in einem Online-Warenhaus abwickeln. Hierzu gibt es folgende Ressourcen:

- Warenkörbe, identifiziert durch eine eindeutige Warenkorb-Nummer;
- Artikel, identifiziert durch eine eindeutige Artikel-Nummer.

Der Anwender soll folgende Funktionen durchführen können:

- Abfragen des Warenkorbs;
- Legen eines Artikels in den Warenkorb;
- Löschen eines Artikels aus dem Warenkorb;
- Abfragen eines Artikels;
- Hinzufügen eines Rezensionstextes zu einem Artikel;
- "Zur Kasse gehen" mit dem aktuellen Warenkorb.

Die Realisierung soll mit einem REST-Webservice erfolgen. Geben Sie eine mögliche Zuordnung der HTTP-Kommandos zu Funktionen an. Definieren Sie auch, wie die Ressourcen in den Anfragen bezeichnet werden könnten.

Aufgabe 72 – Kodierung von Bitfolgen (3)

Entwickeln Sie eine neuartige "2B/4B-Codierung" mit folgenden Eigenschaften:

- Jeweils 2 Bits sollen durch 4 Bits dargestellt werden.
- In der Ausgabe sollen immer exakt genauso viele 0- und 1-Bits vorkommen.
- Neben den Daten sollen auch noch zwei Kombinationen "Start Frame" und "Stop Frame" dargestellt werden.

Bemerkung: es gibt verschiedene xB/yB-Kodierungen (4B/5B, 5B/6B, 8B/10B, 64B/66B um nur einige zu nennen – die 2B/4B-Codierung existiert aber nur in dieser Aufgabe.

a) Entwickeln Sie ein Codierung, die diese Eigenschaften erfüllt

b) Entwickeln Sie eine Digitalschaltung, die zwei Datenbits (b_1, b_0) sowie zwei Steuerleitungen *start* und *stop* auf 4 Code-Bits (y_3, y_2, y_1, y_0) abbildet:

- *start*=1 soll den Start Frame Code erzeugen; dann sind *stop*, b_1, und b_0 jeweils 0.
- *stop*=1 soll den Stop Frame Code erzeugen; dann sind *start*, b_1, und b_0 jeweils 0.
- Für Daten-Codes sind *start* und *stop* jeweils 0.

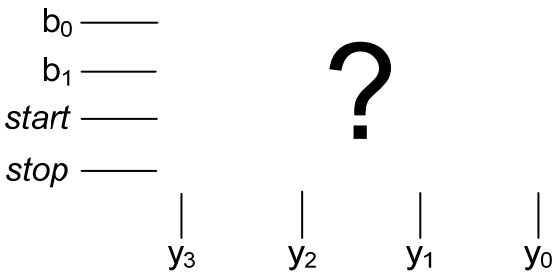


Abbildung 20: Schema der Schaltung zur 2B/4B-Codierung

Aufgabe 73 – Kodierung von Bitfolgen (4)

Kodieren Sie die Bitfolge `1000 0000 0100 0010`

a) mit AMI, b) mit B8ZS, c) mit HDB3.

Stellen Sie jeweils den Signalverlauf grafisch dar.

Aufgabe 74 – Kodierung von Bitfolgen (5)

Zeigen Sie für HDB3, dass die Anzahl zwischen +1- und -1-Signalen sich maximal um 2 unterscheiden kann.

Hinweise:

- Solange keine vier 0-Bits in Folge kommen, wechseln sich 1- und -1-Signale ständig ab. Dadurch alleine kann der Unterschied maximal 1 betragen.
- Das Problem könnten die verschiedenen Varianten sein, vier 0-Bits in Folge zu kodieren. Zeigen Sie hierbei eine Obergrenze für den Unterschied, der durch die jeweilige Variante entstehen kann. Zeigen Sie darüber hinaus, welche Varianten aufeinander folgen können.

Aufgabe 75 – TCP/IP-Prüfsummen

a) Berechnen Sie die TCP/IP-Prüfsumme zu den 16-Bit-Zahlen

`1238, 8761, 1316, 47484, 983, 17987`

(die 16-Bit-Zahlen sind jeweils als vorzeichenloser Integer-Wert dargestellt)

b) Berechnen Sie die TCP/IP-Prüfsumme zu den 16-Bit-Zahlen

`61416, 464, 47856, 7874, 32165, 45687, 10021, 8985, 65501, 57458, 41234`

c) In der Einer-Komplement-Arithmetik gibt es zwei Darstellungen des Wertes 0, nämlich `0…0` und `1…1`. In TCP-Nachrichten und IP-Paketen wird die Prüfsumme, so gebildet, dass die neue Gesamtsumme 0 ist. Der Empfänger bildet also die Gesamtsumme inklusive dem Prüfsummenfeld. Argumentieren Sie, dass nach der Einer-Komplement-Bildung immer `0…0` und nie `1…1` resultieren kann, obwohl die `1…1` auch dem Wert 0 entspricht.

d) Wieso kann ein Router bei der Weiterleitung eines IP-Pakets durch die Dekrementierung des TTL-Feldes einfach das Prüfsummenfeld inkrementieren?

Aufgabe 76 – OLSR (2)

Gegeben ist folgendes Netzwerk:

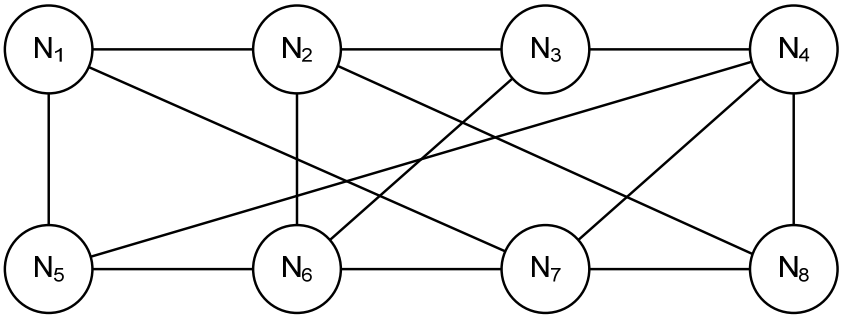


Abbildung 21: Netzwerk zur Berechnung von MPR(N_1)

Berechnen Sie die Menge MPR(N_1).

Aufgabe 77 – Link Reversal Routing

Gegeben folgendes Netzwerk, das Link Reversal Routing verwendet. Die Pfeile geben die Richtungen im gerichteten azyklischen Graphen an.

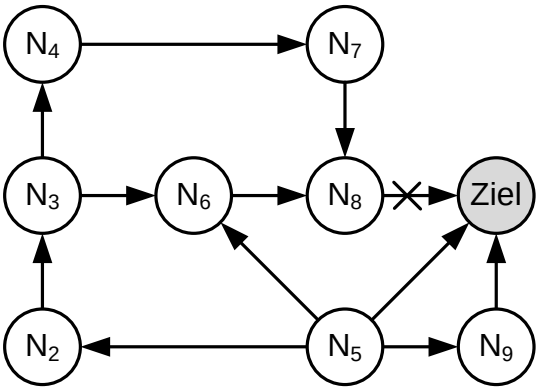


Abbildung 22: Netzwerk zum Link Reversal Routing

a) Führen Sie das Full Reversal aus und zeigen Sie das Netzwerk nach jedem Schritt.

b) Führen Sie das listenbasierte Partial Reversal aus.

Aufgabe 78 – IPv6

a) Wie viele verschiedene Möglichkeiten für 64-Bit-Interface-Identifier gibt es?

b) Wie viele verschiedene Möglichkeiten für 64-Bit-Interface-Identifier gibt es, die aus einer EUI-48 MAC-Adresse gebildet werden?

c) In einem Subnetz mit 1000 Hosts wählt ein neuer Host zufällig einen 64-Bit-Interface-Identifier aus. Wie groß ist die Wahrscheinlichkeit, dass ein neuer Host einen schon existierenden Interface Identifier gewählt hat? Gehen Sie davon aus, dass alle Hosts des Subnetzes eine zufällig gewählten Interface Identifier benutzen (also nicht aus einer MAC-Adresse gebildet).

d) Ein Service-Provider bekommt von der RIR einen /32 Adressblock zugeteilt. Wie vielen Unternehmen kann der Service-Provider maximal einen Adressblock zuteilen, wenn jedes Unternehmen noch einmal 16 Bit Subnetze einrichten möchte?

Aufgabe 79 – TCP allgemein (2)

Beantworten Sie die folgenden Fragen stichwortartig:

a) Was ist der *Three-Way-Handshake* im Zusammenhang mit TCP? Was wird vereinbart?

b) Worin unterscheidet sich der TCP-Verbindungsabbau vom -aufbau?

c) Was ist eine *halboffene* Verbindung?

d) Was ist der TCP-*Pseudo-Header* und wozu wird er verwendet?

Aufgabe 80 – DNS Reverse Lookup

a) Was verbirgt sich hinter dem Reverse Lookup von DNS? Wieso müssen zusätzliche Mechanismen für diese Abbildung eingerichtet werden?

b) Welche Verzeichnisse existieren auf Server-Seite, um einen Reverse Lookup zu ermöglichen.

c) Ein Benutzer möchte den DNS-Namen hinter der IP-Adresse 141.75.201.12 wissen. Welche Schritte werden zur Auflösung durchgeführt?

Aufgabe 81 – Rainbow Tables

a) Was ist die Aufgabe von Rainbow Tables?

b) Sei die Passwort-Menge die Menge der natürlichen Zahlen {0,…99}.

Für ein x bestehend aus den Ziffern x_1, x_0 (d.h. $x=x_1 \cdot 10+x_0$) sei H gegeben durch

$$H(x)=(11 \cdot x_1 + 13 \cdot x_0) \text{ modulo } 100$$

Diese "Hashfunktion" erfüllt offensichtlich keine einzige sinnvolle Eigenschaft kryptographischer Hashfunktionen – sie dient hier nur zur Verdeutlichung des Verfahrens. Sei weiter

Reduk(x)=(x+1) modulo 100.

Bilden Sie eine Rainbow Table für H. Berechnen Sie dazu Ketten mit 4 Hashwerten für die Anfangswerte 1, 3 und 20.

Start	**H**	**Reduk**	**H**	**Reduk**	**H**	**Reduk**	**H**
1							
3							
20							

Was wird hiervon gespeichert?

c) Sie möchten gerne das x ermittelt für das gilt H(x)=61. Wie gehen Sie vor?

d) Was kann man gegen Angriffe über Rainbow-Tables tun? (Kurze Beschreibung des Ansatzes)