

Grundlagen Rechnernetze und Verteilte Systeme

GRNVS (IN0010)

Sommersemester 2016

Prof. Dr.-Ing. Georg Carle

Johannes Naab, Stephan Günther, Maurice Leclaire

Grundlagen Rechnernetze und Verteilte Systeme

SoSe 2016 – vorläufige Version des Handouts

Prof. Dr.-Ing. Georg Carle

Johannes Naab, Stephan M. Günther, Maurice Leclaire

Lehrstuhl für Netzarchitekturen und Netzdienste
Fakultät für Informatik
Technische Universität München

31.03.2016

Grundlagen Rechnernetze und Verteilte Systeme

Kapitel 0 – Organisatorisches, Überblick und Schichtenmodelle

Kapitel 0: Überblick und Schichtenmodelle

Die Vorlesung im Überblick

Zusammenfassung der einzelnen Kapitel

Geschichte des Internets

Von der Entstehung bis zum heutigen Internet

Schichtenmodelle

Was sind Schichtenmodelle?

Wozu sind Schichtenmodelle gut?

Das ISO/OSI-Modell

Zusammenfassung der einzelnen Kapitel (Stand 8. April 2015)

Kapitel 1: Physikalische Schicht

1. Signale, Information und deren Bedeutung
 - ▶ Was sind Signale?
 - ▶ Entropie und Information
2. Klassifizierung von Signalen
 - ▶ Zeit- und Frequenzbereich
 - ▶ Abtastung, Rekonstruktion und Quantisierung
3. Übertragungskanal
 - ▶ Einflüsse des Übertragungskanals auf Signale
 - ▶ Kapazität eines Übertragungskanals (Modell)
4. Nachrichtenübertragung
 - ▶ Quellen- und Kanalkodierung
 - ▶ Impulsformung
 - ▶ Modulation
5. Übertragungsmedien
 - ▶ Elektromagnetisches Spektrum
 - ▶ Koaxialleiter
 - ▶ Twisted-Pair-Kabel
 - ▶ Lichtwellenleiter

Zusammenfassung der einzelnen Kapitel

Kapitel 2: Sicherungsschicht

1. Darstellung von Netzwerken als Graphen

- ▶ Netztopologien
- ▶ Adjazenz- und Distanzmatrix
- ▶ Shortest Path Tree und Minimum Spanning Tree

2. Verbindungscharakterisierung, Mehrfachzugriff und Medienzugriffskontrolle

- ▶ Serialisierungs- und Ausbreitungsverzögerungen
- ▶ Nachrichtenflussdiagramme
- ▶ ALOHA und Slotted ALOHA
- ▶ CSMA, CSMA/CD und CSMA/CA
- ▶ Token Passing

3. Rahmenbildung, Adressierung und Fehlerkennung

- ▶ Erkennung von Rahmengrenzen und Codetransparenz
- ▶ Adressierung und Fehlererkennung
- ▶ Fallstudie: IEEE 802.3u (FastEthernet)
- ▶ Fallstudie: IEEE 802.11a/b/g/n (Wireless LAN)

4. Verbindungen auf Schicht 1 und 2

- ▶ Hubs, Bridges und Switches
- ▶ Collision und Broadcast Domains

Zusammenfassung der einzelnen Kapitel

Kapitel 3: Vermittlungsschicht

1. Vermittlungsarten

- ▶ Leistungsvermittlung
- ▶ Nachrichtenvermittlung
- ▶ Paketvermittlung

2. Adressierung im Internet

- ▶ Internet Protocol (IP)
- ▶ Adressauflösung (ARP)
- ▶ Internet Control Message Protocol (ICMP)
- ▶ Adressklassen (für Classful Routing)
- ▶ Subnetting und Präfixe (für Classless Routing)

3. Routing

- ▶ Statisches Routing
- ▶ Longest Prefix Matching
- ▶ Dynamisches Routing
- ▶ Algorithmen von Bellman-Ford und Dijkstra
- ▶ Routingprotokolle (Distance Vector und Link State)
- ▶ Autonome Systeme

4. Nachfolge von IPv4: IPv6

Zusammenfassung der einzelnen Kapitel

Kapitel 4: Transportschicht

1. Aufgaben der Transportschicht
2. Multiplexing durch Port-Nummern
3. Verbindungslose Übertragung: UDP
 - ▶ Case-Study: UDP
 - ▶ Code-Study: SOCK_DGRAM (C)
4. Verbindungsorientierte Übertragung: TCP
 - ▶ Sliding-Window-Protokolle (Go-Back-N und Selective Repeat)
 - ▶ Case-Study: TCP (Fluss- und Staukontrolle)
 - ▶ Code-Study: SOCK_STREAM (C)
5. Network Address Translation (NAT)

Zusammenfassung der einzelnen Kapitel

Kapitel 5: Die Schichten 5 – 7

1. Schichten
 - ▶ Vor- und Nachteile verschiedener Schichtenmodelle
2. Sitzungsschicht
 - ▶ Dienste
 - ▶ Funktionseinheiten
 - ▶ Synchronisation
 - ▶ Quality of Service
 - ▶ Performance Parameter
3. Darstellungsschicht
 - ▶ Datenkompression (Huffman Code)
4. Anwendungsschicht
 - ▶ Namensauflösung im Internet (DNS)
 - ▶ HTTP
 - ▶ SMTP

Zusammenfassung der einzelnen Kapitel

Kapitel 6: Verteilte Systeme

1. Homogene, skalierbare Paradigmen

- ▶ Message Passing Interface (MPI)
- ▶ MapReduce
- ▶ Pipes, netcat, DUP

2. Remote Procedure Call

- ▶ Funktionsaufrufe und Parameterkodierung
- ▶ Stubs, IDL, Binding
- ▶ Java RMI
- ▶ RPC/RMI

3. Shared Memory

- ▶ NUMA (Non-Uniform Memory Access)
- ▶ Virtueller Speicher
- ▶ Auslagerung
- ▶ Distributed Shared Memory
- ▶ Konsistenz in parallelen Programmen

4. Einbettung in Programmiersprachen

- ▶ Erlang
- ▶ Actor Model

Zurück zu Kapitel 0:

Die Vorlesung im Überblick

Zusammenfassung der einzelnen Kapitel

Geschichte des Internets

Von der Entstehung bis zum heutigen Internet

Schichtenmodelle

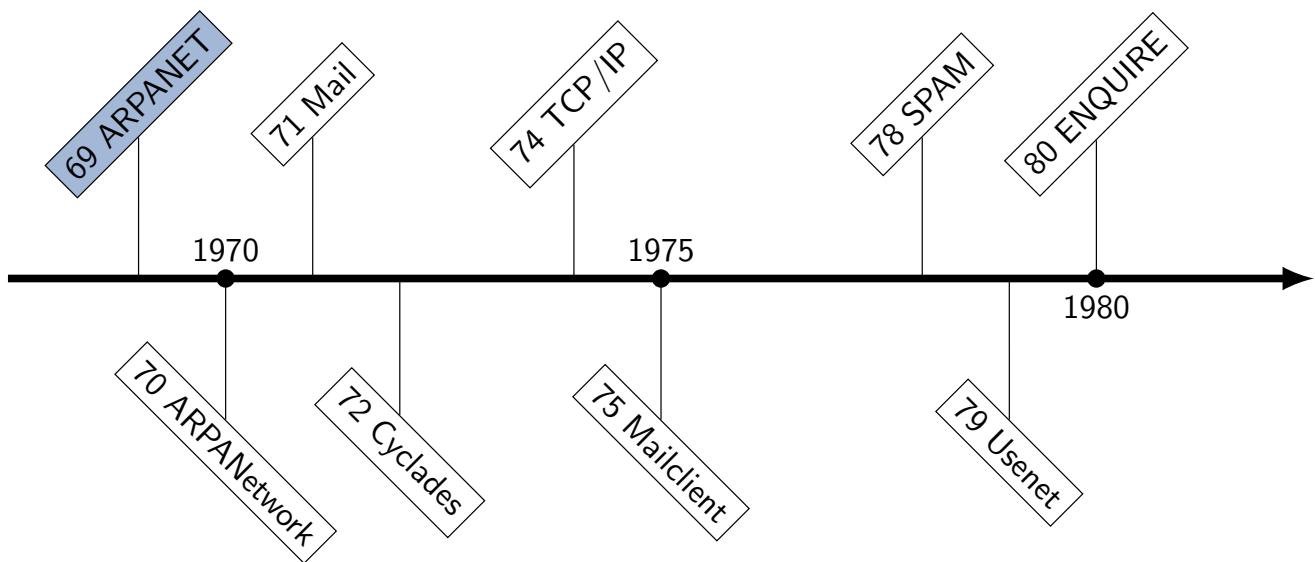
Was sind Schichtenmodelle?

Wozu sind Schichtenmodelle gut?

Das ISO/OSI-Modell

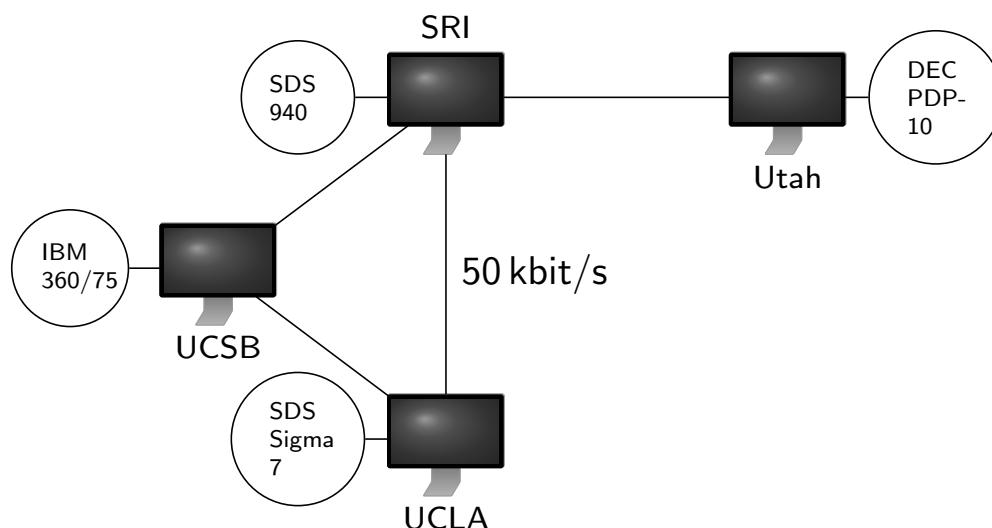
Von der Entstehung bis zum heutigen Internet

Geschichte des Internets: Übersicht bis 1980

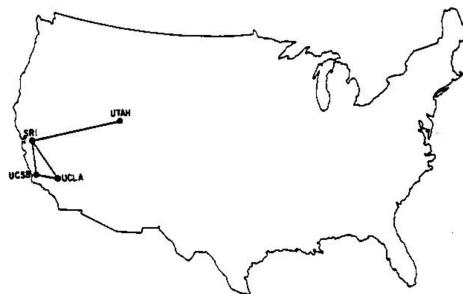


ARPANET mit den ersten 4 Knoten

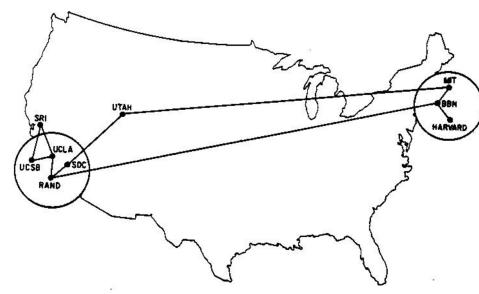
- ▶ University of California, Los Angeles (UCLA) 1.9.1969
- ▶ Stanford Research Institute (SRI) 1.10.1969
- ▶ UC Santa Barbara (UCSB) 1.11.1969
- ▶ University of Utah 12.1969



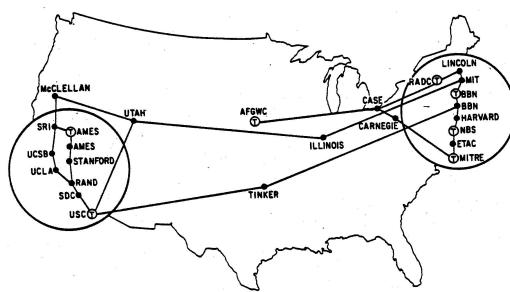
ARPANET von 1969 bis 1977



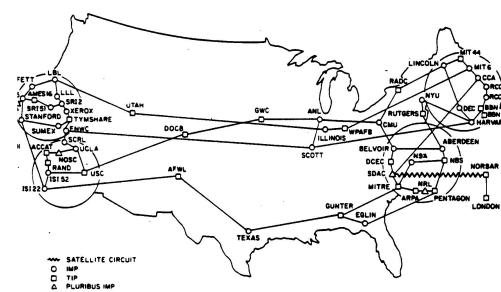
ARPANET 1969, 4 Knoten



ARPANET 1970, 9 Knoten

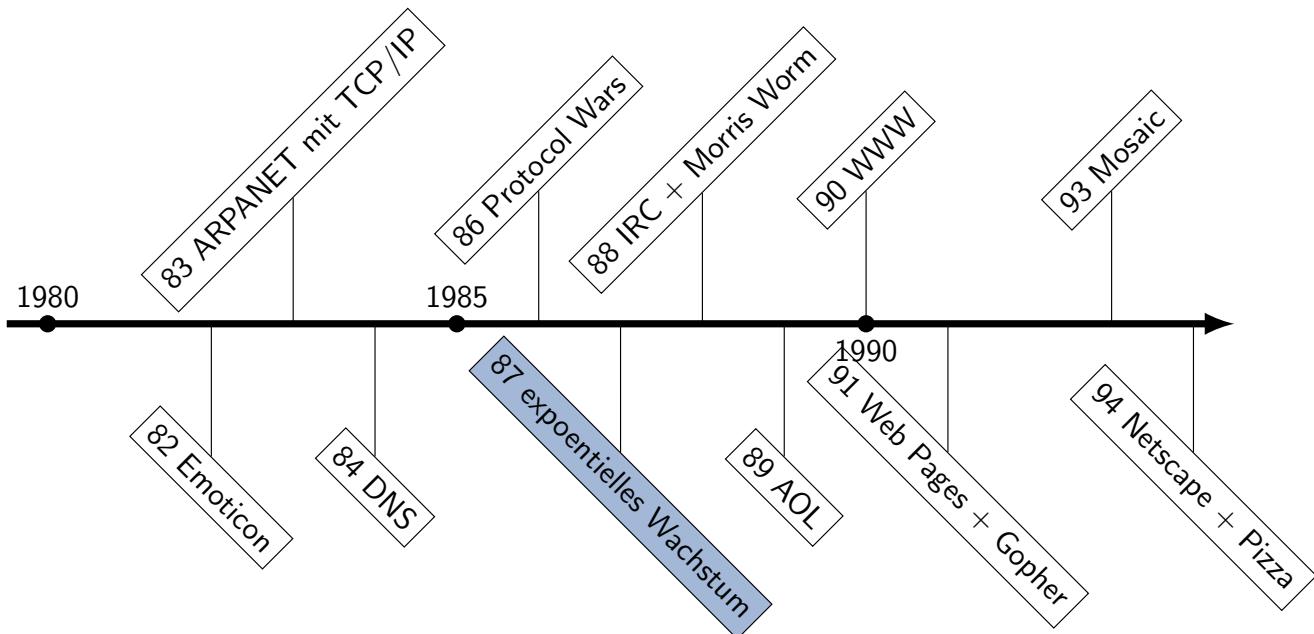


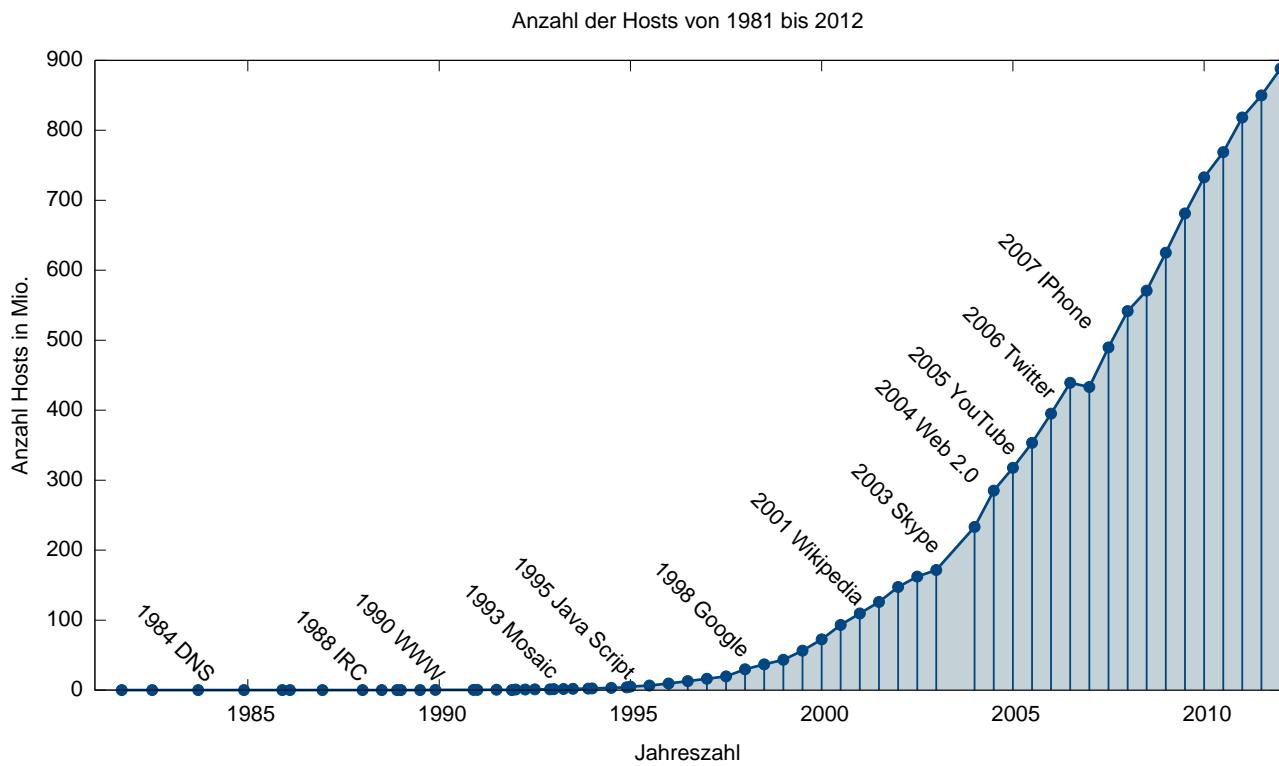
ARPANET 1972, 25 Knoten



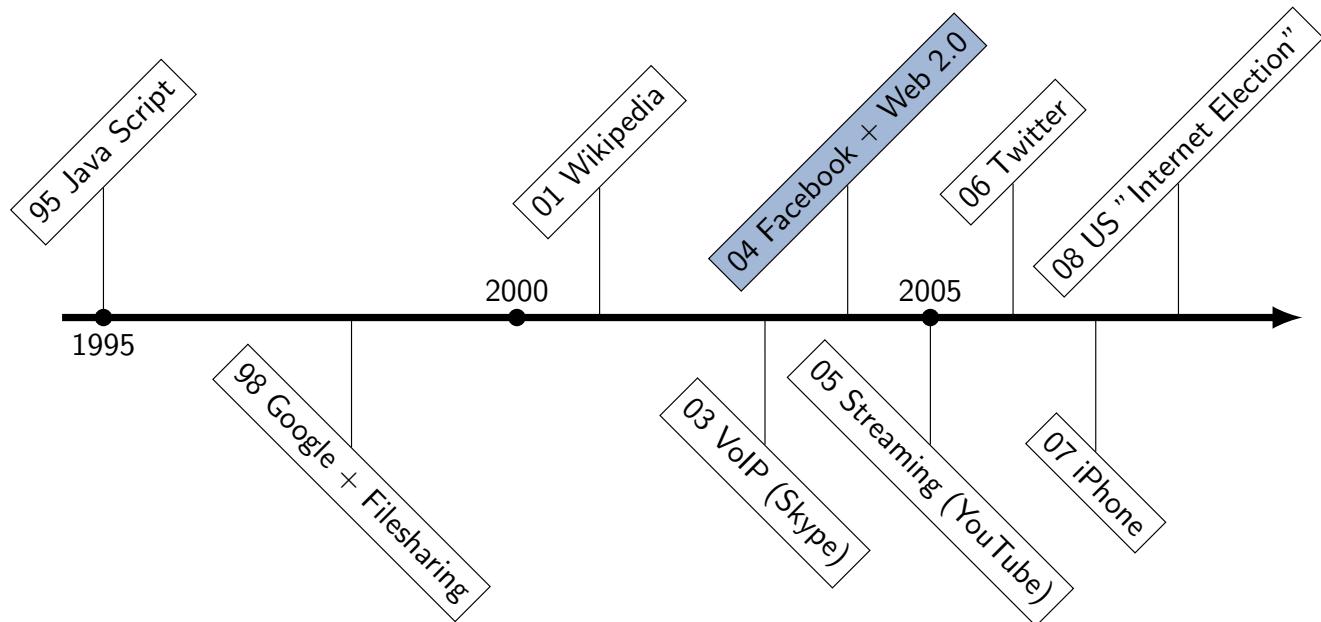
ARPANET 1977, 58 Knoten

Geschichte des Internets: Übersicht von 1980 bis 1994

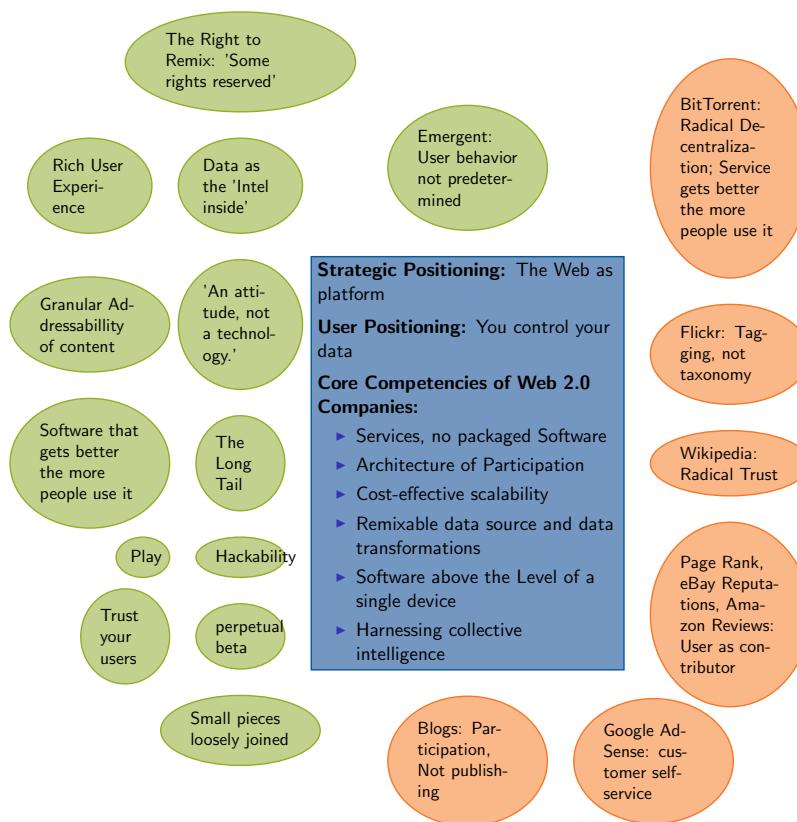




Geschichte des Internets: Übersicht ab 1994



Web 2.0 Meme Map, by Tim O'Reilly [2]



Das Internet heute



Abbildung: Verbindungen zwischen autonomen System im Internet (2007) [1]

Das Internet heute

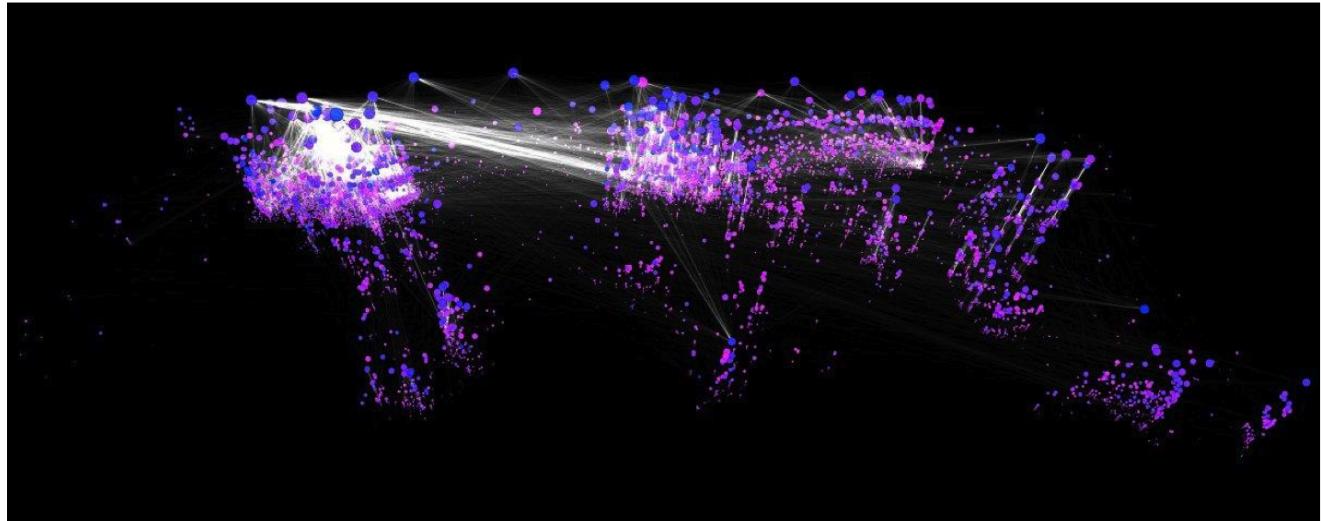


Abbildung: Blau hervorgehoben lang existierende autonome Systeme [1]

Inhalt

Die Vorlesung im Überblick

Zusammenfassung der einzelnen Kapitel

Geschichte des Internets

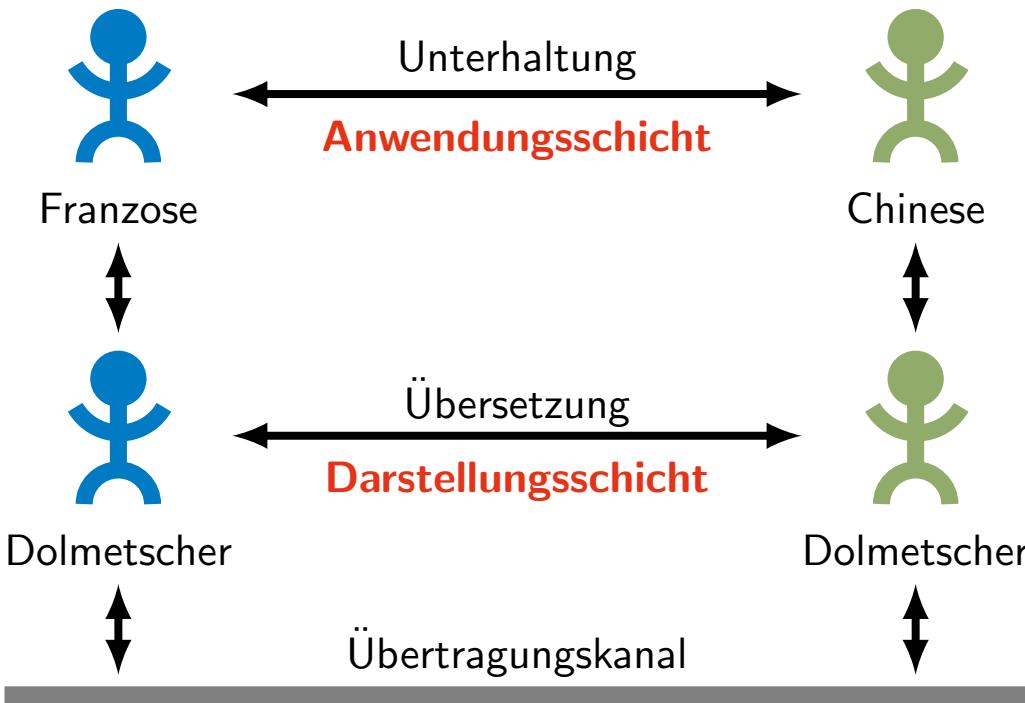
Von der Entstehung bis zum heutigen Internet

Schichtenmodelle

- Was sind Schichtenmodelle?
- Wozu sind Schichtenmodelle gut?
- Das ISO/OSI-Modell

Was sind Schichtenmodelle?

Ein einfaches Beispiel:



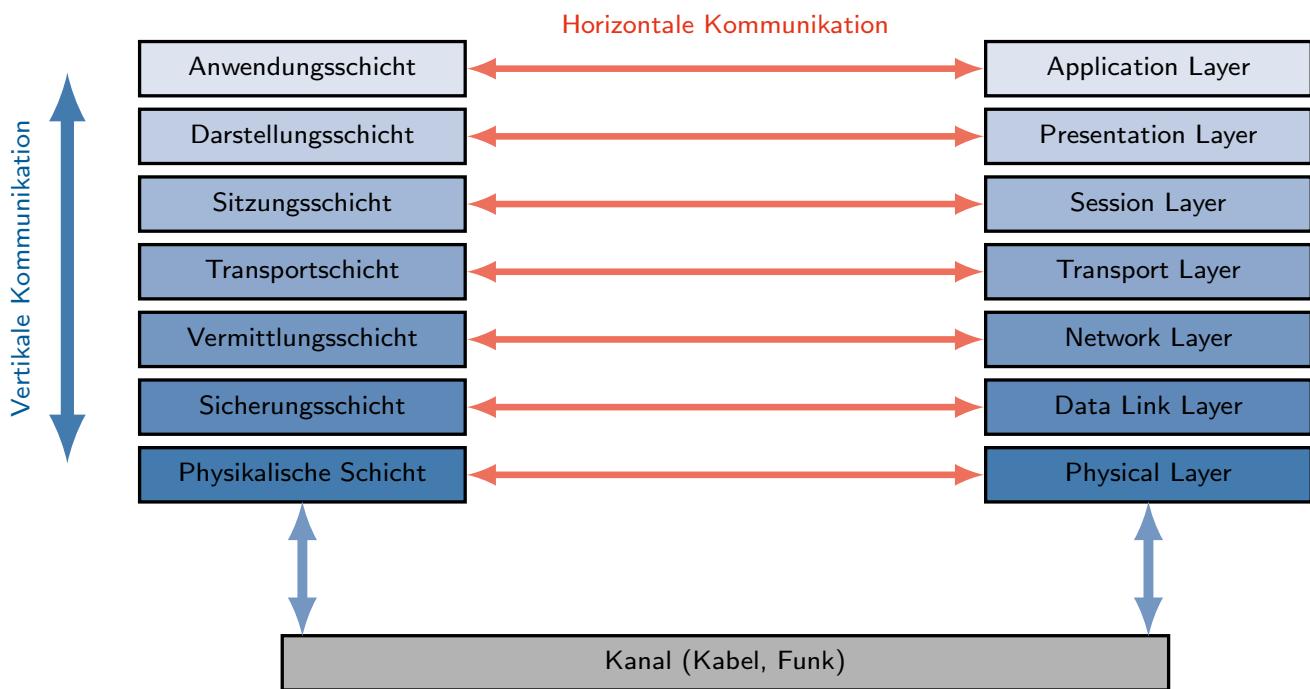
Wozu sind Schichtenmodelle gut?

- ▶ Unterteilung des komplexen Kommunikationsvorgangs
 - ▶ Niedrigere Schichten **bieten** höheren Schichten **Dienste an**
 - ▶ Höhere Schichten **nehmen Dienste** der jeweils niedrigeren Schicht in **Anspruch**
- ▶ Abstraktion von der Implementierung einer Schicht
 - ▶ Festlegung, **welche** Dienste angeboten werden, aber **nicht wie** sie erfüllt werden
 - ▶ Austauschbarkeit einzelner Implementierungen
- ▶ Anwendbar auf beliebige Kommunikationsvorgänge

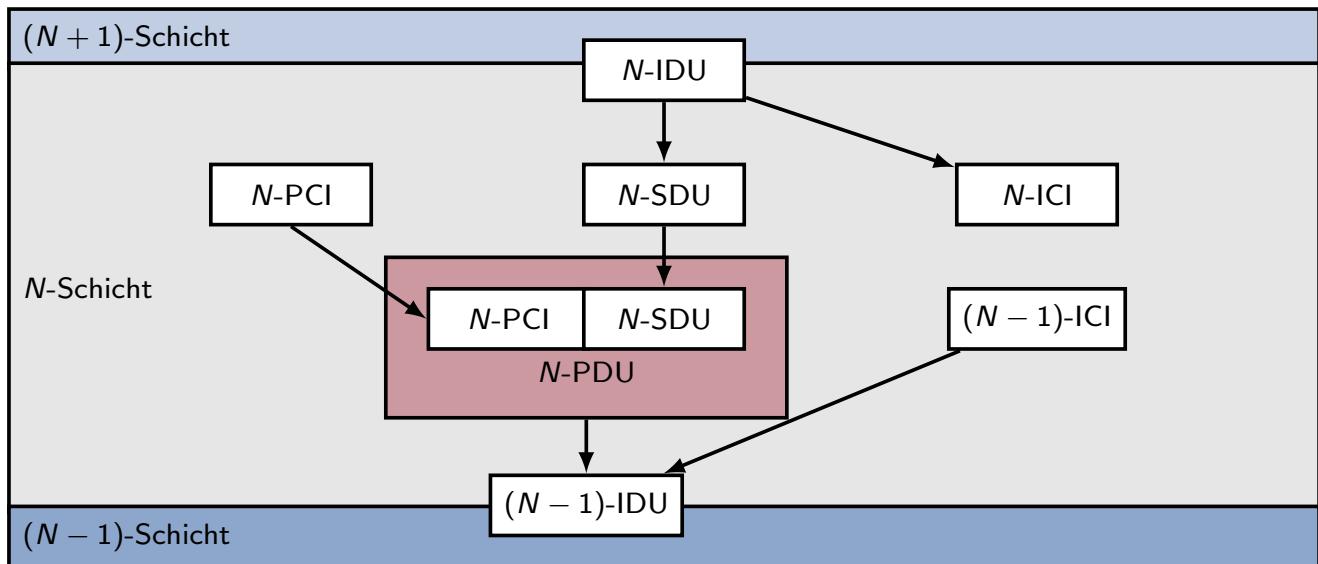
Das ISO/OSI-Modell

- ▶ Entwickelt zwischen 1979 und 1983 von der *International Organization for Standardization (ISO)*
- ▶ OSI = *Open Systems Interconnect*
- ▶ Unterteilt den Kommunikationsvorgang in **7 Schichten**
- ▶ Jede Schicht erbringt bestimmte Dienste (z. B. Aufteilen einer Nachricht in kleinere Pakete)
- ▶ Keine Aussage, wie diese Dienste zu erbringen sind

Schematische Darstellung des OSI-Modells:



Datenaustausch zwischen Schichten



Die $(N+1)$ -Schicht nimmt Dienste der N -Schicht in Anspruch:

- Die N -Schicht erhält eine **Interface Data Unit (IDU)** von der $(N+1)$ -Schicht.

N -IDU enthält aus Sicht der N -Schicht

- Nutzdaten (**Service Data Unit (SDU)**) und
- Kontrollinformationen (**Interface Control Information (ICI)**), welche zum Erbringen des Dienstes notwendig sind (z. B. Adressinformationen).

Die N -Schicht

- erbringt auf der N -SDU die angeforderten Dienste,
- fügt sog. **Protocol Control Information (PCI)** für die N -Schicht der Gegenseite hinzu und
- erzeugt so aus PCI und SDU die **Protocol Data Unit (PDU)**.

Die N -Schicht nutzt den Dienst der $(N-1)$ -Schicht.

- Sie erzeugt eine $(N-1)$ -ICI, und
- übergibt diese zusammen mit der N -PDU als $(N-1)$ -IDU der nächst niedrigeren Schicht

Üblich ist der Begriff **Protocol Data Unit (PDU)**, welcher auf der N -Schicht

- ▶ die (ggf. bearbeiteten) Nutzdaten der $(N - 1)$ -Schicht sowie
 - ▶ Protokollsteuerungsinformationen (Protocol Control Information - PCI) der N -Schicht
- bezeichnet. Die PCI wird dabei häufig in Form eines **Headers** den Nutzdaten vorangestellt.

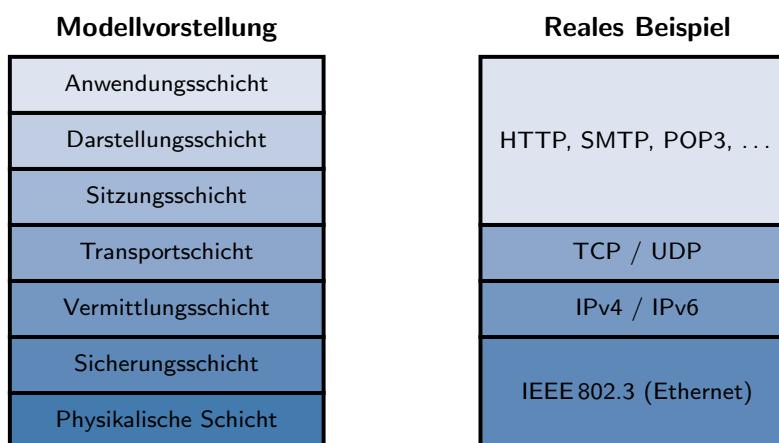
PDUs einiger Schichten haben eigene Bezeichnungen. Man spricht von

- ▶ **Segmenten** auf der Transportschicht,
- ▶ **Paketen** auf der Vermittlungsschicht bzw.
- ▶ **Rahmen** (engl. **Frames**) auf der Sicherungsschicht.

Diese Unterscheidungen ermöglichen es, implizit die gerade betrachtete Schicht anzugeben. Die Verwendung der Begriffe in der Literatur ist allerdings nicht immer einheitlich.

Schwächen des ISO/OSI-Modells

- ▶ Die Trennung der Schichten widerspricht manchmal anderen Interessen (z. B. der Effizienz)
- ▶ Einige Protokolle sind nicht klar einer bestimmten Schicht zuzuordnen, bzw. arbeiten sogar auf mehreren Schichten (**Cross Layer**)
- ▶ Die Zuordnung von Protokollen auf einzelne Schichten kann vom konkreten Einsatz der Protokolle abhängen



Eine kurze Übersicht zum ISO/OSI-Modell finden Sie u.a. in [3].

Bibliography I

- [1] C. Harrison. World City-to-City Connections.
<http://www.chrisharrison.net/index.php/Visualizations/InternetMap>.
- [2] T. O'Reilly. O'Reilly Network: What Is Web 2.0, Sept. 2005.
- [3] E. Stein. *Taschenbuch Rechnernetze und Internet*, chapter Das OSI-Modell, pages 22–28. Fachbuchverlag Leipzig, 2. edition, 2004. Auszug s. Moodle/SVN.

Grundlagen Rechnernetze und Verteilte Systeme

Kapitel 1 – Physikalische Schicht

Inhalt

Signale, Information und deren Bedeutung

Klassifizierung von Signalen

Zeit- und Frequenzbereich
Abtastung, Rekonstruktion und Quantisierung

Übertragungskanal

Kanaleinflüsse
Kanalkapazität

Nachrichtenübertragung

Übertragungsmedien

Signale, Information und deren Bedeutung

Definition (Signale, Symbole)

Signale sind zeitabhängige und messbare physikalische Größen. Definierten messbaren Signaländerungen lässt sich ein Symbol zuordnen. Diese Symbole repräsentieren Information.

Beispiele für Signale:

- ▶ Licht (z.B. Übermittlung von Morsezeichen in der Schifffahrt)
- ▶ Spannung (z.B. Telegraphie)
- ▶ Schall (z.B. gesprochene Sprache; Musik)

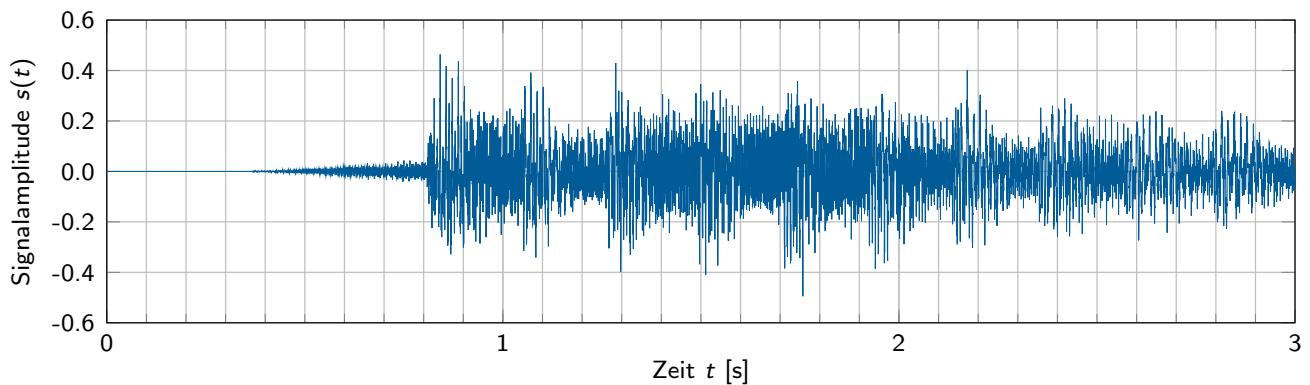


Abbildung: Die ersten 3 s von „Sunrise Avenue – Hollywood Hills“

Definition – Information

Information besteht in der **Unsicherheit**, Veränderungen eines Signals vorhersagen zu können. Der Informationsgehalt eines Zeichens $x \in \mathcal{X}$ hängt von der Wahrscheinlichkeit $p(x)$ ab, dass das informationstragende Signal zum Beobachtungszeitpunkt den diesem Zeichen zugeordneten Wert bzw. Wertebereich annimmt. Der Informationsgehalt I des Zeichens x mit der Auftrittswahrscheinlichkeit $p(x)$ ist definiert als

$$I(x) = -\log_2 p(x) \quad \text{mit} \quad [I] = \text{bit}.$$

Hinweis: Die Schreibweisen $p(x)$ oder p_x verwenden wir manchmal als Kurzform von $\Pr[X = x]$.

Definition – Entropie

Den mittleren Informationsgehalt einer Quelle bezeichnet man als **Entropie**

$$H = - \sum_{x \in \mathcal{X}} p(x) \log_2 (p(x)).$$

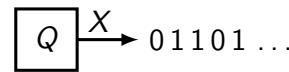
Beispiele:

- ▶ Deterministische (diskrete) Quelle, welche stets das Zeichen 'A' emittiert:



$$I(A) = -\log_2 (\Pr[X = A]) = -\log_2(1) = 0 \text{ bit}$$

- Binäre Quelle, welche auf nicht vorhersehbare Weise die Zeichen '0' oder '1' emittiert:



$$I(0) = -\log_2(\Pr[X = 0]) = -\log_2(0.5) = 1 \text{ bit}$$

$$I(1) = -\log_2(\Pr[X = 1]) = -\log_2(0.5) = 1 \text{ bit}$$

Die Entropie $H(X) = \sum_i p_i I(x_i)$ dieser Quelle beträgt

$$H(X) = -(p_0 \log_2(p_0) + p_1 \log_2(p_1)) = -(-0.5 - 0.5) = 1 \text{ bit/Zeichen.}$$

- Ungeordnete Zeichen eines langen deutschen Textes, d. h. $X \in \{A, B, C, \dots, Z\}$:



$$I(E) = -\log_2(\Pr[X = E]) = -\log_2(0.1740) \approx 2,5223 \text{ bit}$$

Die Entropie $H(X)$ dieser Quelle beträgt

$$H(X) = - \sum_{i=1}^N p_i \log_2(p_i) \approx 4,0629 \text{ bit/Zeichen.}$$

D. h. deutscher Text lässt sich mit durchschnittlich etwas mehr als 4 bit pro Zeichen kodieren.

Hinweis: Dies gilt nur für **gedächtnislose** Quellen oder hinreichend lange Texte. Andernfalls müssen bedingte Wahrscheinlichkeiten berücksichtigt werden.

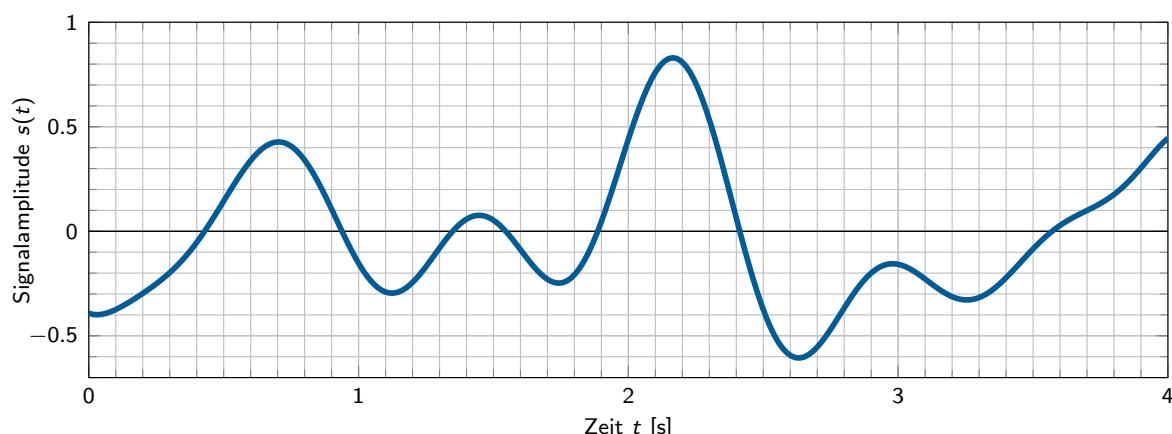
Welche Bedeutung hat ein bestimmtes Signal?

Ein Signal transportiert Information. Erst durch eine **Interpretationsvorschrift** erhält diese Information eine Bedeutung, d. h. es muss eine Abbildung zwischen **Symbolen** (Signalwerten bzw. Wertebereichen) und **Daten** geben.

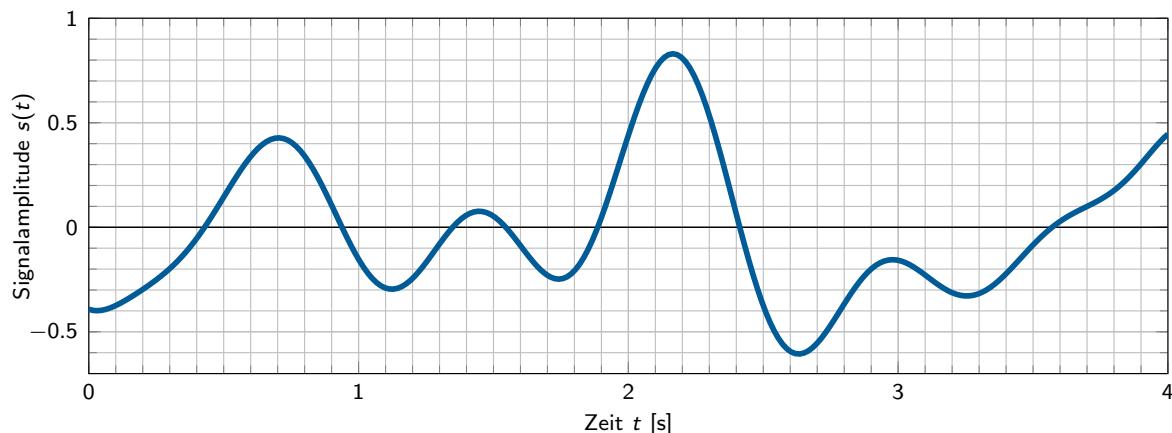
Beispiel: Gegeben sei ein binäres Alphabet mit den Zeichen $X \in \{0,1\}$. Die Interpretationsvorschrift laute

$$x = \begin{cases} 0 & s(t) \leq 0, \\ 1 & \text{sonst.} \end{cases}$$

Welche Bedeutung hat das unten abgebildete Signal?



Offene Fragen



- ▶ In welchen zeitlichen Abständen werden Samples genommen? ([Zeitdiskretisierung](#))
- ▶ Bedeutet häufigeres Abtasten auch automatisch mehr Information? ([Abtasttheorem](#))
- ▶ Wie werden kontinuierliche Signalwerte gerundet? ([Quantisierung](#))
- ▶ Welche Abbildungs- / Interpretationsvorschriften gibt es? ([Leitungskodierung](#))
- ▶ Welche Störfaktoren spielen eine Rolle? ([Rauschen, Dämpfung, Verzerrung, ...](#))
- ▶ Wie werden Fehler erkannt und ggf. korrigiert? ([Kanalkodierung](#))
- ▶ Und wie wird ein derartiges Signal überhaupt erzeugt? ([Impulsformung, Modulation](#))

Inhalt

Signale, Information und deren Bedeutung

Klassifizierung von Signalen

Zeit- und Frequenzbereich
Abtastung, Rekonstruktion und Quantisierung

Übertragungskanal

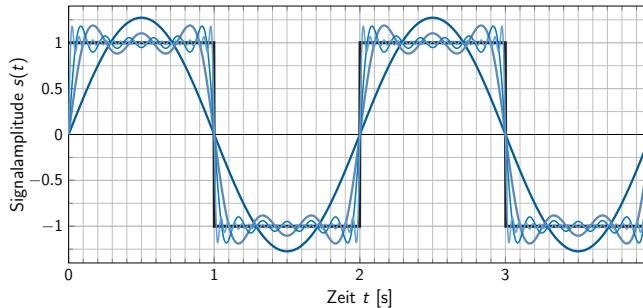
Kanaleinflüsse
Kanalkapazität

Nachrichtenübertragung

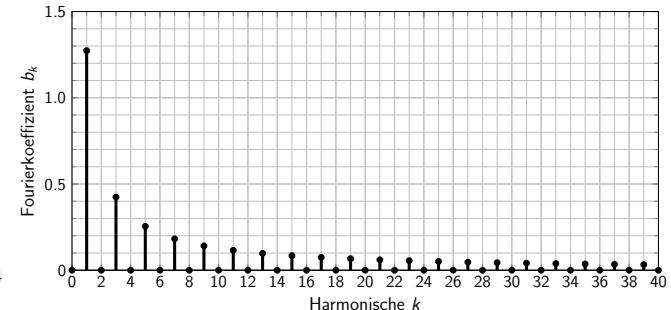
Übertragungsmedien

Zeit- und Frequenzbereich

(Periodische) Zeitsignale lassen sich als Überlagerung von Sinus- und Kosinusschwingungen unterschiedlicher Frequenzen auffassen:



(a) Zeitsignal $s(t)$



(b) Spektrum $S(f)$

Fourier-Reihe: (mit $\omega = 2\pi/T$, Periodendauer hier $T = 2$ s)

Im Grenzwert $N \rightarrow \infty$ gilt:

$$s(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos(k\omega t) + b_k \sin(k\omega t))$$

Fourierreihe

Ein **periodisches** Signal $s(t)$ lässt sich als Summe gewichteter Sinus- und Kosinus-Schwingungen darstellen. Die so entstehende Reihenentwicklung von $s(t)$ bezeichnet man als **Fourierreihe**:

$$s(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos(k\omega t) + b_k \sin(k\omega t)). \quad (1)$$

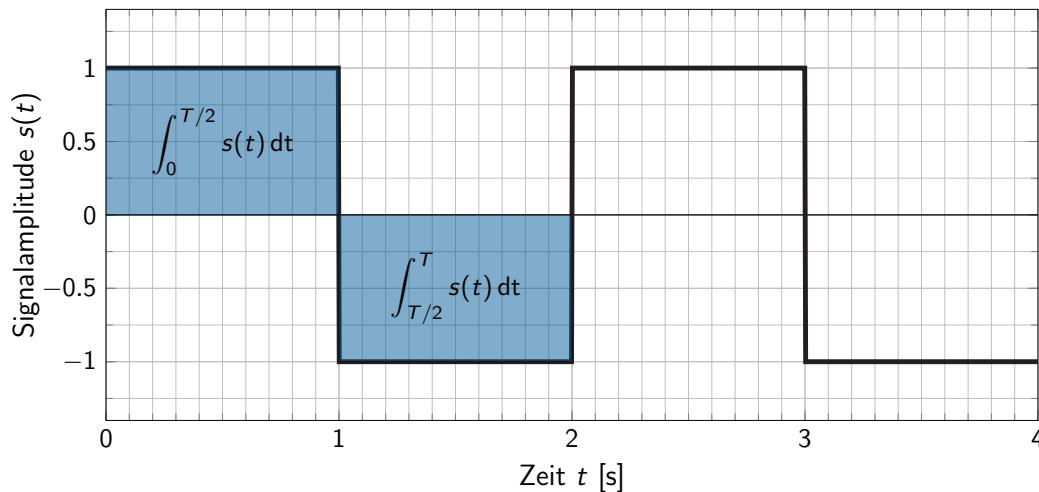
Das k -te Summenglied bezeichnet man auch als k -te **Harmonische**. Das konstante Glied $a_0/2$ repräsentiert eine Verschiebung der Signalamplitude bezüglich der Ordinate (y-Achse) und damit den konstanten Anteil der Funktion. Die **Kreisfrequenz** $\omega = 2\pi/T$ stellt lediglich eine Normierung bezüglich der Periodendauer T des Signals dar.

Die Koeffizienten (Gewichte) a_k und b_k lassen sich wie folgt bestimmen:

$$a_k = \frac{2}{T} \int_0^T s(t) \cos(k\omega t) dt \quad \text{und} \quad b_k = \frac{2}{T} \int_0^T s(t) \sin(k\omega t) dt. \quad (2)$$

Einfache Signaleigenschaften „by inspection“

- ▶ Die Bestimmung der Koeffizienten a_k und b_k ist nur Rechenarbeit
- ▶ Einige Eigenschaften sieht man einfachen Signalen aber leicht an, z. B.:

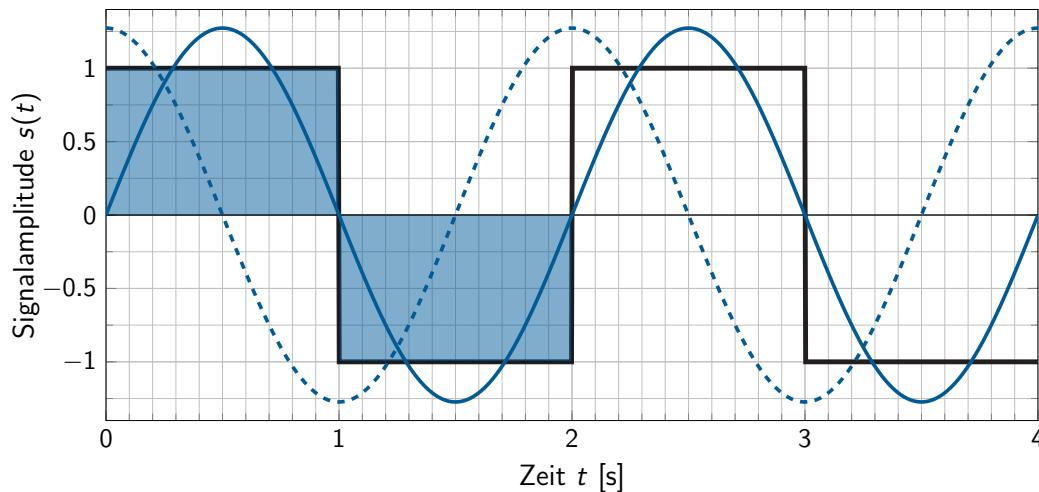


- ▶ Punktsymmetrie zu $(T/2, 0)$ $\Rightarrow a_0 = \frac{2}{T} \int_0^T s(t) dt = 0$
- ▶ Kein Gleichanteil (d.h. keine Verschiebung entlang der Ordinate)

Frage: Was ist mit dem Signal $s'(t) = s(t) + c$ mit $c > 0$?

Einfache Signaleigenschaften „by inspection“

- ▶ Die Bestimmung der Koeffizienten a_k und b_k ist nur Rechenarbeit
- ▶ Einige Eigenschaften sieht man einfachen Signalen aber leicht an, z. B.:



- ▶ Alle Gewichte für den Kosinus sind null, also $a_k = 0 \forall k \in \mathbb{N}$
- ▶ Grund: $s(t)$ ist genau in Phase mit dem Sinus

Frage: Was ist, wenn man die Phase von $s(t)$ um 90° verschiebt?

Bislang haben wir nur periodische Signale betrachtet. Was ist mit **nicht-periodischen** Signalen?

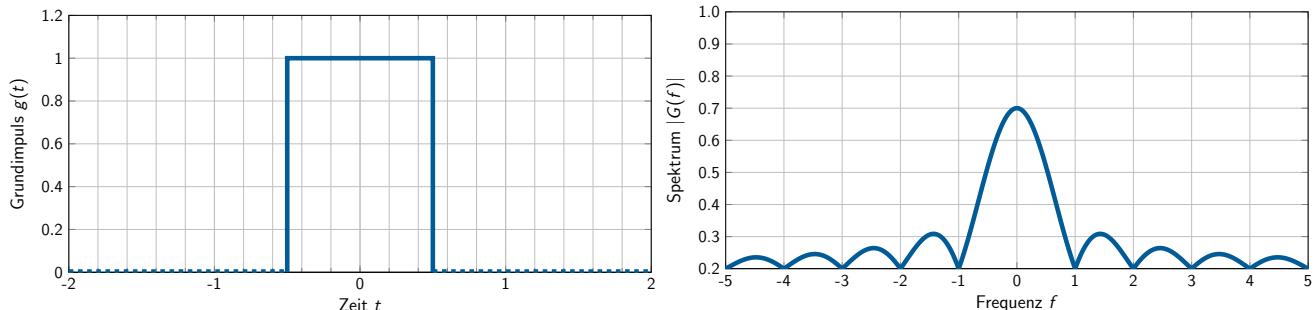
- ▶ Keine Entwicklung als Fourierreihe möglich
- ▶ Kontinuierliches (anstatt diskretes) Spektrum
- ▶ Frequenzbereich erhält man mittels **Fouriertransformation**

Fouriertransformation

Die Fourier-Transformierte einer stetigen, integrierbaren Funktion $s(t)$ ist gegeben als

$$s(t) \xrightarrow{\text{Fourier}} S(f) = \frac{1}{\sqrt{2\pi}} \int_{t=-\infty}^{\infty} s(t) e^{-i2\pi ft} dt.$$

Beispiel: Rechteckimpuls und zugehöriges Spektrum



$$\text{rect}(t) \xrightarrow{\text{Fourier}} \frac{1}{\sqrt{2\pi T}} \text{sinc}\left(\frac{f}{T}\right) \quad \text{mit } \text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

Abtastung, Rekonstruktion und Quantisierung [2]

Natürlich vorkommende Signale sind **zeitkontinuierlich** und **wertkontinuierlich**, d. h. sie nehmen zu unendlich vielen Zeitpunkten beliebige reelle Werte an.

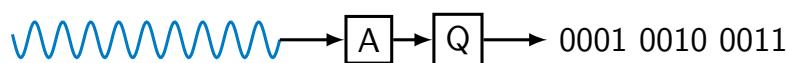
Problem für Computer:

- ▶ Endlicher Speicher
- ▶ Endliche Rechengenauigkeit

Lösung: **Diskretisierung** von Signalen im

- ▶ Zeitbereich (**Abtastung**) und
- ▶ Wertbereich (**Quantisierung**).

Ein zeit- und wertdiskretes Signal ist **digital** und wird in **Wörtern** fester Länge gespeichert.



Vergleiche: Nutzung von Fix- bzw. Gleitkommazahlen anstelle von reellen Zahlen entspricht einer Rundung (Quantisierung) auf eine endliche Anzahl diskreter Stufen.

Abtastung

Das Signal $s(t)$ wird mittels des Einheitsimpulses (Dirac-Impulses) $\delta[t]$ in äquidistanten Abständen T_a (Abtastintervall) für $n \in \mathbb{Z}$ abgetastet:

$$\hat{s}(t) = s(t) \sum_{n=-\infty}^{\infty} \delta[t - nT_a], \text{ mit } \delta[t - nT_a] = \begin{cases} 1 & t = nT_a, \\ 0 & \text{sonst.} \end{cases}$$

Da $\hat{s}(t)$ nur zu den Zeitpunkten nT_a für ganzzahlige n von Null verschieden ist, vereinbaren wir die Schreibweise $\hat{s}[n]$ für zeitdiskrete aber wertkontinuierliche Signale.

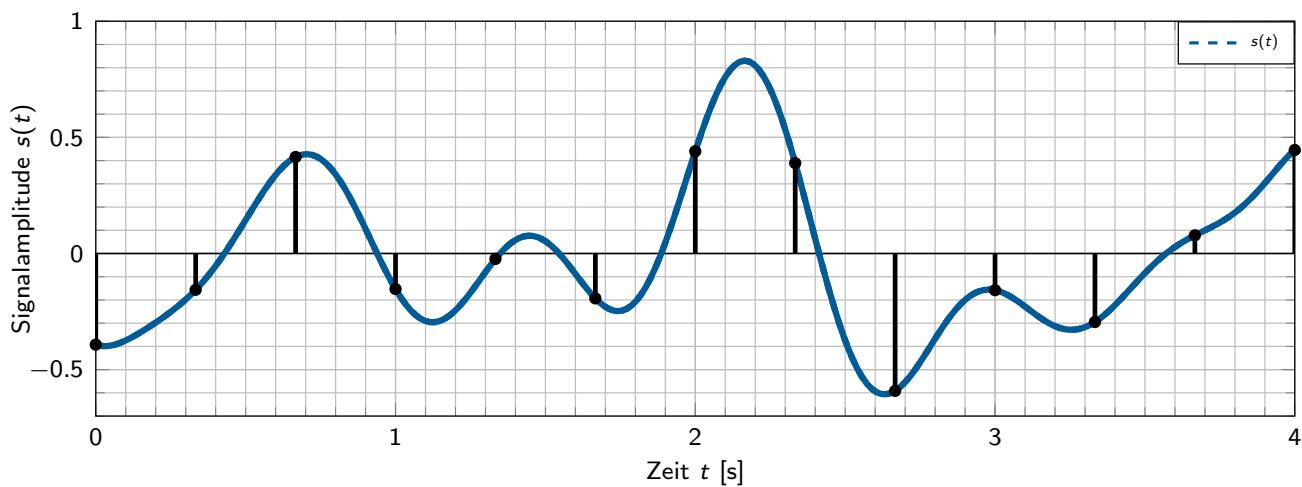


Abbildung: Zeitkontinuierliches Signal $s(t)$ und Abtastwerte $\hat{s}[n]$

Rekonstruktion

Mittels der Abtastwerte $\hat{s}[n]$ ist es möglich, das ursprüngliche Signal $s(t)$ zu rekonstruieren:

$$s(t) \approx \sum_{n=-\infty}^{\infty} \hat{s}[n] \cdot \text{sinc}\left(\frac{t - nT_a}{T_a}\right).$$

- ▶ Abtastwerte sind **Stützstellen** und
- ▶ dienen als Gewichte für eine passende **Ansatzfunktion** (trigonometrische Interpolation, vgl. Polynominterpolation → Numerisches Programmieren).

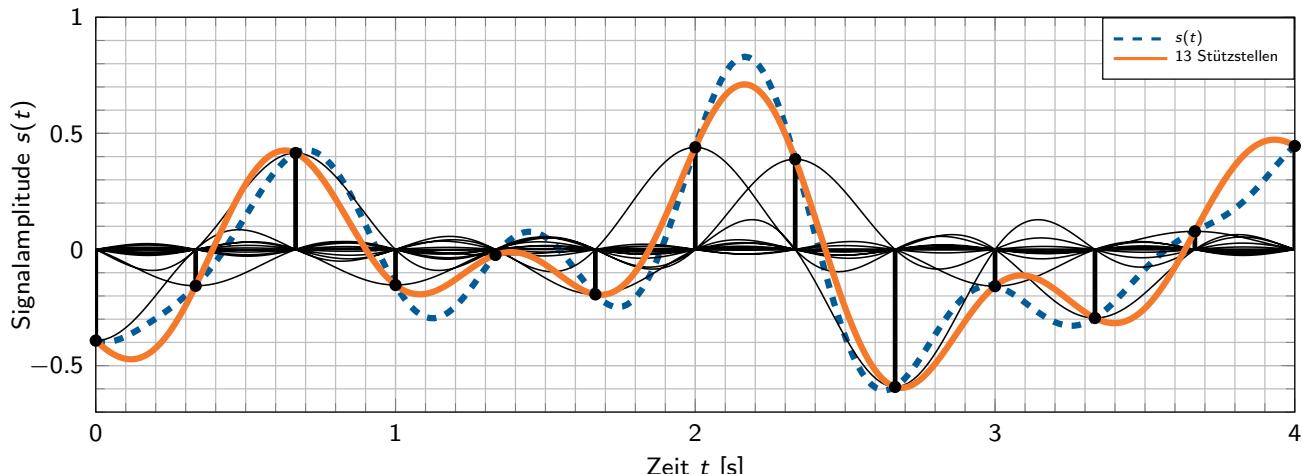


Abbildung: Die Summe der gewichteten Ansatzfunktionen nähert sich dem ursprünglichen Signal in Abhängigkeit der Anzahl der Summenglieder.

Wann ist eine verlustfreie Rekonstruktion möglich?

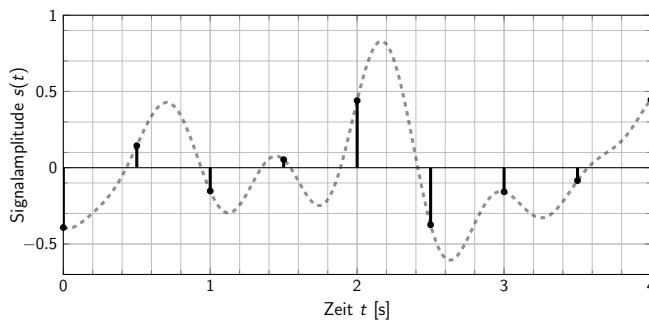
- Die Multiplikation im Zeitbereich entspricht einer Faltung im Frequenzbereich:

$$s(t) \cdot \delta[t - nT] \circledast \frac{1}{T} S(f) * \delta[f - n/T].$$

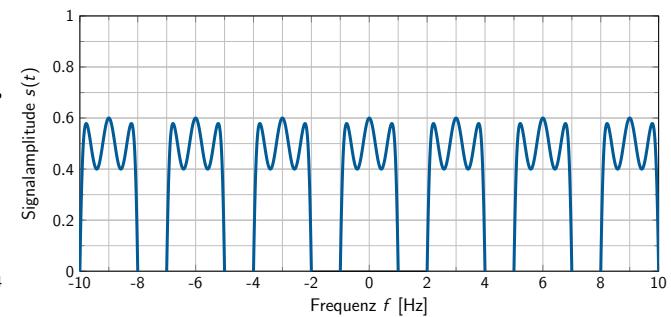
- Diese Faltung mit Einheitsimpulsen entspricht einer Verschiebung von $S(f)$ entlang der Abszisse.

Folglich entspricht die Abtastung des Signals $s(t)$ in Abständen T_a der periodischen Wiederholung seines Spektrums $S(f)$ in Abständen von $f_a = 1/T_a$.

Beispiel: Abtastung eines auf B bandbegrenzten Signals $s(t)$ mit der Abtastfrequenz $f_a = 3B$



(a) Abgetastetes Signal $\hat{s}[n]$



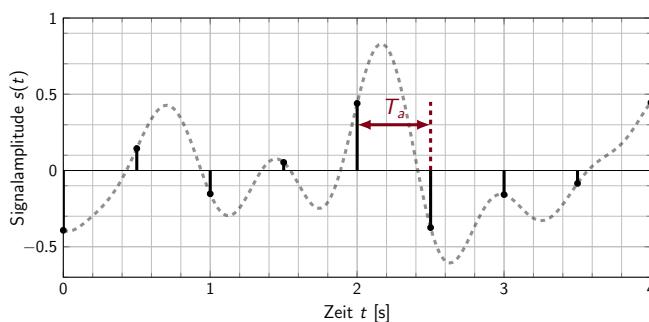
(b) Zugehöriges Spektrum $\hat{S}(f)$ (schematisch)

Abtasttheorem von Shannon und Nyquist

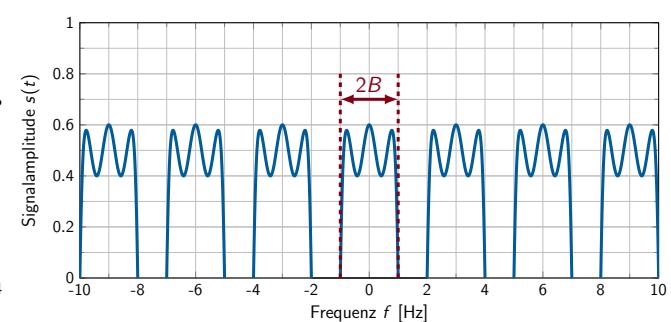
Ein auf $|f| \leq B$ bandbegrenztes Signal $s(t)$ ist vollständig durch äquidistante Abtastwerte $\hat{s}[n]$ beschrieben, sofern diese nicht weiter als $T_a = 1/2B$ auseinander liegen. Die Abtastfrequenz, welche eine vollständige Signalrekonstruktion erlaubt, ist folglich durch

$$f_a > 2B$$

nach unten beschränkt.



(a) Abgetastetes Signal $\hat{s}[n]$



(b) Zugehöriges Spektrum $\hat{S}(f)$ (schematisch)

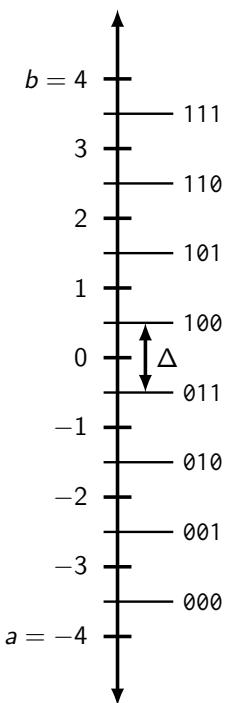
- Wählt man $f_a < 2B$, so überlappen sich die periodischen Wiederholungen des Spektrums
- Diesen Effekt bezeichnet man als **Aliasing**
- Eine **verlustfreie** Rekonstruktion ist in diesem Fall **nicht** möglich

Quantisierung

- ▶ Bei N bit Wortbreite sind $M = 2^N$ diskrete Signalstufen unterscheidbar
- ▶ Diese werden in einem Quantisierungsintervall $I_Q = [a, b]$ „sinnvoll“ verteilt
- ▶ Was ist „sinnvoll“?

Beispiel: Lineare Quantisierung mit mathematischem Runden

- ▶ Optimal, wenn alle Werte innerhalb I_Q mit gleicher Wahrscheinlichkeit auftreten
- ▶ Stufenbreite $\Delta = \frac{b - a}{M}$
- ▶ Innerhalb I_Q beträgt der maximale Quantisierungsfehler $q_{\max} = \Delta/2$
- ▶ Signalwerte außerhalb I_Q werden auf obere bzw. untere Grenze von I_Q normiert \Rightarrow Außerhalb I_Q ist der Quantisierungsfehler unbeschränkt

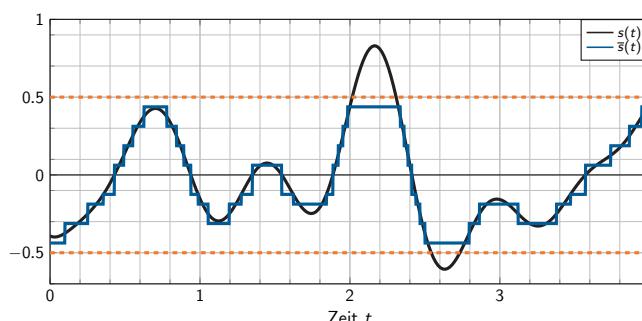


Was ist, wenn die Signalwerte nicht gleichverteilt sind?

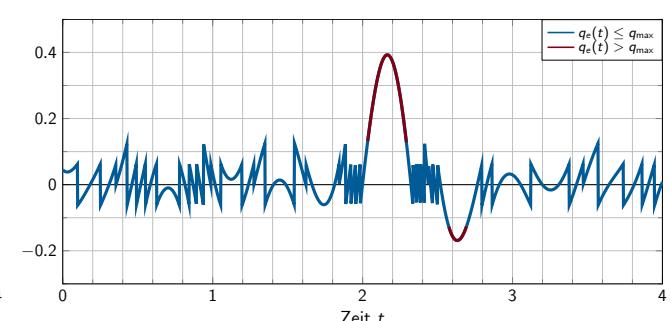
- ▶ Lineare Quantisierung ist typischerweise suboptimal
- ▶ Nicht-lineare Quantisierung wird beispielsweise bei der Digitalisierung von Sprache oder Musik eingesetzt.

Quantisierung (Beispiel)

Lineare Quantisierung im Intervall $I = [-0,5; 0,5]$ mit $N = 3$ bit:



(c) Quantisiertes Signal $\bar{s}(t)$ (blau)



(d) Quantisierungsfehler $q_e(t) = s(t) - \bar{s}(t)$

Codewort	000	001	010	011	100	101	110	111
Signalstufe	$-7/16$	$-5/16$	$-3/16$	$-1/16$	$1/16$	$3/16$	$5/16$	$7/16$

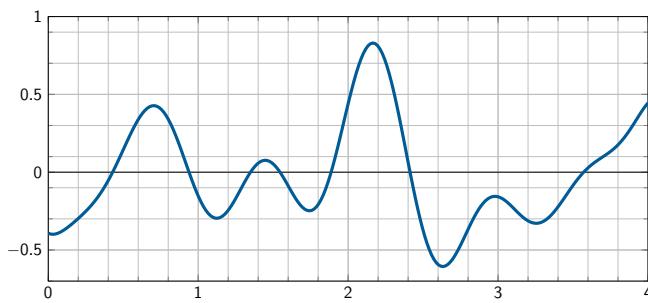
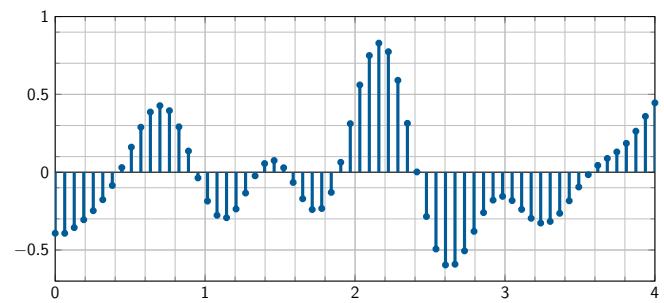
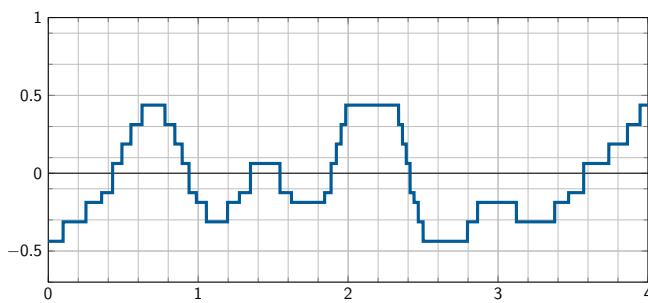
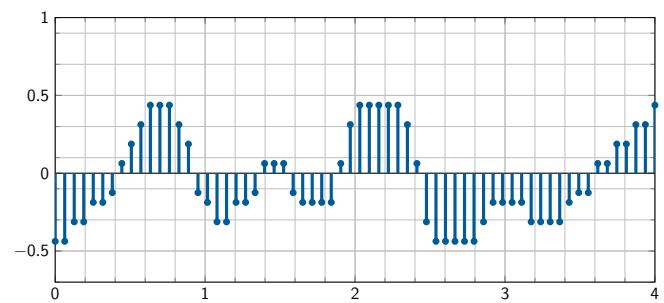
Frage:

Warum liegt die höchste Signalstufe bei $7/16$ und nicht bei $1/2$?

Anmerkungen:

- ▶ Die Zuweisung von Codewörtern zu Signalstufen ist im Prinzip willkürlich
- ▶ Häufig wählt man jedoch einen Code, welcher die Auswirkung einzelner Bitfehler reduziert (z. B. Gray-Code: Benachbarte Codewörter unterscheiden sich nur in jeweils einer binären Ziffer, d.h. die Hamming-Distanz ist 1.)

Übersicht: Signaltypen

(e) Analog $s(t)$ (f) Zeitdiskret und wertkontinuierlich $\hat{s}[n]$ (g) Zeitkontinuierlich und wertdiskret $\bar{s}(t)$ (h) Digital $s[n]$

Inhalt

Signale, Information und deren Bedeutung

Klassifizierung von Signalen

Zeit- und Frequenzbereich
Abtastung, Rekonstruktion und Quantisierung

Übertragungskanal

Kanaleinflüsse
Kanalkapazität

Nachrichtenübertragung

Übertragungsmedien

Aus dem letzten Teilkapitel sollten wir wissen:

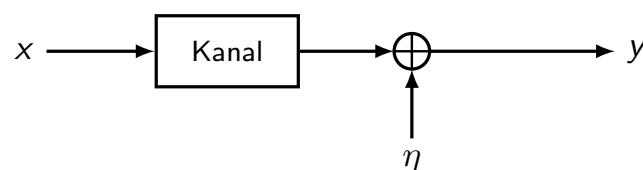
- ▶ Was sind die Unterschiede zwischen analogen, zeitdiskreten, wertdiskreten und digitalen Signalen?
- ▶ Wie muss ein Signal abgetastet werden, so dass keine Information verloren geht?
- ▶ Unter welchen Bedingungen kann ein natürlich vorkommendes Signal aus abgetasteten und quantisierten Werten verlustfrei rekonstruiert werden?
- ▶ Wie sollten die Abtastwerte quantisiert werden, wenn innerhalb des Quantisierungsintervalls jeder Signalpegel gleich wahrscheinlich ist?

In diesem Abschnitt klären wir die folgenden Fragen:

- ▶ Welchen Einfluss hat der Übertragungskanal auf ein Signal?
- ▶ Wie hoch ist die theoretisch maximal erzielbare Übertragungsrate?

Kanaleinflüsse

Modellvorstellung eines (linearen, zeitinvarianten) Kanals mit einem Ein- und Ausgang:

**Unser Modell berücksichtigt:**

- ▶ **Dämpfung** (Signalamplitude des Nutzsignals am Ausgang ist geringer als am Eingang)
- ▶ **Tiefpassfilterung** (höhere Frequenzen werden stärker gedämpft als niedrige)
- ▶ **Verzögerung** (die Übertragung benötigt eine gewisse Zeit)
- ▶ **Rauschen** in Form von **Additive White Gaussian Noise (AWGN)**¹

Wir berücksichtigen unter anderem nicht (weil zu kompliziert):

- ▶ **Interferenzen** durch andere Übertragungen
- ▶ **Reflektionen** eigener Signale
- ▶ **Zeitvariante** Einflüsse, z. B. können Objekte und Personen eine Funkübertragung beeinflussen

¹ AWGN ist eine vereinfachende Modellvorstellung von Rauschprozessen. In der Realität gibt es kein AWGN.

Beispiel:

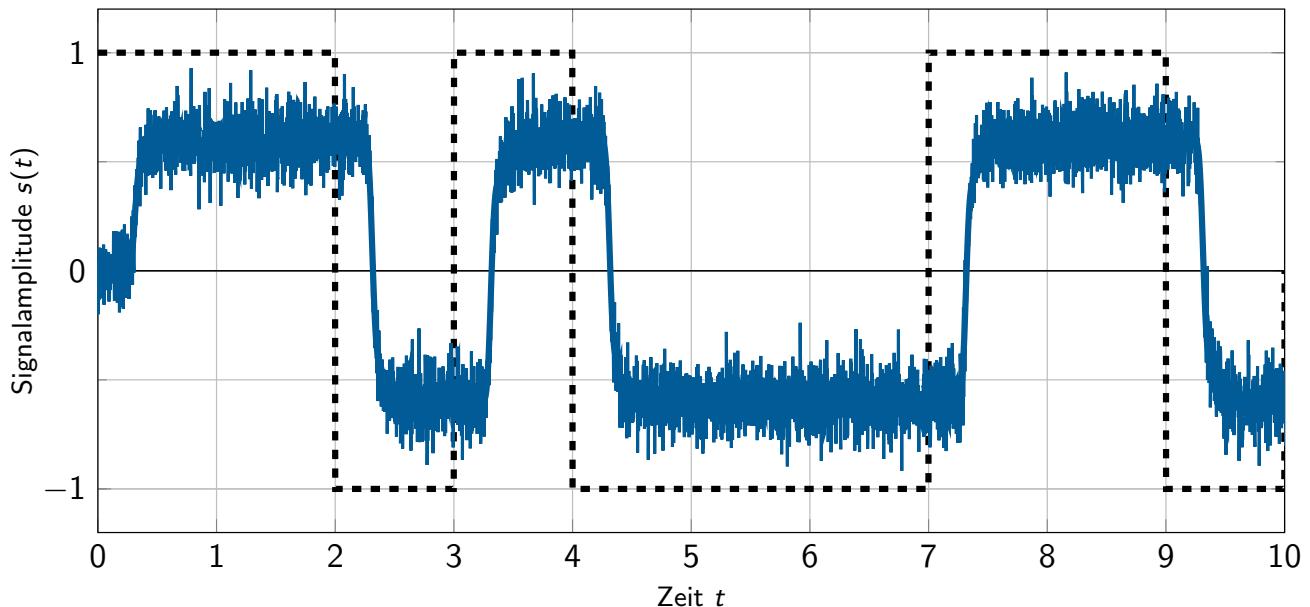


Abbildung: Sendesignal nach Dämpfung und Tiefpasseinflüssen durch den Kanal sowie mit AWGN

Kanalkapazität

Wir haben bereits gesehen, dass

- ▶ ein Kanal wie ein Tiefpass wirkt und
- ▶ zusätzliches Rauschen die Übertragung stört.

Wegen der für Kanäle typischen Tiefpasscharakteristik kann man von einer **Kanalbandbreite B** sprechen:

- ▶ Niedrige Frequenzen passieren ungehindert (Tiefpass)
- ▶ Hohe Frequenzen werden gedämpft
- ▶ Ab einer bestimmten Frequenz ist die Dämpfung so stark, dass die betreffenden Signalanteile vernachlässigt werden können

Vereinfacht nehmen wir eine scharfe Grenze für B an:

- ▶ Frequenzanteile $|f| < B$ passieren
- ▶ Frequenzanteile $|f| \geq B$ werden gesperrt

Wie hoch ist die erzielbare Datenrate auf einem Kanal mit Bandbreite B ?

Hierfür benötigen wir einen Zusammenhang zwischen

- ▶ der Kanalbandbreite B ,
- ▶ der Anzahl M unterscheidbarer Signalstufen und
- ▶ dem Verhältnis zwischen der Leistung des Nutzsignals und des Rauschens.

Rauschfreier, binärer Kanal

Wir erinnern uns an das Abtasttheorem:

- ▶ Ein auf B bandbegrenztes Signal muss mind. mit der Frequenz $2B$ abgetastet werden,

so dass das Signal verlustfrei rekonstruiert werden kann, d. h. dass keine Information verloren geht.

Anders herum betrachtet: Man erhält aus einem Signal

- ▶ bis zu $2B$ unterscheidbare² und voneinander unabhängige Werte.

Tastet man häufiger ab, gewinnt man keine neue Information. Dies führt zu einer neuen Interpretation der Frequenz $f = 2B$, welche auch als **Nyquist-Rate** bezeichnet wird.

Definition: Nyquist-Rate

Sei B die Grenzfrequenz eines bandbegrenzten Kanals. Dann ist die Nyquist-Rate $f_N = 2B$

- ▶ eine untere Schranke für die minimale Abtastrate, die eine vollständige Rekonstruktion des Signals erlaubt,
- ▶ eine obere Schranke für die Anzahl an Werten je Zeiteinheit, die nach der Übertragung über den Kanal unterscheidbar und unabhängig voneinander sind.

² Hinreichend empfindliche Messsysteme vorausgesetzt

Rauschfreier, M -ärer Kanal

Angenommen es können nicht nur zwei sondern $M = 2^N$ unterscheidbare Symbole übertragen werden. Wie ändert sich die erzielbare Datenrate?

Wir erinnern uns an Quantisierung und Entropie:

- ▶ Mit einer Wortbreite von N bit lassen sich $M = 2^N$ diskrete Signalstufen darstellen.
- ▶ Emittiert eine Quelle alle Zeichen (Signalstufen) mit der gleichen Wahrscheinlichkeit, so ist die Entropie (und damit die mittlere Information) der Quelle maximal.

Folglich erhalten wir für die Übertragungsrate

- ▶ über einen Kanal der Bandbreite B
- ▶ die maximale Rate $R_{\max} = 2B \log_2(M)$ bit.

Hartleys Gesetz

Auf einem Kanal der Bandbreite B mit M unterscheidbaren Signalstufen ist die Kanalkapazität durch

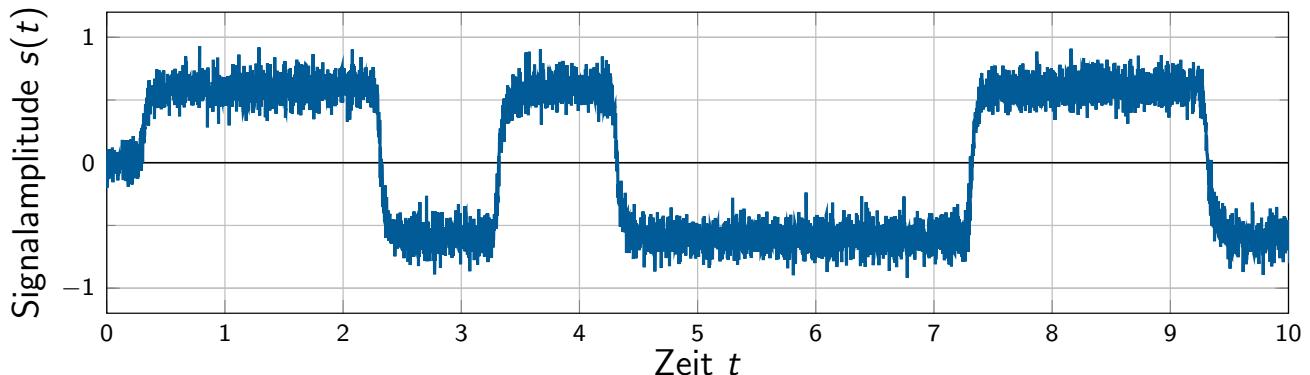
$$C_H = 2B \log_2(M) \text{ bit}$$

nach oben begrenzt.

Interessant: Wenn wir beliebig viele Signalstufen voneinander unterscheiden könnten, wäre die erzielbare Datenrate unbegrenzt! Wo ist das Problem?

Rauschen

- ▶ Rauschen macht es schwer, Signalstufen auseinanderzuhalten
- ▶ Je feiner die Signalstufen gewählt werden, desto schwieriger wird dies



Maß für die Stärke des Rauschens:

$$\text{SNR} = \frac{\text{Signalleistung}}{\text{Rauschleistung}} = \frac{P_S}{P_N}$$

Das **Signal to Noise Ratio (SNR)** wird in der Einheit dB angegeben:

$$\text{SNR dB} = 10 \cdot \log_{10}(\text{SNR})$$

Beispiel: $P_S = 1 \text{ mW}$, $P_N = 0.5 \text{ mW}$

$$\text{SNR} = 10 \cdot \log_{10} \left(\frac{1}{0.5} \right) \text{ dB} \approx 3.0 \text{ dB}$$

Signalleistung

Definition (Leistung eines Signals)

Der Erwartungswert des Quadrats der Signalamplitude entspricht der **Signalleistung**. Die Varianz (Streuung) der Signalamplitude entspricht der Signalleistung ohne deren Gleichanteil und stellt die informationstragende Leistung eines Signals dar.

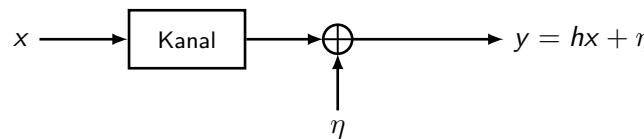
Die Signalleistung eines erwartungswertfreien³ Signals $y(t)$ beträgt

$$\begin{aligned} \text{Var}[y] &= \int_{-\infty}^{\infty} y^2 p(y) \, dy \\ &= \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} y^2(t) \, dy = P_y. \end{aligned}$$

³ Als „erwartungsfrei“ bezeichnet man im üblichen Sprachgebrauch eine Zufallsvariable, deren Erwartungswert null ist. Dies kann zu Missverständnissen führen, da es auch Zufallsvariablen gibt, deren Erwartungswert nicht existiert.

Rauschbehafteter, M -ärer Kanal

- ▶ Keine Quantisierungsfehler (da wir nur an der Rausch-Abhängigkeit interessiert sind)
- ▶ Das (erwartungswertfreie) Sendesignal x werde durch den Kanal um den Faktor $h < 1$ gedämpft und von (erwartungswertfreiem) **unabhängigem** additivem Rauschen η überlagert



Für die Signalleistung⁴ P_y am Kanalausgang erhalten wir

$$P_y = \text{Var}[y] = \text{Var}[hx + \eta] = h^2 \text{Var}[x] + \text{Var}[\eta].$$

Für den Spezialfall, dass $\eta \sim \mathcal{N}(0, \sigma_\eta^2)$ (**Gaussian Channel**), lässt sich zeigen, dass ein ebenfalls normalverteiltes $x \sim \mathcal{N}(0, \sigma_x^2)$ die Datenrate maximiert. Wir erhalten in diesem Fall

$$P_y = h^2 \sigma_x^2 + \sigma_\eta^2.$$

Setzt man nun P_y ins Verhältnis zur Rauschleistung $P_N = \sigma_\eta^2$, so erhalten wir

$$\frac{h^2 \sigma_x^2 + \sigma_\eta^2}{\sigma_\eta^2} = 1 + \frac{h^2 \sigma_x^2}{\sigma_\eta^2} = 1 + \frac{P_S}{P_N} = 1 + \text{SNR}.$$

P_S ist die **Leistung des Nutzsignals**, die beim Empfänger ankommt.

⁴ erwartungswertfreie ergodische Signale: $\text{Var}[y] = \int_{-\infty}^{\infty} y^2 p(y) dy = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} y^2(t) dt = P_y$.

Shannon-Hartley-Theorem

Auf einem Kanal der Bandbreite B mit additiven weißen Rauschen mit Rauschleistung P_N und Signalleistung P_S beträgt die obere Schranke für die erreichbare Datenrate

$$C_S = B \log_2 \left(1 + \frac{P_S}{P_N} \right) \text{ bit.}$$

Herleitung des Theorems: siehe Shannons Veröffentlichung **Communication in the Presence of Noise** von 1949. **Vergleich mit Hartleys Gesetz (aufschlussreich!):**

$$C_H = 2B \log_2(M) = 2B \log_2 \left(\frac{b-a}{\Delta} \right).$$

- ▶ Die Intervallgrenzen a, b beziehen sich hier auf das unquantisierte Signal
- ▶ Mit $\alpha = a + \Delta/2$ und $\beta = b - \Delta/2$ als minimale bzw. maximale quantisierte Signalamplitude erhalten wir

$$C_H = 2B \log_2 \left(\frac{\beta - \alpha + \Delta}{\Delta} \right) = B \log_2 \left(\left(1 + \frac{\beta - \alpha}{\Delta} \right)^2 \right). \quad (3)$$

Wird (3) ausmultipliziert, kommt aber etwas anderes raus!

- ▶ C_S berücksichtigt **nur additives Rauschen des Kanals, aber keine Quantisierungsfehler**.
- ▶ C_H berücksichtigt **nur die Signalstufen und damit die Quantisierung („Quantisierungsrauschen“), aber keine Kanaleinflüsse**.
- ▶ Der fehlende gemischte Term, wenn man (3) ausmultipliziert und mit C_S vergleicht, liegt in der **Unabhängigkeitsannahme** des Nutzsignals und des Rauschens begründet ($E[x\eta] = E[x]E[\eta]$). Der Quantisierungsfehler ist natürlich nicht unabhängig vom Eingangssignal – aus diesem Grund lässt sich (3) nicht ohne Näherung in die selbe Form wie C_S bringen.

Zusammenfassung

Die Kanalkapazität C ist durch zwei Faktoren beschränkt:

► **Die Anzahl M der unterscheidbaren Symbole**

Selbst ein rauschfreier Kanal hilft nichts, wenn wir nur zwei Symbole nutzen (können).

► **Signal-to-Noise Ratio (SNR)**

Ist das Signal-zu-Rausch-Verhältnis SNR zu gering, muss ggf. der Abstand Δ zwischen den Signalstufen erhöht und damit die Anzahl unterscheidbarer Symbole verringert werden, um eine zuverlässige Unterscheidung gewährleisten zu können.

Für die tatsächliche Kanalkapazität C gilt also folgende obere Schranke:

$$C < \min\{C_H, C_S\} = \min\{2B \log_2(M), B \log_2(1 + \text{SNR})\} \text{ bit.}$$

Anmerkungen:

- Das ist nur ein Modell – mit stark vereinfachenden Annahmen.
 - Wie man einen Kanalcode mit genau der richtigen Menge Redundanz konstruieren kann, so dass C maximiert wird, ist ein offenes Problem der Informationstheorie. (← Challenge!)
 - Wir sprechen hier von Datenraten im informationstheoretischen Sinn, d. h. die zu übertragenden Daten liegen redundanzfrei vor. Dies ist in praktischen Systemen nie gewährleistet:
 - Nutzdaten werden vor dem Senden nicht zwangsläufig (und niemals optimal) komprimiert
 - Zusätzlich zu den Nutzdaten werden **Kontrollinformationen (Header)** benötigt (→ später)
- ⇒ Die tatsächlich erzielbare Netto-Datenrate liegt unterhalb der informationstheoretischen Schranke.

Inhalt

Signale, Information und deren Bedeutung

Klassifizierung von Signalen

Zeit- und Frequenzbereich
Abtastung, Rekonstruktion und Quantisierung

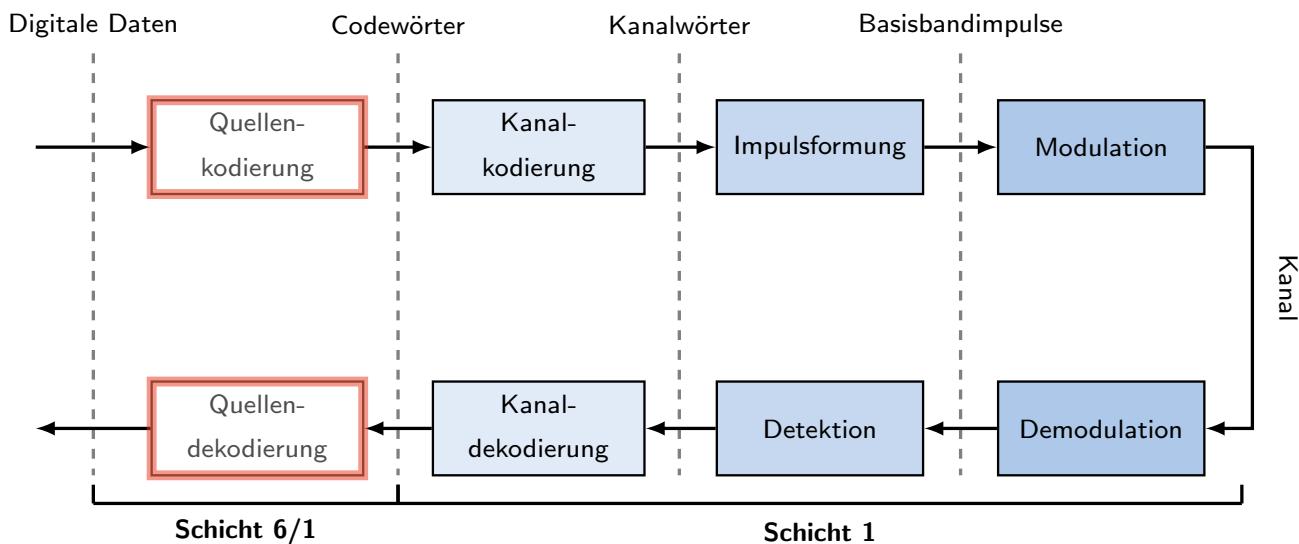
Übertragungskanal

Kanaleinflüsse
Kanalkapazität

Nachrichtenübertragung

Übertragungsmedien

Nachrichtenübertragung



Quellenkodierung [1]

Quellenkodierung (Source Coding)

Ziel der Quellenkodierung ist es,

- ▶ Redundanz aus den zu übertragenden Daten zu entfernen durch Abbildung von Bitsequenzen auf Codewörter, was einer verlustlosen Datenkompression entspricht.

Die Quellenkodierung kann in unterschiedlichen Schichten des ISO/OSI-Modell vorkommen:

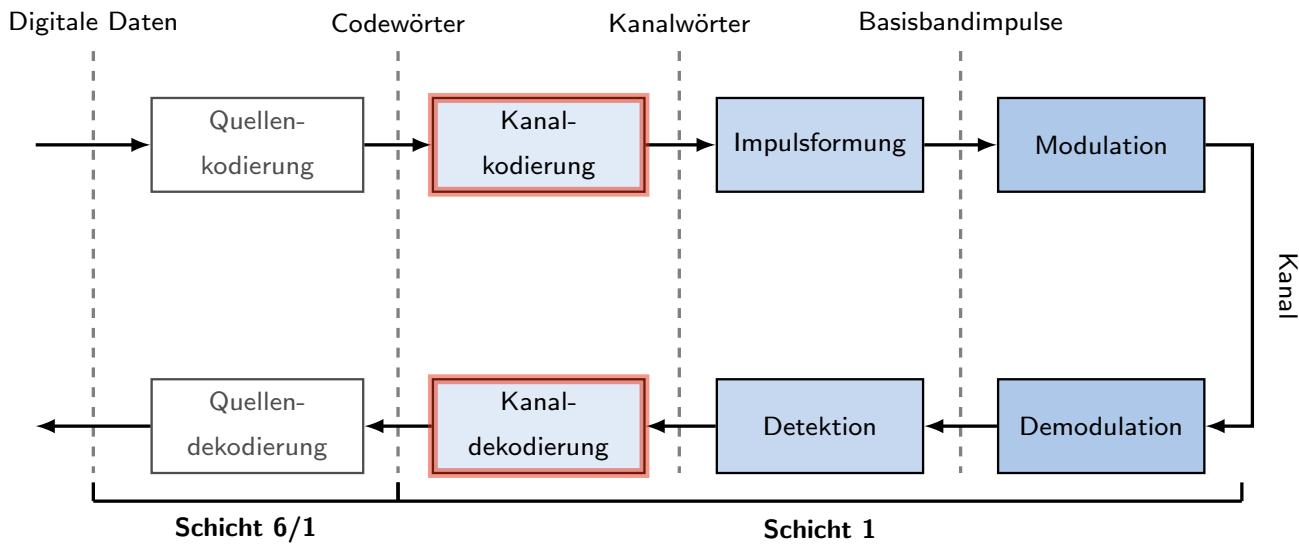
- ▶ Datenkompression kann auf der Darstellungsschicht (Schicht 6) stattfinden
- ▶ Daten können bereits in komprimierter Form vorliegen (verlustlos komprimierte Dateiformate, z. B. ZIP, PNG, FLAC)
- ▶ Im Mobilfunkbereich (digitale Sprachübertragung) kann die Quellenkodierung einer niedrigen Schicht zugeordnet werden
- ▶ In lokalen Netzwerken (Ethernet, WLAN) findet i. d. R. keine Quellenkodierung statt

Beispiele:

- ▶ Huffman-Code
- ▶ Run-Length-Encoding (RLE)

In Kapitel 5 (Darstellungsschicht) gehen wir auf den Huffman-Code ein

Übersicht



Kanalkodierung [1]

Jeder realisierbare Übertragungskanal ist unzuverlässig. Ein Maßstab dafür ist die Bitfehlerwahrscheinlichkeit p_e :

- ▶ Bei Ethernet über Kupferkabel charakteristisch: $p_e \approx 10^{-8}$
- ▶ Bei WLAN charakteristisch: $p_e \approx 10^{-6}$ oder mehr
- ▶ Bei ungesicherter Funkübertragung charakteristisch: $p_e \approx 10^{-4}$ oder mehr

Gedankenspiel:

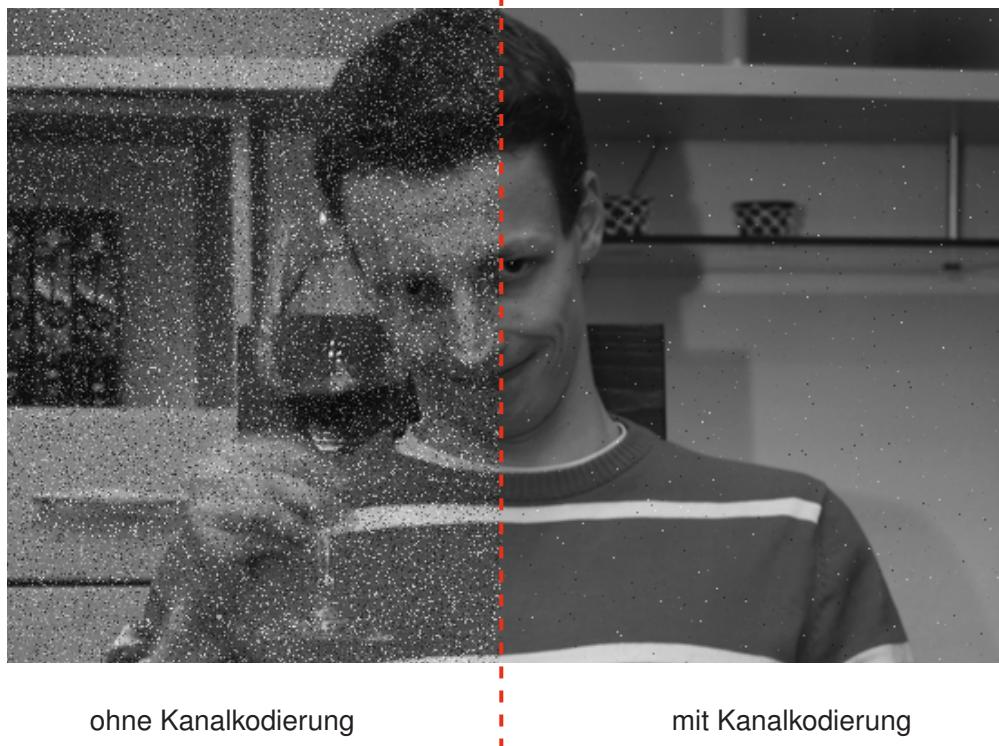
- ▶ Ungesicherte Funkverbindung mit $p_e = 10^{-4}$, Fehler unabhängig und gleichverteilt
 - ▶ Paketlänge $L = 1500 \text{ B} = 12000 \text{ bit}$
 - ▶ $\Pr[\text{„Kein Bitfehler im Paket“}] = (1 - 10^{-4})^{12000} \approx 30\%$
- ⇒ 70 % der übertragenen Datenpakete würden mind. einen Bitfehler enthalten.

Kanalkodierung (Channel Coding)

Ziel der Kanalkodierung ist es, den zu übertragenden Daten gezielt Redundanz hinzuzufügen, so dass eine möglichst große Anzahl an

- ▶ Bitfehlern erkannt und
- ▶ korrigiert werden kann.

Beispiel: Unkomprimiertes Bild (Bitmap) über einen verlustbehafteten Kanal versendet



Geringfügige Übertragungsfehler sind in analogen Systemen tolerierbar:

- ▶ Rauschen / Knacken bei einer Telefonverbindung
- ▶ „Schnee“ im analogen Fernsehen
- ▶ UKW-Radio

In digitalen Systemen können Übertragungsfehler schwerwiegende Konsequenzen haben, z.B.:

- ▶ Übertragung komprimierter Daten (ggf. Fehlerfortpflanzung bei der Dekodierung)
- ▶ Übertragung verschlüsselter Daten (ggf. Fehlerfortpflanzung bei der Entschlüsselung)
- ▶ Fehlerfreie Übertragung kann gefordert sein (z. B. heruntergeladenes Programm kann schon bei einzelnen Bitfehlern unbrauchbar sein)

Es werden also zusätzliche Protokolle und Mechanismen benötigt, um trotz Kanalkodierung auftretende Übertragungsfehler

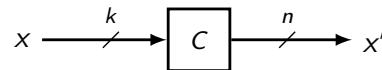
- ▶ zumindest zu erkennen und
- ▶ bei Bedarf eine Übertragung zu wiederholen.

⇒ Zusammenspiel von **Prüfsummen** und **Quittungsprotokollen**,
typischerweise auf Schichten 2, 4 bzw. 7.

Kanalkodierung: Blockcodes

Blockcodes unterteilen den Datenstrom

- ▶ in Blöcke der Länge k und
- ▶ übersetzen diese in Kanalwörter der Länge $n > k$ wobei
- ▶ die zusätzlichen $n - k$ bit für Fehlererkennung und Rekonstruktion verwendet werden.



Das Verhältnis $R = \frac{k}{n}$ wird als **Coderate** bezeichnet.

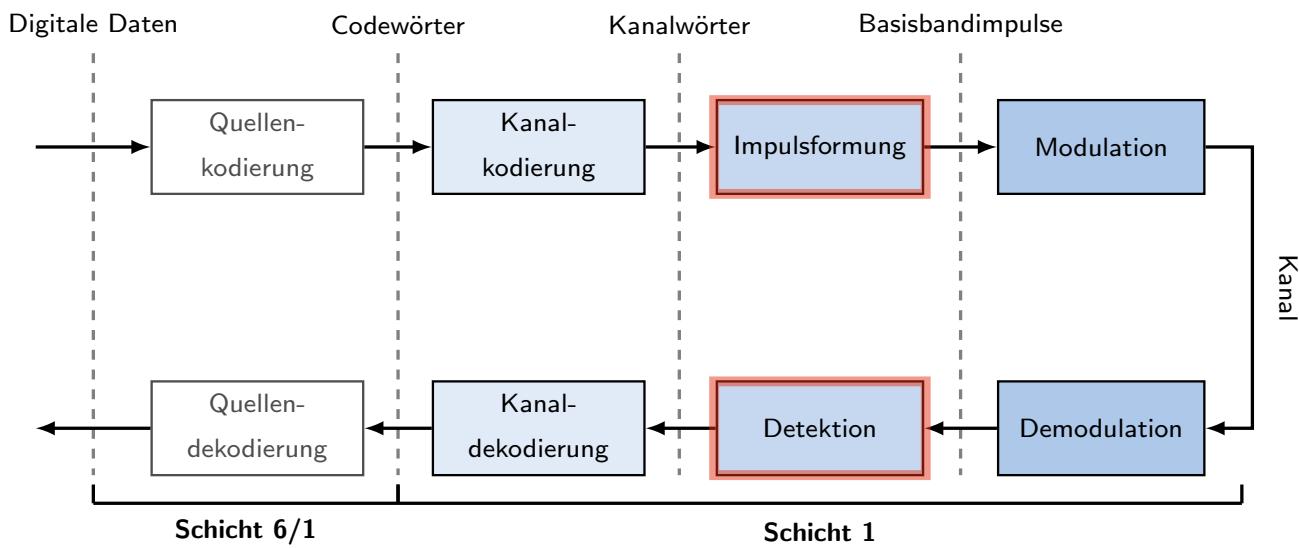
Beispiel: Repetition Code

- ▶ $k = 1, n = 3$, Abbildung: $0 \mapsto 000, 1 \mapsto 111$
- ▶ Dekodierfehler, wenn mind. 2 Bit pro Block verfälscht wurden:

$$\Pr[\text{„Dekodierfehler“}] = \binom{3}{2} p_e^2 (1 - p_e) + \binom{3}{3} p_e^3 \approx \Big|_{p_e=10^{-4}} 3 \cdot 10^{-8}$$

- ▶ Neues Problem:
 - ▶ Die zu sendende Anzahl an Bits wird verdreifacht
 - ▶ Im fehlerfreien Fall würde die erzielbare Datenrate also auf $R = 1/3$ sinken
- ⇒ Kosten-/Nutzenverhältnis zwischen Fehlerwahrscheinlichkeit und Redundanz abhängig von der momentanen Bitfehlerrate g

Übersicht



Impulsformung [1]

Impulsformung

Ziel der (**Basisband-**)**Impulsformung** ist es, aus einem Datenstrom ein analoges Basisbandsignal (Frequenzen f nahe Null) zu erzeugen. Hierzu werden

- ▶ in regelmäßigen Abständen **gewichtete** Sendegrundimpulse erzeugt,
- ▶ von denen jeder ein Bit oder eine Gruppe von Bits darstellt.

Beispiel:

- ▶ Gegeben sei der Datenstrom 00 01 10 11 00 11 00 11
- ▶ Rechteckimpulse werden mit einem von vier Symbolen $d \in \{\pm 0.5, \pm 1.5\}$ gewichtet
- ▶ Wir wählen eine Zuordnung zwischen Gruppen von $N = 2$ Bit und den $M = 4$ Symbolen:
 $00 \mapsto d = -1.5, 01 \mapsto d = -0.5, 10 \mapsto d = 0.5, 11 \mapsto d = 1.5$

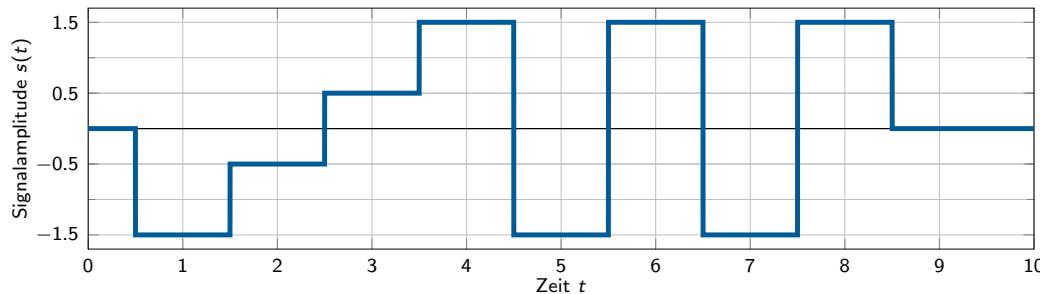
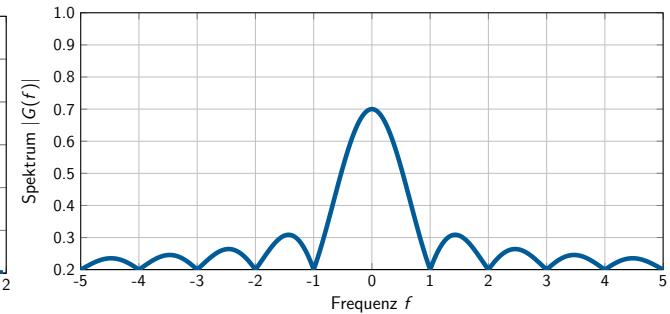
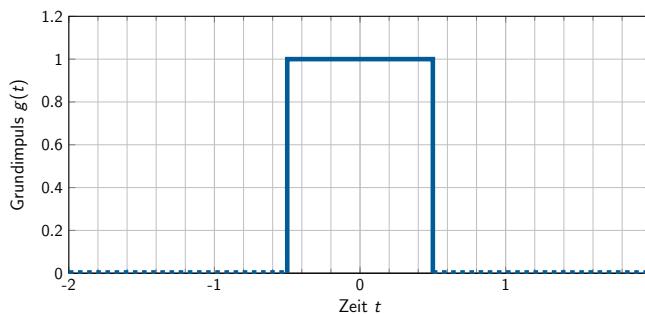


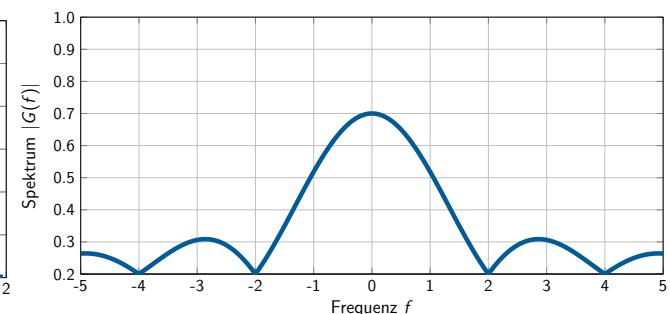
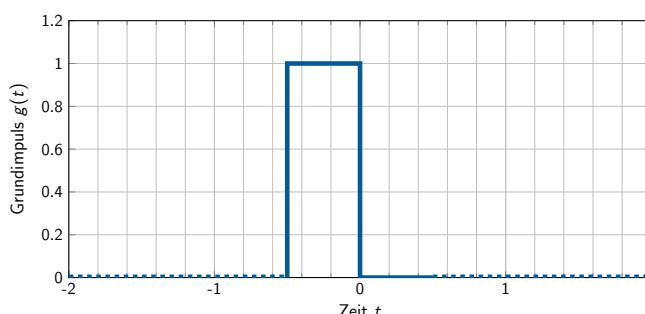
Abbildung: Basisbandsignal $s(t) = \sum_{n=1}^8 d_n \cdot \text{rect}(t - nT)$.

Grundimpulse (Beispiele)

- ▶ Non-Return-to-Zero-Impuls (NRZ) bzw. Rechteckimpuls

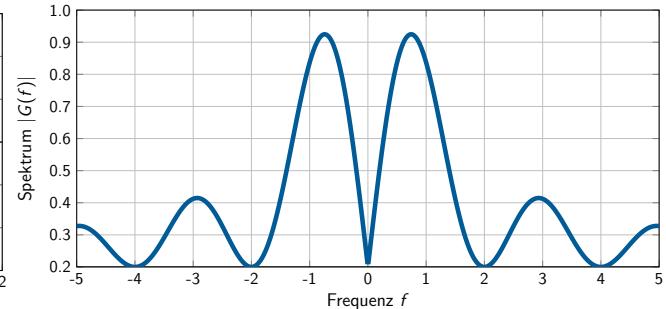
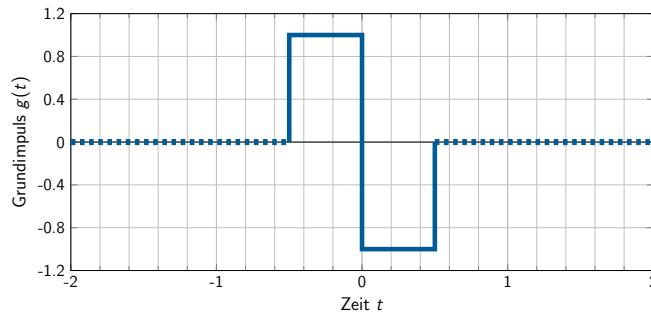


- ▶ Return-to-Zero-Impuls (RZ)

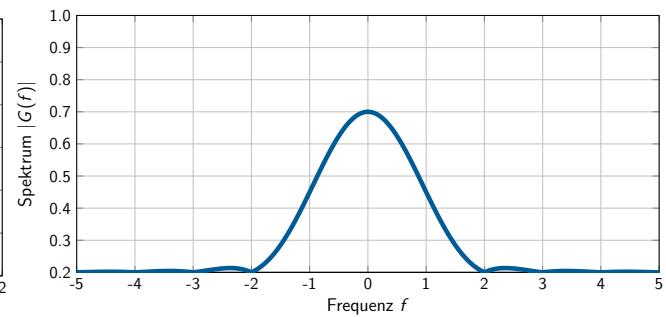
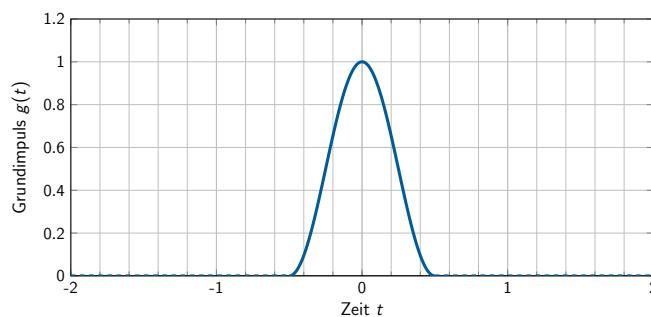


Grundimpulse (Beispiele)

- ▶ Manchester-Impuls



- ▶ \cos^2 -Impuls



Leitungscodes

Leitungscodes (nicht zu verwechseln mit **Kanalcodes**) definieren die Abfolge von einer bestimmten Art von Grundimpulsen, welche Bits oder Gruppen von Bits repräsentieren.

Definition: Symbol

Im Kontext von Leitungscodes verstehen wir unter einem **Symbol** eine phys. messbare Veränderung des Zeitsignals. Einzelne Grundimpulse bestehen also aus einem oder mehreren Symbolen.

Wichtige Eigenschaften von Leitungscodes:

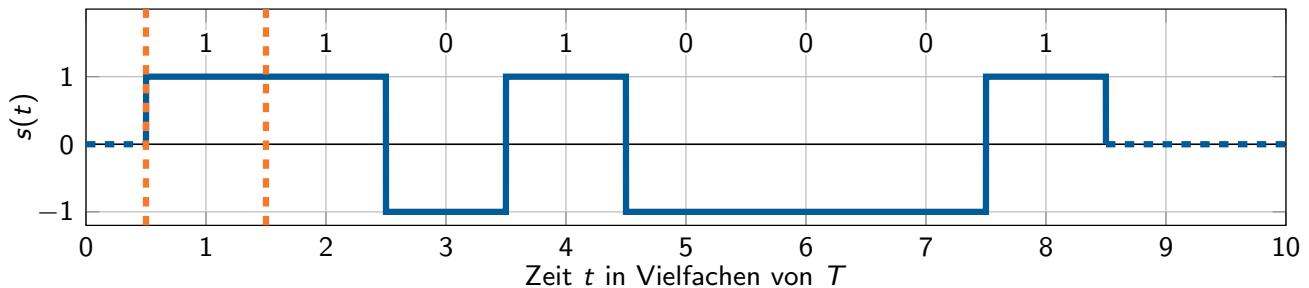
- ▶ Anzahl der Signalstufen (binär, ternär, ...)
- ▶ Anzahl kodierter Bits pro Symbol
- ▶ Schrittgeschwindigkeit (Symbolrate, Baudrate), Einheit bd

Optionale Eigenschaften von Leitungscodes:

- ▶ Taktrückgewinnung
- ▶ Gleichstromfreiheit
- ▶ Bereitstellung von Steuerzeichen (4B5B-Kodierung → später)

Je nach Art der verwendeten Grundimpulse und deren Abfolge haben Leitungscodes Einfluss auf die benötigte Kanalbandbreite. Als Daumenregel gilt: **Je mehr abrupte Signalwechsel stattfinden, desto breiter ist das benötigte Spektrum.** (s. Beispiele)

Non-Return-To-Zero (NRZ)



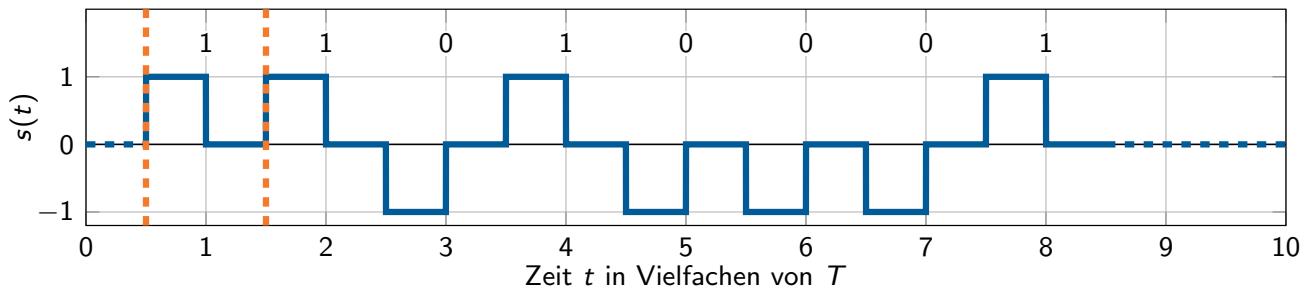
Kodievorschrift:

- ▶ Grundimpuls $g(t) = \text{rect}(t)$ (Rechteckimpuls) mit Periodendauer T
- ▶ Mögliche Zuweisung der Gewichte $d_n = \begin{cases} 1 & b_n = 1 \\ -1 & b_n = 0 \end{cases}$
- ▶ Sendesignal ist definiert als $s(t) = \sum_{n=0}^{\infty} d_n \cdot \text{rect}(t - nT)$

Eigenschaften:

- ▶ Binärer Code (lediglich zwei Signalstufen)
- ▶ Effizienz 1 bit / Symbol
- ▶ Keine Taktrückgewinnung (lange Null- oder Einsfolgen)
- ▶ Keine Gleichstromfreiheit
- ▶ Relativ breites Spektrum

Return-To-Zero (RZ)



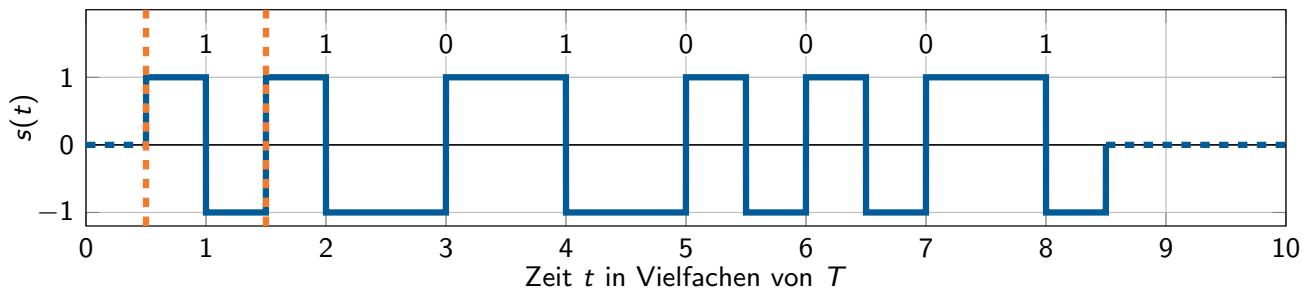
Kodievorschrift:

- ▶ Grundimpuls $g(t) = \text{rz}(t)$ (Return-to-Zero) mit Periodendauer T
- ▶ Mögliche Zuweisung der Gewichte $d_n = \begin{cases} 1 & b_n = 1 \\ -1 & b_n = 0 \end{cases}$
- ▶ Sendesignal ist definiert als $s(t) = \sum_{n=0}^{\infty} d_n \cdot \text{rz}(t - nT)$

Eigenschaften:

- ▶ Binärer Code (lediglich zwei Signalstufen)
- ▶ Effizienz 1 bit / 2 Symbole
- ▶ Taktrückgewinnung durch erzwungene Pegelwechsel einfach
- ▶ Keine Gleichstromfreiheit
- ▶ Breiteres Spektrum als NRZ

Manchester-Code



Kodievorschrift:

- ▶ Grundimpuls $g(t) = \text{manc}(t)$ (Manchester) mit Periodendauer T
- ▶ Mögliche Zuweisung der Gewichte $d_n = \begin{cases} 1 & b_n = 1 \\ -1 & b_n = 0 \end{cases}$
- ▶ Sendesignal ist definiert als $s(t) = \sum_{n=0}^{\infty} d_n \cdot \text{manc}(t - nT)$

Eigenschaften:

- ▶ Binärer Code (lediglich zwei Signalstufen)
- ▶ Effizienz 1 bit/2 Symbole
- ▶ Taktrückgewinnung durch erzwungene Pegelwechsel einfach
- ▶ Gleichstromfreiheit gewährleistet, da jeder Grundimpuls für sich gleichstromfrei ist
- ▶ Sehr breites und langsam abklingendes Spektrum

Multi-Level-Transmit 3 (MLT3)



Kodievorschrift:

- ▶ Grundimpuls $g(t) = \text{rect}(t)$ (Rechteckimpuls) mit Periodendauer T
- ▶ Gewichte $d_n = \sin\left(\frac{\pi}{2} \sum_{k=0}^n b_k\right)$ (\rightarrow abhängig von der Anzahl der bislang beobachteten 1-Bits)
- ▶ Sendesignal ist definiert als $s(t) = \sum_{n=0}^{\infty} d_n \cdot \text{rect}(t - nT)$

Eigenschaften:

- ▶ Terner Code (drei Signalstufen)
- ▶ Effizienz 1 bit/ Symbol
- ▶ Keine Taktrückgewinnung (lange Folge gleicher Bits)
- ▶ Keine Gleichstromfreiheit
- ▶ Schmales Spektrum, da die Grundperiode durch den periodischen Signalverlauf reduziert wird

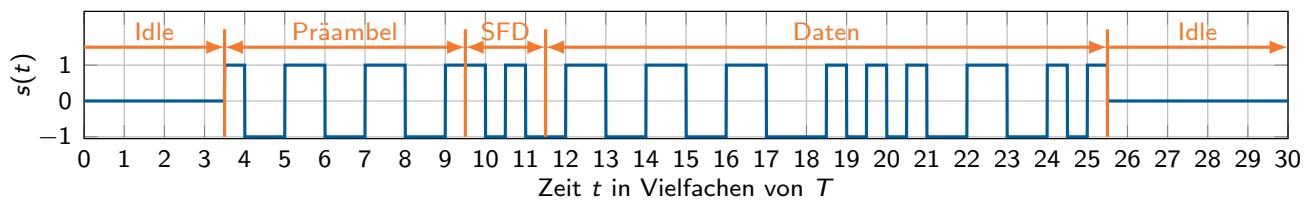
Offene Fragen: Wie kann der Empfänger erkennen,

- ▶ ob detektierte Symbole überhaupt Daten repräsentieren (Medium könnte „idle“ sein) und
- ▶ wie kann der Beginn bzw. das Ende einer Nachricht erkannt werden?

Möglichkeit 1: Coderegelverletzung

- ▶ Ist das Medium idle, können ungültige Basisbandimpulse gesendet werden
- ▶ Vor Beginn einer Nachricht kann eine fest definierte Anzahl alternierender Bits gesendet werden (**Präambel**)
- ▶ Beginn der Nachricht wird durch eine zweite Sequenz angezeigt (**Start Frame Delimiter**)
- ▶ Dies funktioniert mit NRZ, RZ und Manchester Code (z. B. Nullpegel), nicht aber mit MLT3 (Nullpegel bedeutet hier eine Folge von 0-bits)

Beispiel: Manchester-Code mit Präambel



- ▶ Präambel ermöglicht Taksynchronisation
- ▶ Start Frame Delimiter (SFD) am Ende der Präambel signalisiert Beginn der Nachricht
- ▶ Coderegelverletzung (Nullpegel) zeigt Idle-Zustand an
- ▶ Verwendet bei **IEEE 802.3a/i** (10 Mbit/s Ethernet über Koaxial- bzw. Twisted-Pair-Kabel → später)

Möglichkeit 2: Steuerzeichen

- ▶ Definiere einen Blockcode, welcher Kanalwörter in Gruppen von k Bits unterteilt und auf $n > k$ Bits abbildet
- ▶ Dieser Blockcode dient **nicht der Fehlerkorrektur** (Aufgabe der Kanalkodierung), sondern lediglich der **Bereitstellung von Steuerzeichen**
- ▶ Die Abbildung kann dabei so gewählt sein, dass bei der Übertragung gültiger Kanalwörter
 - ▶ Taktrückgewinnung und
 - ▶ Gleichstromfreiheit
auch mit Leitungscodes wie NRZ, RZ und MLT3 möglich werden.
- ▶ Ungültige Codewörter, die weder Datenwörter noch Steuerzeichen darstellen, können zur **Fehlererkennung** verwendet werden

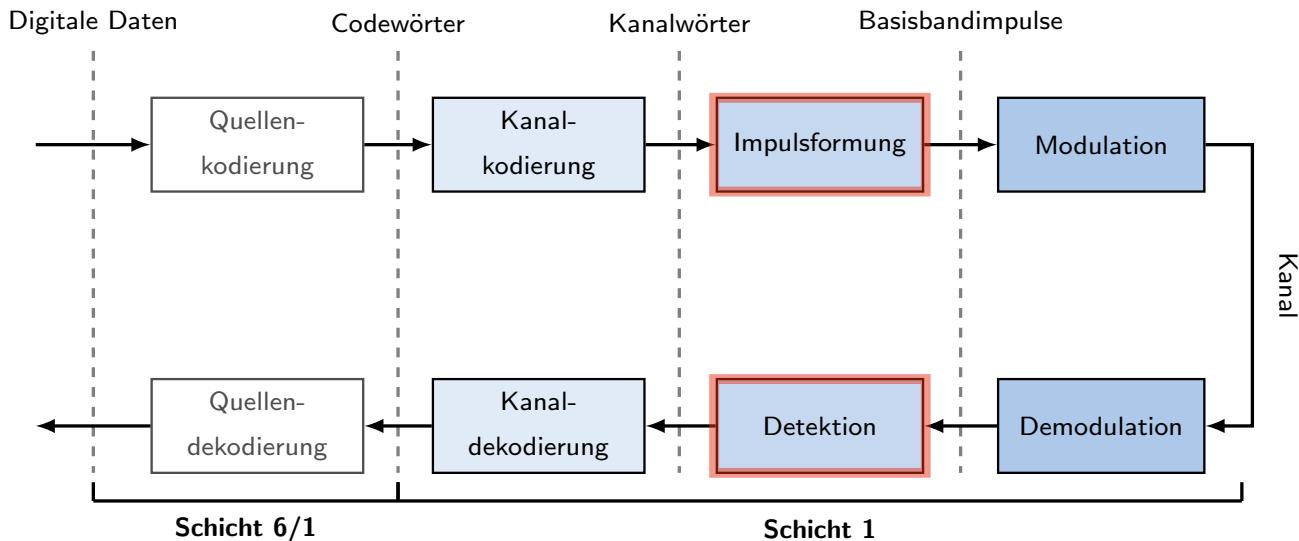
Beispiel 1: 4B5B-Code

- ▶ $k = 4$ bit werden auf $n = 5$ bit abgebildet
- ▶ Die Zuordnung zwischen Kanalwörtern und Codewörtern wird so gewählt, dass in jedem Block von 5 bit mind. ein Signalwechsel auftritt (Taktrückgewinnung bei NRZ und MLT3)
- ▶ Die zusätzlichen Codewörter werden als Steuerzeichen verwendet (Start/Stop, Idle, ...)
- ▶ Verwendet bei **IEEE 802.3u** (100 Mbit/s FastEthernet über Twisted-Pair-Kabel)

Beispiel 2: 8B10B-Code

- ▶ $k = 8$ bit werden auf $n = 10$ bit abgebildet
- ▶ Zuordnung ähnlich wie bei 4B5B, allerdings wird hier im zeitlichen Mittel auch Gleichstromfreiheit gewährleistet
- ▶ Verwendet u. a. bei PCIe, Serial-ATA, USB ...

Übersicht



Modulation [3]

Bislang haben wir nur **Basisbandsignale** betrachtet:

- Zeitlich verschobene Grundimpulse werden gewichtet
- Zeitlich begrenzte Grundimpulse (wir haben nur solche kennengelernt) besitzen ein **unendlich ausgedehntes Spektrum**
- Sofern der Übertragungskanal exklusiv für die Basisbandübertragung zur Verfügung steht, ist das zunächst kein Problem

Was ist, wenn der Kanal von mehreren Übertragungen **zeitgleich** verwendet wird?

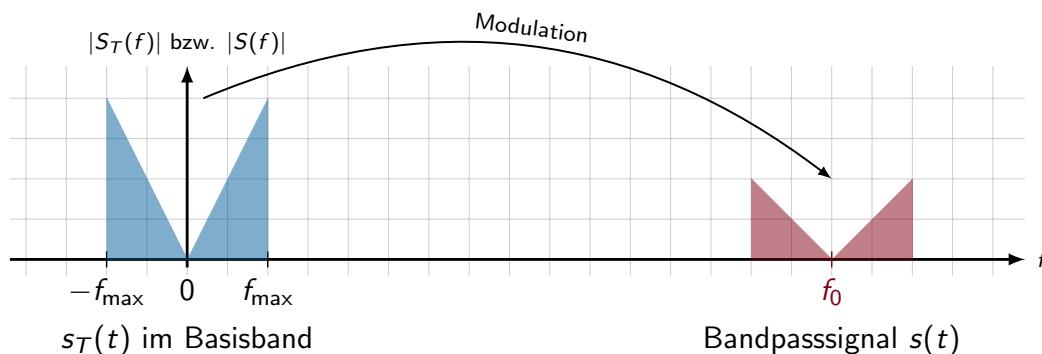
- Das Basisbandsignal (bzw. dessen Grundimpulse) wird **tiefpass-gefiltert**, was eine Begrenzung des Spektrums (und damit einer leichten Verfälschung des Zeitsignals) entspricht
- Anschließend kann das gefilterte Basisbandsignal auf ein **Trägersignal moduliert** werden
- Dies entspricht einer **Verschiebung des Spektrums** (Multiplikation im Zeitbereich ist bedingt eine Verschiebung im Frequenzbereich)
- Teilen sich mehrere Übertragungen auf diese Art einen Kanal, so sprechen wir von **Frequency Division Multiplex (FDM)**

Prinzipieller Ablauf digitaler Modulationsverfahren

- ▶ Die Grundimpulse $g(t)$ werden mittels Tiefpassfilterung auf eine maximale Frequenz f_{\max} beschränkt. Die so gefilterten Impulse bezeichnen wir als $g_T(t)$.
- ▶ Das Modulationssignal $s_T(t)$ wird wie im Basisband durch eine gewichtete Überlagerung zeitlich verschobener Grundimpulse erzeugt.
- ▶ Das Modulationssignal wird auf ein **Trägersignal** der Frequenz f_0 aufmoduliert:

$$s(t) = s_T(t) \cdot \sin(2\pi f_0 t) = \left(\sum_{n=0}^{\infty} d_n \cdot g_T(t - nT) \right) \sin(2\pi f_0 t).$$

Schematischer Ablauf im Frequenzbereich:

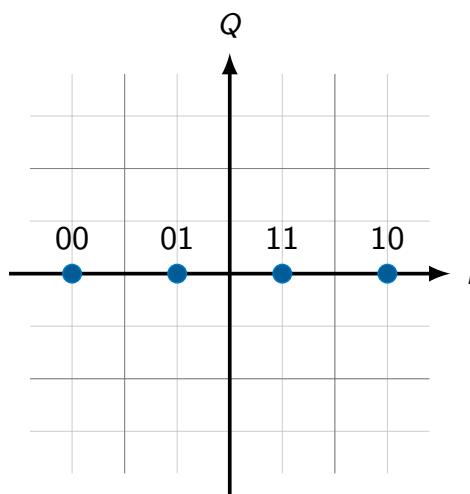


4-ASK (Amplitude Shift Keying)

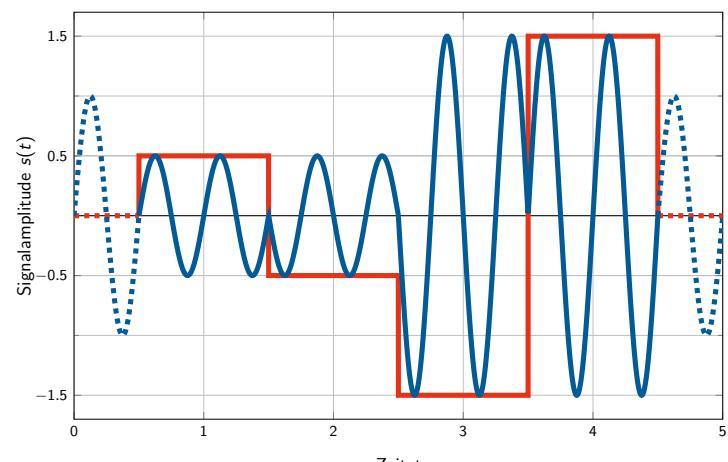
- ▶ Es werden 4 Signalstufen unterschieden $\Rightarrow 2 \text{ bit/Symbol}$
- ▶ Es wird nur die Amplitude des Trägersignals moduliert

Beispiel: Signalraum $S = \{-\frac{3}{2}, -\frac{1}{2}, \frac{1}{2}, \frac{3}{2}\}$

- ▶ Je zwei aufeinanderfolgende Bits des Datenstroms werden auf ein Symbol $d \in S$ abgebildet, z. B. $00 \mapsto -\frac{3}{2}, 01 \mapsto -\frac{1}{2}, \dots$
- ▶ Die Symbolsequenz d_n verändert die Amplitude eines Grundimpulses (z. B. Rechteckimpuls)
- ▶ Das so entstehende Basisbandsignal wird mit einem Trägersignal multipliziert (Modulation)



(a) Signalraumuordnung

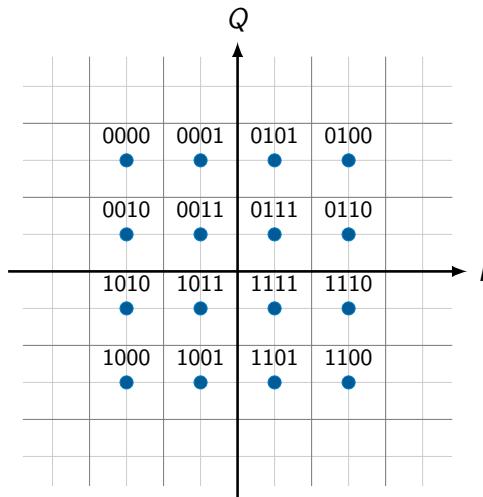


(b) Sendesignal $s(t)$ (blau), Modulationssignal $s_T(t)$ (rot)

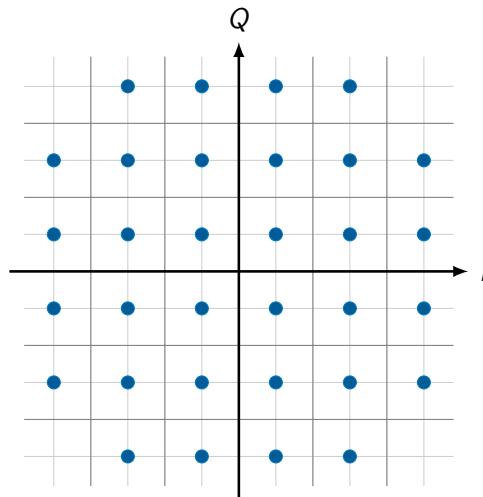
Quadratur-Amplituden-Modulation (QAM)

- ▶ Man kann kosinus- und sinus-förmige Trägersignale mischen
- ▶ Trennung durch Orthogonalität von Sinus und Kosinus möglich
- ▶ Der Kosinus wird als **Inphase-Anteil**, der Sinus als **Quadratur-Anteil** bezeichnet
- ▶ Die Datenrate lässt sich auf diese Weise verdoppeln

$$s(t) = \left(\sum_{n=0}^{\infty} d_{In} \cdot g_T(t - nT) \right) \cos(2\pi f_0 t) - \left(\sum_{n=0}^{\infty} d_{Qn} \cdot g_T(t - nT) \right) \sin(2\pi f_0 t)$$



(c) 16-QAM

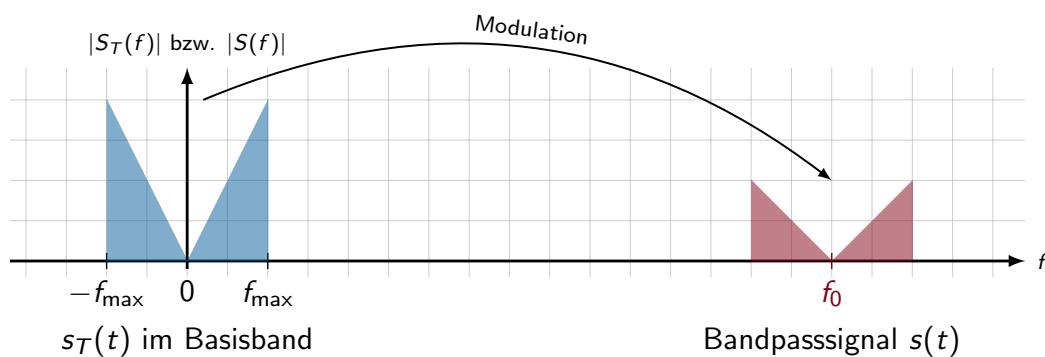


(d) 32-QAM

- ▶ QAM verdoppelt die Datenrate nochmals?
- ▶ Sensationell, wir haben Shannon widerlegt!

Natürlich nicht: [4] Durch die Frequenzverschiebung belegt das Bandpasssignal die **doppelte Bandbreite** im Vergleich zum Basisbandsignal. Es entsteht ein

- ▶ **oberes Seitenband**, welches den nicht-negativen Frequenzanteilen im Basisband entspricht, sowie ein
- ▶ **unteres Seitenband**, welches den nicht-positiven Frequenzanteilen im Basisband entspricht.



- ▶ Durch die Modulation wurde also die benötigte Bandbreite verdoppelt
 - ▶ Dieser „verlorene Freiheitsgrad“ kann durch die Mischung von Sinus- und Kosinus-Trägern wieder kompensiert werden.

Die obere Schranke für die erzielbare Datenrate gilt natürlich weiterhin.

Zusammenfassung

Was wir wissen sollten:

- ▶ Was sind die Unterschiede und Ziele zwischen **Quellenkodierung**, **Kanalkodierung** und **Leitungskodierung**?
- ▶ Wie funktionieren einfache **Block-Codes**, z. B. Repetition-Code?
- ▶ Warum werden trotz aller Kodierverfahren zusätzliche Verfahren zur **Fehlererkennung** benötigt?
- ▶ Wie funktionieren die in diesem Kapitel eingeführten **Leitungscodes**?
- ▶ Was sind die jeweiligen Vor- und Nachteile der hier eingeführten Leitungscodes?
- ▶ Wie könnte man diese Leitungscodes auf mehr als zwei oder drei Signalstufen erweitern?
- ▶ Was ist das Prinzip von **Modulationsverfahren**?
- ▶ Wie funktioniert **Frequenzmultiplex**?
- ▶ Wie hängen Signalraumzuordnung, Modulationsverfahren und die erzielbare Datenrate zusammen?
- ▶ **Für Interessierte:** Wie funktioniert **Phase Shift Keying (PSK)** und wie sieht eine gültige Signalraumzuordnung für PSK aus?

Inhalt

Signale, Information und deren Bedeutung

Klassifizierung von Signalen

Zeit- und Frequenzbereich
Abtastung, Rekonstruktion und Quantisierung

Übertragungskanal

Kanaleinflüsse
Kanalkapazität

Nachrichtenübertragung

Übertragungsmedien

Übertragungsmedien

Wir unterscheiden zwischen

- ▶ leitungsgebundener und
- ▶ nicht-leitungsgebundener Übertragung

sowie zwischen

- ▶ akustischen und
- ▶ elektromagnetischen Wellen.

Im Bereich der digitalen Datenübertragung kommen überwiegend elektromagnetische Wellen zum Einsatz. Wenige Ausnahmen bilden hier

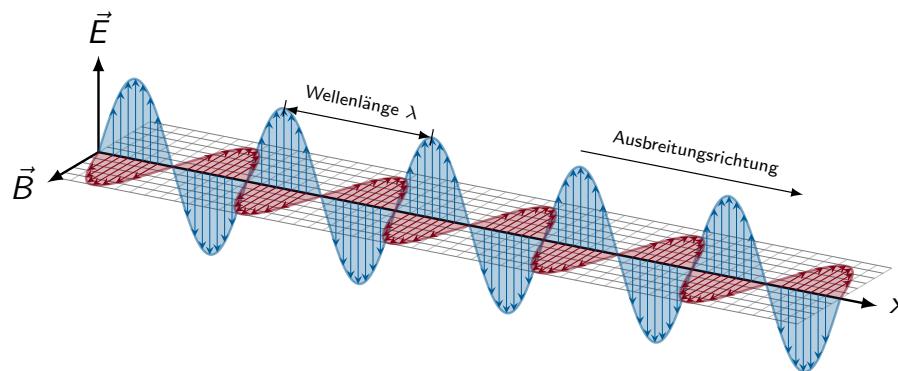
- ▶ Tonwahlverfahren (z. B. „Einwahl“ bei alten Modemverbindungen) sowie
- ▶ einige experimentelle Verfahren, z. B. kabellose Kommunikation unter Wasser.

Im Folgenden verschaffen wir uns einen Überblick über

- ▶ Frequenzen im EM-Spektrum,
- ▶ was EM-Wellen überhaupt sind und
- ▶ und welche Arten von Übertragungsmedien bei leitungsgebundenen Verfahren häufig zum Einsatz kommen.

Elektromagnetische Wellen

Elektromagnetische Wellen bestehen aus einer elektrischen (\vec{E}) und magnetischen (\vec{B}) Komponente, welche jeweils orthogonal zueinander und zur Ausbreitungsrichtung stehen:

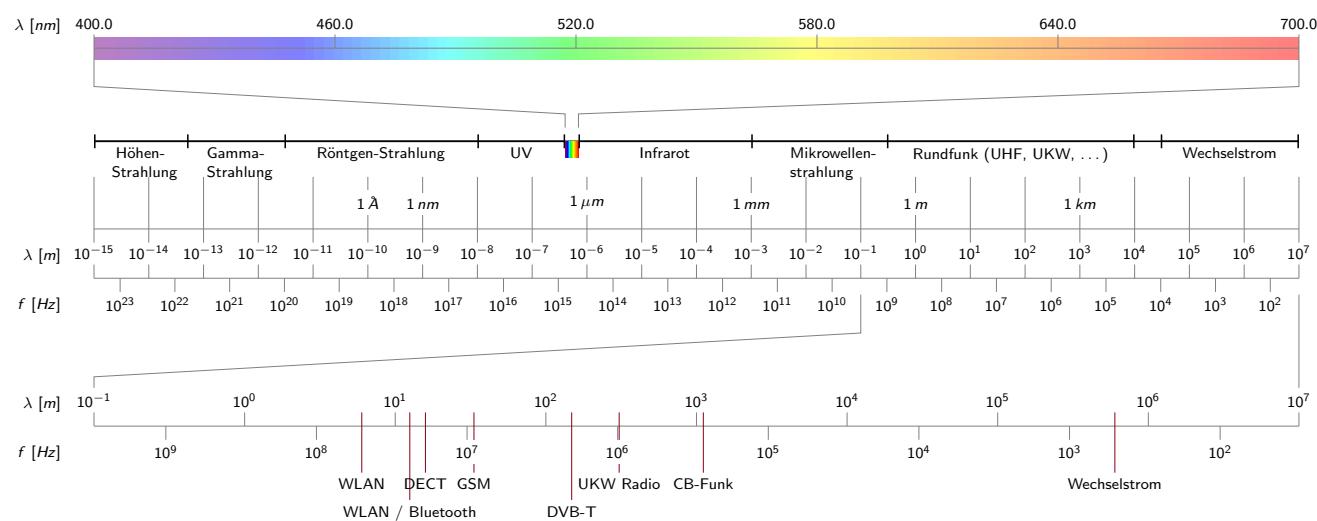


Wichtige Eigenschaften:

- ▶ Ausbreitung im Vakuum mit Lichtgeschwindigkeit $c \approx 3 \cdot 10^8$ m/s
- ▶ Im Gegensatz zu Schallwellen wird kein Medium zur Ausbreitung benötigt
- ▶ Innerhalb eines Mediums (Leiter, Luft) beträgt die Ausbreitungsgeschwindigkeit νc , wobei $0 < \nu < 1$ als **relative Ausbreitungsgeschwindigkeit** bezeichnet wird, z. B. $\nu = 0.9$ in Lichtwellenleitern oder $\nu = 2/3$ in Koaxialleitern
- ▶ Die **Wellenlänge** λ beschreibt die räumliche Ausdehnung einer Wellenperiode
- ▶ Die **Frequenz** f ergibt sich aus Wellenlänge und Lichtgeschwindigkeit zu $f = c/\lambda$

Elektromagnetisches Spektrum

Die untenstehende Abbildung zeigt eine schematische Darstellung des EM-Spektrums:

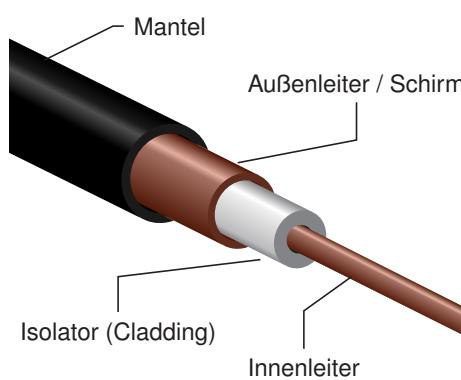


Zur digitalen Datenübertragung werden überwiegend genutzt:

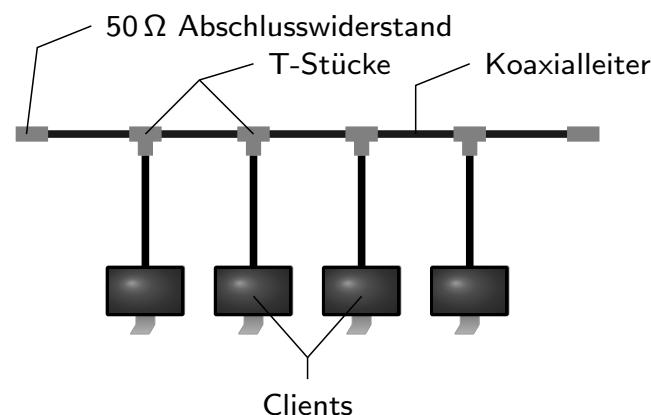
- ▶ das Frequenzband zwischen einigen MHz und einigen GHz,
- ▶ das optische Spektrum bis zu etwa $\lambda \approx 1\text{ nm}$ sowie
- ▶ Frequenzen im Basisband bis zu einigen hundert MHz.

Koaxialkabel

- ▶ Eingesetzt u.a. für IEEE 802.3a („10Base2 Ethernet“), 10 Mbit/s
- ▶ Ein langer gemeinsamer Bus, an den alle Teilnehmer angeschlossen sind
- ▶ Heute (bis auf Kabelverteilnetze, insb. Cable TV, sowie industrielles Umfeld) von geringerer Bedeutung
- ▶ Ähnliche Koaxialkabel (mit anderer Dämpfung) kommen im TV-Kabelnetz zum Einsatz



(a) Schematischer Aufbau



(b) 10Base2 Bus (IEEE 802.3a)

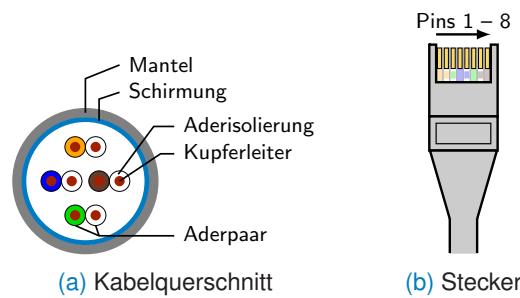
Twisted-Pair-Kabel

Allgemeines:

- ▶ 2 oder 4 Aderpaare aus Kupferlitzen
- ▶ Jedes Aderpaar ist verdrillt (daher die Bezeichnung **twisted pair**)
- ▶ Zweite Ader eines Paars führt inversen Signalpegel (differentielle Kodierung)
- ▶ Verdrillung und inverse Signalpegel reduzieren **Übersprechen (Crosstalk)**
- ▶ RJ-45 oder schmälerer RJ-11 Steckverbinder

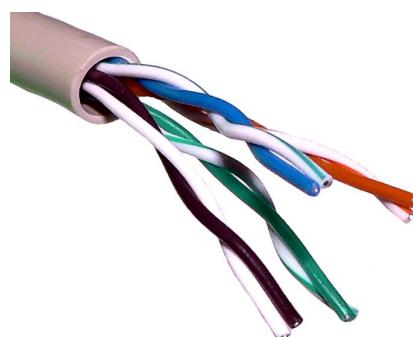
Verwendung:

- ▶ Lokale Netzwerke (die meisten Ethernet-Standards) mit RJ-45 Steckverbinder
- ▶ Telefonanschluss (analog und ISDN) mit RJ-11 Steckverbinder

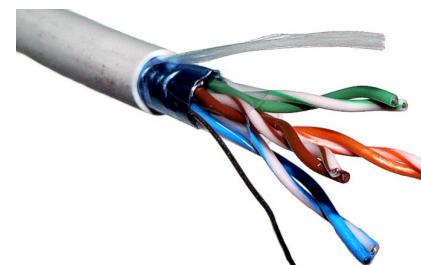


Je nach Schirmung unterscheidet man

- ▶ UTP (unshielded twisted pair)
- ▶ STP (shielded twisted pair)
- ▶ S/UTP (screened / unshielded twisted pair)
- ▶ S/STP (screened / shielded twisted pair)



(c) UTP



(d) Screened UTP

Schirmung hat Einfluss auf

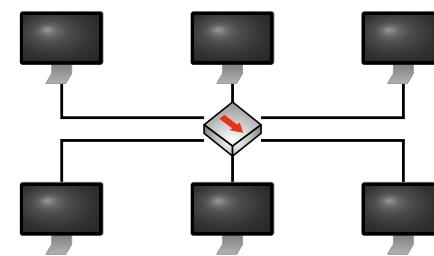
- ▶ die Signalqualität (z. B. Übersprechen von elektr. Leitungen) und
- ▶ die Flexibilität der Kabel (gut geschirmte Kabel sind steifer).

Twisted-Pair-Kabel für 100BASE-TX

- ▶ Verbindung mehrerer Computer über Hub mittels Straight-Through-Kabel

PC		Hub
Tx+	1	Rx+
Tx-	2	Rx-
Rx+	3	Tx+
	4	4
	5	5
Rx-	6	Tx-
	7	7
	8	8

(a) Straight-Through



(b) Hub erzeugt phys. Bus, halbduplex

- ▶ Direktverbindung zweier Computer mittels Cross-Over-Kabel

PC		PC
Tx+	1	Tx+
Tx-	2	Tx-
Rx+	3	Rx+
	4	4
	5	5
Rx-	6	Rx-
	7	7
	8	8

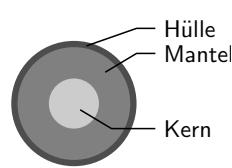
(a) Cross-Over



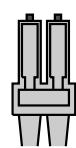
(b) Punkt-zu-Punkt, vollduplex

Optische Leiter

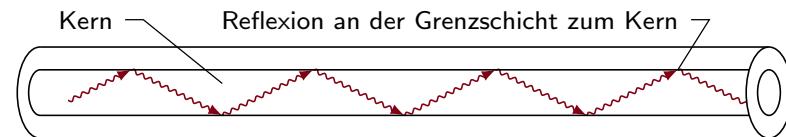
- ▶ Licht wird innerhalb des Faserkerns weitergeleitet
- ▶ Kern und Mantel besitzen jeweils unterschiedliche optische Dichten
→ Brechungsindex sorgt für annähernde Totalreflexion
- ▶ Single-Mode-Fasern vermeiden Streuung durch sehr geringen Kerndurchmesser
→ geringe Verluste, aber sehr empfindlich (Kabelbruch)
- ▶ Multi-Mode-Fasern haben einen größeren Kerndurchmesser und neigen daher zum Streuen
→ höhere Verluste, aber weniger empfindlich



(a) Kabelquerschnitt



(b) LC-Stecker



(c) Seitenansicht einer Faser

Vorteile gegenüber elektrischen Leitern:

- ▶ Sehr hohe Datenraten möglich
- ▶ Weite Strecken überbrückbar
- ▶ Kein Übersprechen
- ▶ Galvanische Entkopplung von Sender und Empfänger

Zusammenfassung

Zur digitalen Kommunikation werden **elektromagnetische Wellen**

- ▶ im Frequenzbereich bis zu einigen GHz bzw.
- ▶ im optischen Spektrum genutzt.

Als Übertragungsmedien kommen

- ▶ **elektrische Leiter** (Kupferkabel) sowie
- ▶ **optische Leiter**

in verschiedenen Ausführungen zum Einsatz.

- ▶ **Funkübertragungen** benötigen kein Medium, da sich elektromagnetische Wellen (im Gegensatz zu Schallwellen) im Vakuum ausbreiten
- ▶ Das verwendete Medium hat Einfluss auf die **Ausbreitungsgeschwindigkeit**

Im nächsten Kapitel beantworten wir die Fragen,

- ▶ wie Knoten auf ein ggf. gemeinsames Medium zugreifen können (**Medienzugriff**) und
- ▶ wie Nachrichten an einen bestimmten benachbarten Knoten gesendet werden können (**Adressierung**).

Bibliography I

- [1] E. Stein. *Taschenbuch Rechnernetze und Internet*, chapter Codierung und Modulation, pages 59–66. Fachbuchverlag Leipzig, 2. edition, 2004. Auszug s. Moodle/SVN.
- [2] M. Werner. *Nachrichtentechnik – eine Einführung für alle Studiengänge*, chapter Digitale Signalverarbeitung und Audio-Codierung, pages 72 – 80. Vieweg + Teubner, 6. edition, 2007. Auszug s. Moodle/SVN.
- [3] M. Werner. *Nachrichtentechnik – eine Einführung für alle Studiengänge*, chapter Digitale Übertragung im Basisband, pages 127 – 136. Vieweg + Teubner, 6. edition, 2007. Auszug s. Moodle/SVN.
- [4] M. Werner. *Nachrichtentechnik – eine Einführung für alle Studiengänge*. Vieweg + Teubner, 6. edition, 2007.

Grundlagen Rechnernetze und Verteilte Systeme

Kapitel 2 – Sicherungsschicht

Worum geht es in diesem Kapitel?

Problemstellung und Motivation

Darstellung von Netzwerken als Graphen

Netztopologien
Adjazenz- und Distanzmatrix
Erzeugung von Baumstrukturen

Verbindungscharakterisierung, Mehrfachzugriff, Medienzugriffskontrolle

Verbindungscharakterisierung
Medienzugriff
ALOHA und Slotted ALOHA
CSMA, CSMA/CD, CSMA/CA
Token Passing

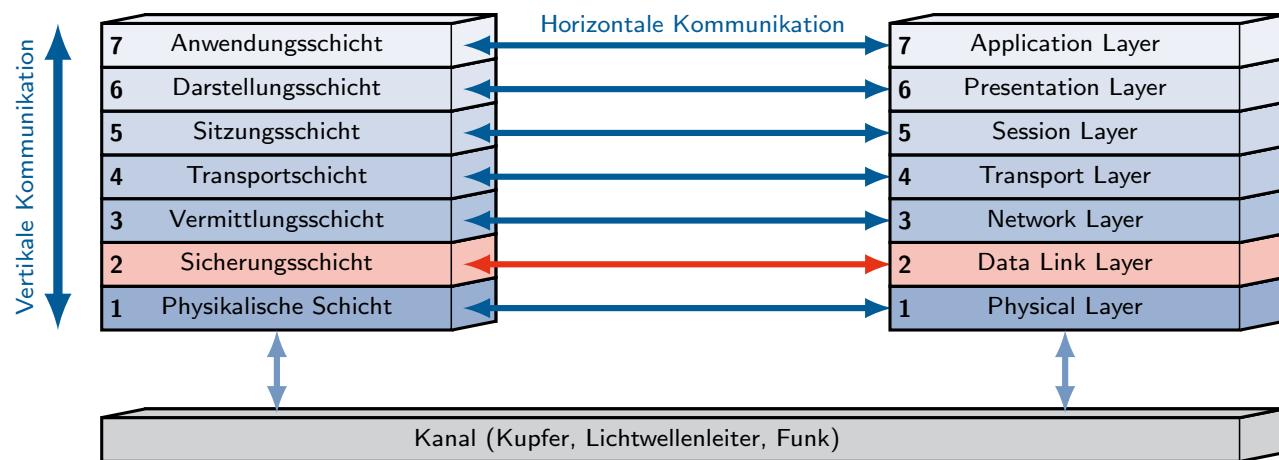
Rahmenbildung, Adressierung und Fehlererkennung

Erkennung von Rahmengrenzen und Codetransparenz
Adressierung und Fehlererkennung

Verbindung auf Schicht 1 und 2

Hubs, Bridges und Switches

Einordnung im ISO/OSI-Modell



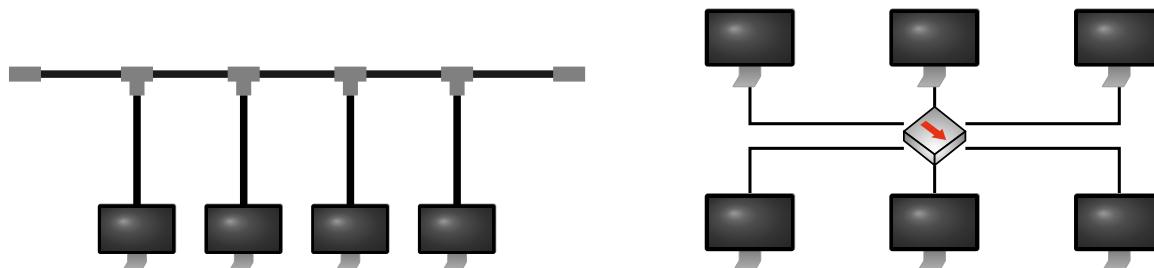
Problemstellung und Motivation

Wir beschäftigen uns zunächst mit sog. **Direktverbindungsnetzen**, d. h.

- ▶ alle angeschlossenen Knoten sind **direkt erreichbar** und
- ▶ werden mittels **einfacher Adressen** der Schicht 2 identifiziert,
- ▶ es findet **keine Vermittlung** statt,
- ▶ eine **einfache Weiterleitung** (in Form von „Bridging“ oder „Switching“) ist aber möglich.

Beispiele:

- ▶ einzelne lokale Netzwerke



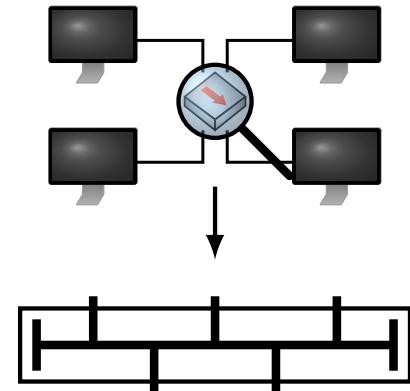
- ▶ Verbindung zwischen Basisstation und Mobiltelefon
- ▶ Bus-Systeme innerhalb eines Computers, z. B. PCIe

Die wesentlichen Aufgaben der Sicherungsschicht sind

- ▶ die **Steuerung des Medienzugriffs**,
- ▶ die **Prüfung übertragener Nachrichten** auf Fehler und
- ▶ die **Adressierung** innerhalb von Direktverbindungsnetzen.

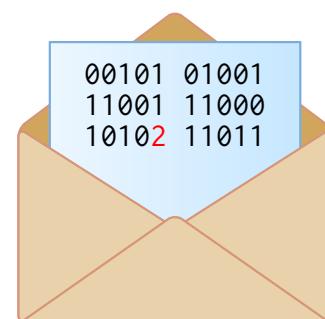
Steuerung des Medienzugriffs:

- ▶ **Hubs** z. B. erzeugen nur auf den ersten Blick eine Stern topologie
- ▶ Intern werden alle angeschlossenen Computer zu einem **Bus** verbunden
- ▶ Gleichzeitiges Senden von zwei Stationen führt zu **Kollisionen** und daher zum Verlust von Nachrichten



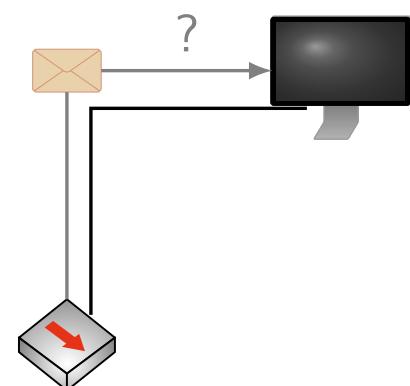
Prüfung übertragener Nachrichten auf Fehler:

- ▶ Trotz Kanalkodierung treten Übertragungsfehler auf
- ▶ Diese müssen erkannt werden
- ▶ Defekte Nachrichten werden nicht an höhere Schichten weitergegeben
- ▶ Die **Wiederholung** einer Übertragung ist häufig Aufgabe höherer Schichten



Adressierung:

- ▶ Eine Nachricht kann von vielen Knoten empfangen werden, z. B. bei Bus-Verbindungen oder Funknetzwerken
- ▶ Der jeweilige Empfänger muss entscheiden können, ob eine Nachricht für ihn bestimmt ist



Worum geht es in diesem Kapitel?

Problemstellung und Motivation

Darstellung von Netzwerken als Graphen

Netztopologien
Adjazenz- und Distanzmatrix
Erzeugung von Baumstrukturen

Verbindungscharakterisierung, Mehrfachzugriff, Medienzugriffskontrolle

Verbindungscharakterisierung
Medienzugriff
ALOHA und Slotted ALOHA
CSMA, CSMA/CD, CSMA/CA
Token Passing

Rahmenbildung, Adressierung und Fehlererkennung

Erkennung von Rahmengrenzen und Codetransparenz
Adressierung und Fehlererkennung

Verbindung auf Schicht 1 und 2

Hubs, Bridges und Switches

Darstellung von Netzwerken als Graphen

Motivation

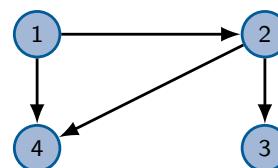
- ▶ Zur Darstellung von Netztopologien und Knotenverbindungen werden häufig gerichtete oder ungerichtete Graphen verwendet.
- ▶ Im Folgenden führen wir die entsprechende Notation und grundlegende Begriffe ein.

Gerichtete Graphen

Ein **asymmetrisches** Netzwerk lässt sich als **gerichteter** Graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ darstellen, wobei

- ▶ \mathcal{N} eine Menge von Knoten (**Nodes** bzw. **Vertices**) und
- ▶ $\mathcal{A} = \{(i,j) \mid i, j \in \mathcal{N} \wedge i, j \text{ sind gerichtet verbunden}\}$ eine Menge gerichteter Kanten (**Arcs**) bezeichnet.

Beispiel: $\mathcal{N} = \{1, 2, 3, 4\}$, $\mathcal{A} = \{(1,2), (2,3), (2,4), (1,4)\}$

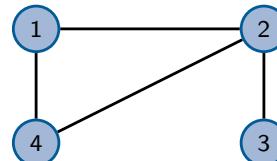


Ungerichtete Graphen

Ein **symmetrisches** Netzwerk lässt sich als **ungerichteter** Graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ darstellen, wobei

- ▶ \mathcal{N} eine Menge von Knoten und
- ▶ $\mathcal{E} = \{\{i,j\} \mid i, j \in \mathcal{N} \wedge i, j \text{ sind ungerichtet verbunden}\}$ eine Menge ungerichteter Kanten (**Edges**) bezeichnet.

Beispiel: $\mathcal{N} = \{1, 2, 3, 4\}$, $\mathcal{E} = \{\{1,2\}, \{2,3\}, \{2,4\}, \{1,4\}\}$



Hinweis zur Notation

Ungerichtete Graphen können als gerichtete Graphen mit sym. Kanten verstanden werden. Eine ungerichtete Kante $\{i,j\}$ eines ungerichteten Graphen mit Kantenkosten c_{ij} entspricht also den beiden gerichteten Kanten (i,j) und (j,i) eines gerichteten Graphen mit Kantenkosten $c_{ji} = c_{ij}$.



Pfade in Netzwerken

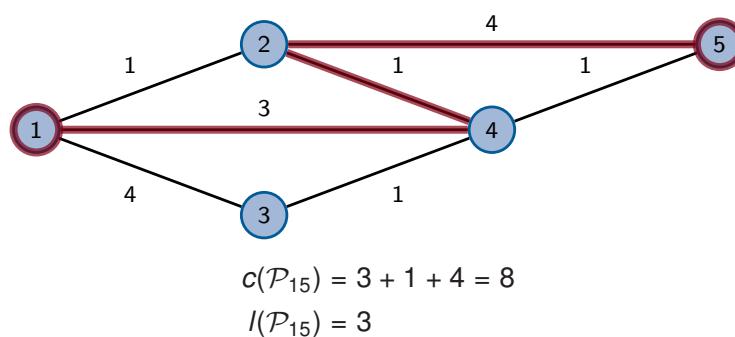
- ▶ Ein **Pfad** zwischen zwei Knoten¹ $s, t \in \mathcal{N}$ ist eine Menge

$$\mathcal{P}_{st} = \{(s,i),(i,j), \dots, (k,l),(l,t)\}$$

gerichteter Kanten, die s und t miteinander verbinden.

- ▶ Die **Pfadkosten** entsprechen der Summe der Kantenkosten: $c(\mathcal{P}_{st}) = \sum_{(i,j) \in \mathcal{P}_{st}} c_{ij}$.
- ▶ Die **Pfadlänge** entspricht der Anzahl der Kanten auf dem Pfad: $l(\mathcal{P}_{st}) = |\mathcal{P}_{st}|$. Die Pfadlänge wird auch **Hop Count** genannt.

Beispiel: $\mathcal{P}_{15} = \{(1,4),(4,2),(2,5)\}$



¹ Eine Nachrichtenquelle wird häufig mit s (engl. source) abgekürzt, eine Senke mit t (engl. terminal).

Netztopologien

Die **Topologie** beschreibt die Struktur, wie Knoten miteinander verbunden sind. Wir unterscheiden die

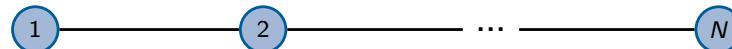
- ▶ **physikalische** Topologie und die
- ▶ **logische** Topologie.

Wichtige Topologien (Beispiele)

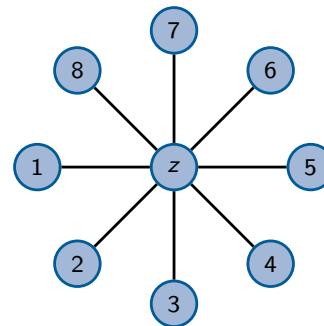
- ▶ **Punkt-zu-Punkt** (engl. **Point-to-Point**)



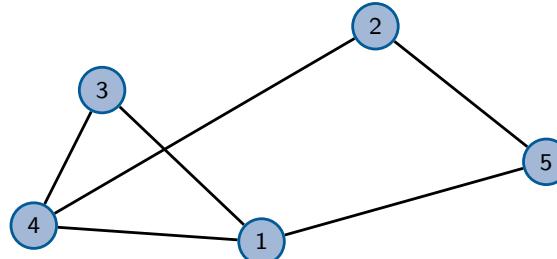
- ▶ **Kette**



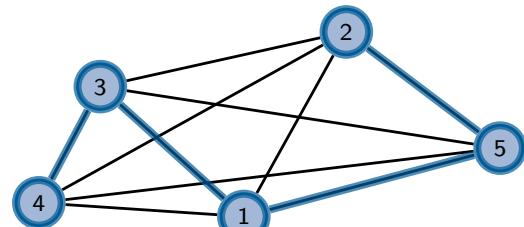
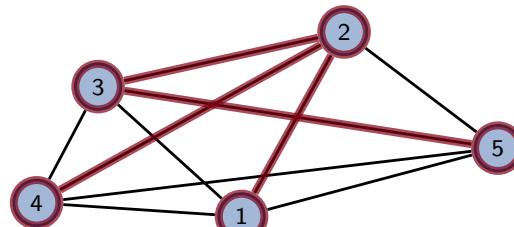
- ▶ **Stern**



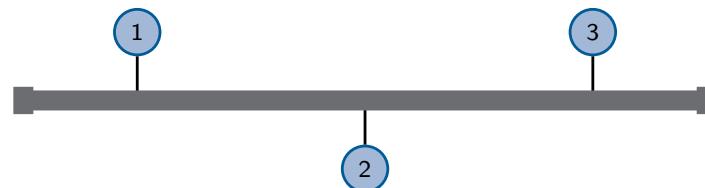
- ▶ **Vermischung** (engl. **Mesh**)



- ▶ **Baum** (meist logische Topologie)



- ▶ **Bus**



Adjazenzmatrix

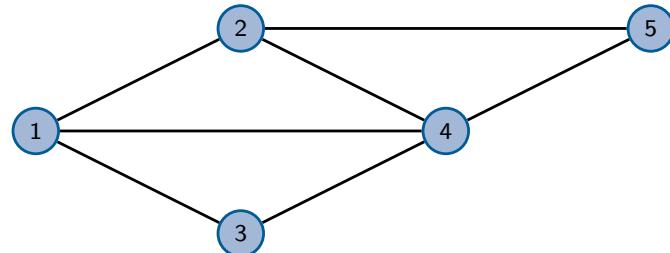
Netzwerke lassen sich leicht als Matrizen schreiben. Die **Adjazenzmatrix**

$$\mathbf{A} = (a_{ij}) = \begin{cases} 1 & \exists(i,j) \in \mathcal{A} \\ 0 & \text{sonst} \end{cases}, \quad \forall i,j \in \mathcal{N}, \quad \mathbf{A} \in \{0,1\}^{N \times N}$$

gibt an, ob Knoten i mit Knoten j verbunden ist.

Beispiel:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$



- ▶ Das Element a_{ij} der Matrix \mathbf{A} ist 1, wenn eine Verbindung von Knoten i zu Knoten j besteht.
- ▶ \mathbf{A} ist symmetrisch ($\mathbf{A} = \mathbf{A}^T$), wenn die Kanten ungerichtet sind, d. h. zu jeder Kante (i,j) auch eine antiparallele Kante (j,i) existiert.

Distanzmatrix

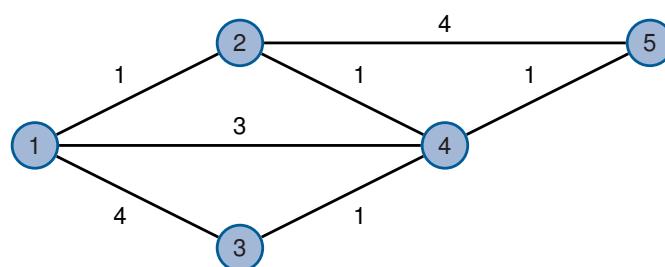
Die **Distanzmatrix**

$$\mathbf{D} = (d_{ij}) = \begin{cases} c_{ij} & \exists(i,j) \in \mathcal{A} \\ 0 & \text{wenn } i = j, \quad \forall i,j \in \mathcal{N}, \\ \infty & \text{sonst} \end{cases} \quad \mathbf{D} \in \mathbb{R}_{0+}^{N \times N}$$

enthält die Kosten der Pfade der Länge 1 zwischen allen Knotenpaaren.

Beispiel:

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 4 & 3 & \infty \\ 1 & 0 & \infty & 1 & 4 \\ 4 & \infty & 0 & 1 & \infty \\ 3 & 1 & 1 & 0 & 1 \\ \infty & 4 & \infty & 1 & 0 \end{bmatrix}$$



- ▶ Das Element d_{ij} der Matrix \mathbf{D} gibt die Distanz zwischen Knoten i und Knoten j an.
- ▶ Existiert keine direkte Verbindung zwischen i und j , so ist $d_{ij} = \infty$.
- ▶ \mathbf{D} ist symmetrisch, wenn das Netzwerk symmetrisch ist, d. h. zu jeder Kante (i,j) auch eine antiparallele Kante (j,i) mit denselben Kosten existiert.

Frage: Wie erhält man die Matrix, welche die Kosten eines kürzesten Pfads zwischen je zwei Knoten enthält?

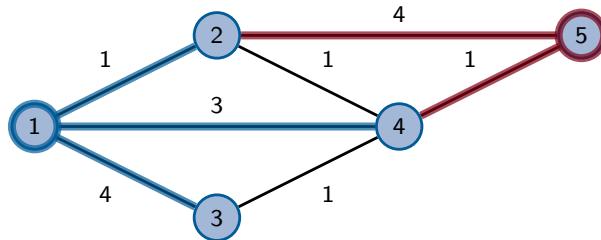
Antwort: Man potenziert \mathbf{D} bzgl. des min-plus-Produkts

$$\mathbf{D}^n = \mathbf{D}^{n-1} \otimes \mathbf{D} \text{ mit } d_{ij}^n = \min_{k \in \mathcal{N}} \{ d_{ik}^{n-1} + d_{kj} \}.$$

- ▶ Die Matrix \mathbf{D}^n enthält die Länge eines jeweils kürzesten Pfades über höchstens n Hops.
- ▶ Für ein endliches n konvergiert die Potenzreihe, so dass $\mathbf{D}^{n+1} = \mathbf{D}^n = \mathbf{D}^*$.

Beispiel: Wie entsteht Element (1,5) der Matrix \mathbf{D}^2 ?

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 4 & 3 & \infty \\ 1 & 0 & \infty & 1 & 4 \\ 4 & \infty & 0 & 1 & \infty \\ 3 & 1 & 1 & 0 & 1 \\ \infty & 4 & \infty & 1 & 0 \end{bmatrix}$$



- ▶ Zeile 1 gibt die Kosten eines jeweils kürzesten Pfades der Länge höchstens 1 von Knoten 1 zu allen anderen Knoten an,
- ▶ Spalte 5 gibt die Kosten an, mit denen Knoten 5 von allen anderen Knoten über einen kürzesten Pfad der Länge höchstens 1 erreicht werden kann.

Wie oft muss multipliziert werden?

- ▶ Der Wert n , so dass $\mathbf{D}^n = \mathbf{D}^{n+1} = \mathbf{D}^*$ gilt, ist durch den längsten einfachen Pfad im Netzwerk beschränkt.
- ▶ Der längste einfache Pfad ist durch die Anzahl N der Knoten beschränkt.

$$\Rightarrow n < N$$

Im vorherigen Beispiel reicht bereits $n = 3$ aus, obwohl $N = 5$ gilt.

Die Matrix \mathbf{D}^* enthält die Kosten eines jeweils kürzesten Pfades zwischen je zwei Knoten und löst damit das All-pair-shortest-distance-Problem (apsd).

Erzeugung von Baumstrukturen

Ein Baum ist ein **zusammenhängender** aber **schleifenfreier** Graph. Wir unterscheiden im folgenden zwei spezielle Arten von Bäumen:

- ▶ **Shortest Path Tree (SPT)**

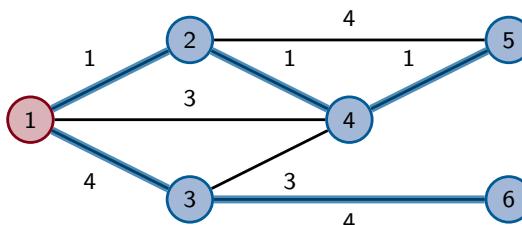
Verbindet einen Wurzelknoten mit jeweils minimalen Kosten mit jedem anderen Knoten des Netzwerks.

- ▶ **Minimum Spanning Tree (MST)**

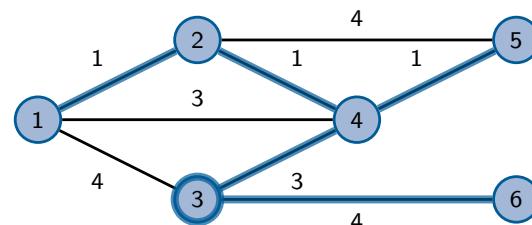
Verbindet alle Knoten des Netzwerks mit insgesamt minimalen Kosten.

Diese Bäume minimieren unterschiedliche Metriken und sind i. A. **nicht identisch**.

Beispiel:



(a) Shortest Path Tree (SPT) mit Wurzelknoten 1



(b) Minimum Spanning Tree (MST)

In Kapitel 3 werden wir zwei Algorithmen zur Erzeugung von SPTs kennen lernen / wiederholen:

- ▶ Algorithmus von Bellman-Ford (basiert auf dem min-plus-Produkt)
- ▶ Dijkstras-Algorithmus (Greedy-Prinzip)

Übersicht

Problemstellung und Motivation

Darstellung von Netzwerken als Graphen

Netztopologien
Adjazenz- und Distanzmatrix
Erzeugung von Baumstrukturen

Verbindungscharakterisierung, Mehrfachzugriff, Medienzugriffskontrolle

Verbindungscharakterisierung
Medienzugriff
ALOHA und Slotted ALOHA
CSMA, CSMA/CD, CSMA/CA
Token Passing

Rahmenbildung, Adressierung und Fehlererkennung

Erkennung von Rahmengrenzen und Codetransparenz
Adressierung und Fehlererkennung

Verbindung auf Schicht 1 und 2

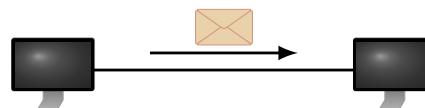
Hubs, Bridges und Switches

Verbindungscharakterisierung

Eine Verbindung zwischen zwei Knoten kann hinsichtlich einiger grundlegender Eigenschaften charakterisiert werden:

- ▶ Übertragungsrate
- ▶ Übertragungsverzögerung
- ▶ Übertragungsrichtung
- ▶ Mehrfachzugriff (Multiplexing)

Zunächst betrachten wir eine **Punkt-zu-Punkt-Verbindung**:



Übertragungsrate

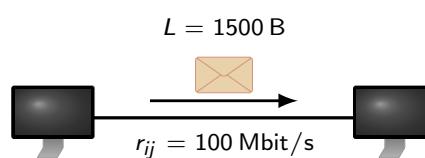
Definition (Übertragungsrate und Serialisierungszeit)

Die **Übertragungsrate** r in bit/s bestimmt die notwendige Zeit, um L Datenbits auf ein Übertragungsmedium zu legen. Sie bedingt die **Serialisierungszeit**

$$t_s = \frac{L}{r}.$$

Die Serialisierungszeit bzw. Übertragungsverzögerung wird im Englischen als **Serialization Delay** bzw. **Transmission Delay** bezeichnet (vgl. t_s).

Beispiel:



$$t_s = \frac{L}{r} = \frac{1500 \cdot 8 \text{ bit}}{100 \cdot 10^6 \text{ bit/s}} = 120 \mu\text{s}$$

Frage: Wann empfängt Knoten j das **erste Bit** der Nachricht?

Ausbreitungsgeschwindigkeit

In Kapitel 1 haben wir bereits gesehen, dass Signale i. d. R. elektromagnetische Wellen sind, welche sich mit Lichtgeschwindigkeit im Medium ausbreiten.

Definition (Ausbreitungsverzögerung)

Die **Ausbreitungsverzögerung** über eine Distanz d röhrt von der endlichen Ausbreitungsgeschwindigkeit von Signalen, welche relativ zur Lichtgeschwindigkeit im Vakuum $c \approx 300\,000\text{ km/s}$ angegeben wird:

$$t_p = \frac{d}{\nu c}.$$

Der Wert $0 < \nu < 1$ ist die relative Ausbreitungsgeschwindigkeit in einem Medium. Für Kupfer gilt beispielsweise $\nu \approx 2/3$. Die Ausbreitungsverzögerung wird im Englischen als **Propagation Delay** bezeichnet (vgl. Benennung t_p).

Beispiel:

- ▶ Im Beispiel auf der vorherigen Folie haben wir exemplarisch die Serialisierungszeit zu $t_s = 120\text{ }\mu\text{s}$ bestimmt
- ▶ Angenommen die Knoten i und j sind $d_{ij} = 100\text{ m}$ voneinander entfernt
- ▶ Bei Lichtgeschwindigkeit benötigen Signale für diese Strecke gerade einmal 334 ns
⇒ j empfängt bereits das erste Bit der Nachricht, wenn i gerade das 33ste Bit sendet!

Frage: Wie lange dauert es, bis j das **letzte Bit** der Nachricht empfangen hat?

Übertragungszeit und Nachrichtenflussdiagramm

In einem **Nachrichtenflussdiagramm** bzw. **Weg-Zeit-Diagramm** lässt sich die zeitliche Abfolge beim Senden und Empfangen von Nachrichten grafisch veranschaulichen:

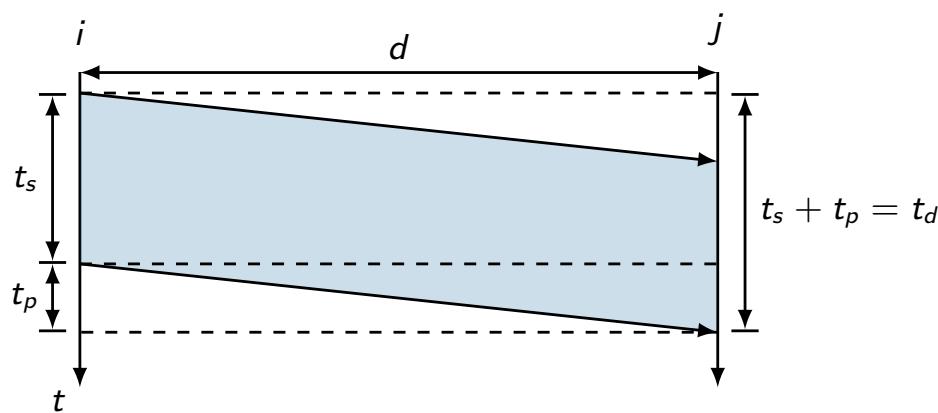


Abbildung: Nachrichtenflussdiagramm

- ▶ Die Gesamtverzögerung t_d (Delay) ergibt sich daher zu $t_d = t_s + t_p = \frac{L}{r} + \frac{d}{\nu c}$.
- ▶ Die Ausbreitungsverzögerung kann bei der Bestimmung von t_d u. U. vernachlässigt werden. Dies hängt allerdings von r , L und d ab! (s. Übung)

Bandbreitenverzögerungsprodukt

Durch die endliche Ausbreitungsverzögerung besitzt ein Übertragungskanal eine gewisse „Speicherkapazität“ C , welche als **Bandbreitenverzögerungsprodukt** bekannt ist.

Definition (Bandbreitenverzögerungsprodukt)

Als Bandbreitenverzögerungsprodukt bezeichnet man die Anzahl an Bits (Kapazität)

$$C = t_p r = \frac{d}{\nu c} r,$$

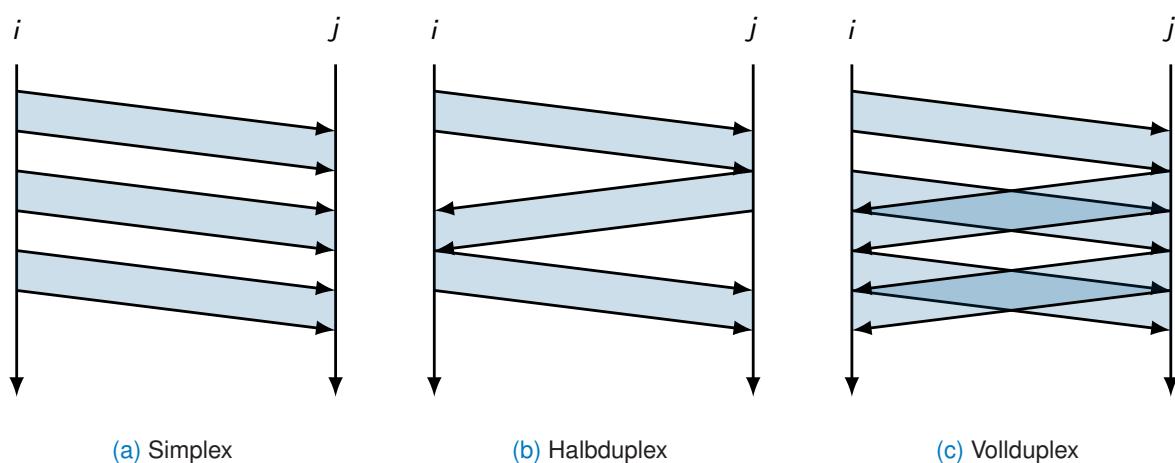
die sich in einer Senderichtung gleichzeitig auf der Leitung befinden können.

Beispiel:

- ▶ Leitung mit $r = 1 \text{ Gbit/s}$
- ▶ Länge $d = 10 \text{ m}$
- ▶ $\nu = 2/3$ (Kupferleitung)
- ▶ $C = t_p \cdot r = \frac{d}{\nu c} \cdot r = \frac{10 \text{ m}}{2 \cdot 10^8 \text{ m/s}} \cdot 10^9 \frac{\text{bit}}{\text{s}} \approx 50 \text{ bit}$

Übertragungsrichtung

Hinsichtlich der Übertragungsrichtung unterscheidet man:



Die Art der Verbindung hängt dabei ab von

- ▶ den Fähigkeiten des Übertragungskanals,
- ▶ dem Medienzugriffsverfahren und
- ▶ den Anforderungen der Kommunikationspartner.

Mehrfachzugriff (Multiplexing)

Häufig ist es von Vorteil, Nachrichten unterschiedlicher Teilnehmer gemeinsam über eine Leitung zu übertragen:

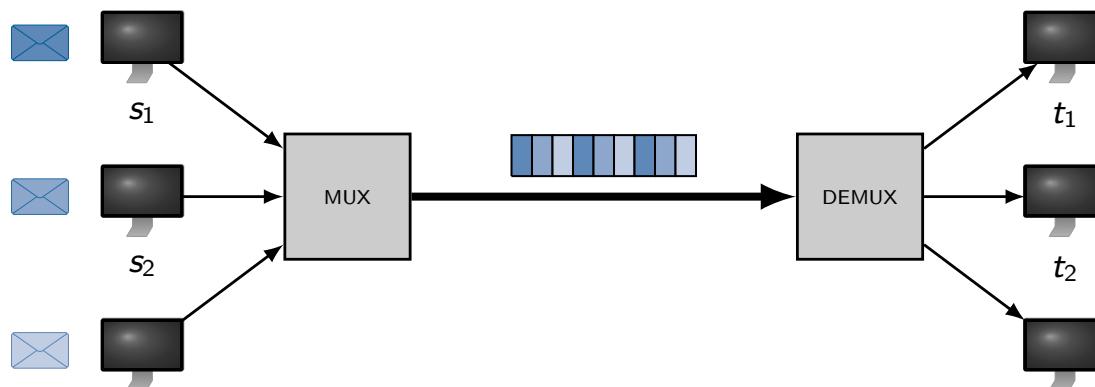


Abbildung: Deterministisches Zeitmultiplex-Verfahren

Ein anderes Zeitmultiplex-Verfahren haben wir bereits kennen gelernt:

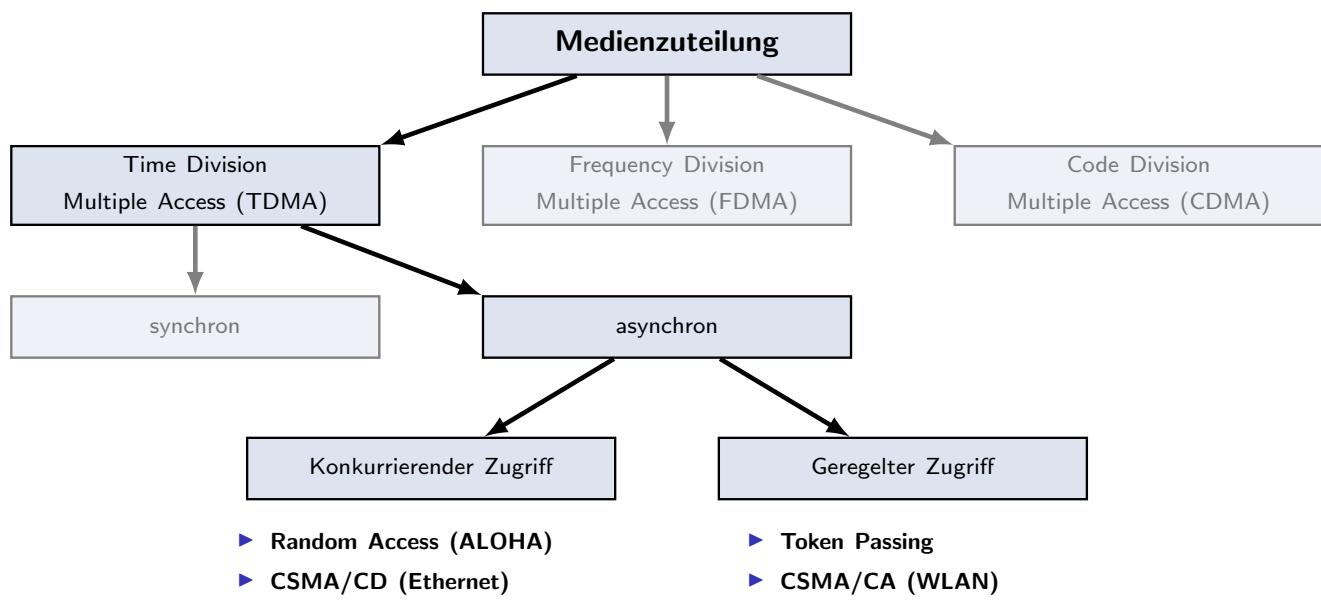
- ▶ Werden mehrere Computer mittels eines **Hubs** miteinander verbunden,
- ▶ so bildet das Hub ein **gemeinsames geteiltes Medium**,
- ▶ auf das die Computer mittels eines **nicht-deterministischen Medienzugriffsverfahrens abwechselnd zugreifen**.

Übersicht über Multiplex-Verfahren

- ▶ **Zeitmultiplex (Time Division Multiplex, TDM)**
(s. vorherige Folie)
 - ▶ Deterministische Verfahren z. B. im Telefonnetz, bei ISDN-Verbindungen und im Mobilfunk
 - ▶ Nichtdeterministische Verfahren (konkurrierender Zugriff) in paketbasierten Netzwerken (z. B. Ethernet, WLAN)
- ▶ **Frequenzmultiplex (Frequency Division Multiplex, FDM)**
Aufteilung des Kanals in unterschiedliche Frequenzbänder (spektrale Zerlegung) und Zuweisung Frequenzbänder an Kommunikationspartner (s. Kapitel 1).
 - ▶ Omnipräsent bei Funkübertragungen (z. B. unterschiedliche Radiosender)
 - ▶ Einsatz bei Glasfaserübertragungen („Modes“ mit unterschiedlicher Farbe)
 - ▶ Koexistenz von ISDN und DSL auf derselben Leitung
- ▶ **Raummultiplex (Space Division Multiplex, SDM)**
Verwendung mehrerer paralleler Übertragungskanäle.
 - ▶ „Kanalbündelung“ bei ISDN
 - ▶ **MIMO (Multiple-In Multiple-Out)** bei kabellosen Übertragungen (Verwendung mehrerer Antennen schafft mehrere Übertragungskanäle)
- ▶ **Codemultiplex (Code Division Multiplex, CDM)**
Verwendung orthogonaler Alphabete und Zuweisung der Alphabete an Kommunikationspartner.
 - ▶ Die Mobilfunktechnologie UMTS repräsentiert eine Variante von CDMA
 - ▶ Eine weitere Variante von CDMA im Mobilfunkbereich, CDMA2000, findet sich u.a. im Netz des amerikanischen Providers Verizon (c.f. „CDMA-iPhone“)

Mehrfachzugriff und Medienzugriffskontrolle [2]

Einige der (statischen) Multiplexing-Verfahren eignen sich auch als **Mehrfachzugriffsverfahren**:



Diese ausgewählten vier **Zugriffsverfahren** werden wir im Folgenden näher kennenlernen.

Bewertungskriterien für Medienzugriffsverfahren sind unter anderem:

- ▶ Durchsatz, d. h. Gesamtanzahl an Nachrichten pro Zeiteinheit, die übertragen werden können
- ▶ Verzögerung für einzelne Nachrichten
- ▶ Fairness zwischen Teilnehmern, die sich dasselbe Medium teilen
- ▶ Implementierungsaufwand für Sender und Empfänger

Problem bei synchronem TDMA

- ▶ Der Kanal wird statisch zwischen Teilnehmern aufgeteilt
- ▶ Datenverkehr ist aber **stossartig** bzw. **burst-artig**, d. h. ein Teilnehmer überträgt kurz mit hoher Bandbreite und danach längere Zeit nicht mehr
- ▶ Bandbreite steht während Ruhepausen anderen Teilnehmern nicht zur Verfügung

Lösungsidee: Asynchrones (flexibles) TDMA

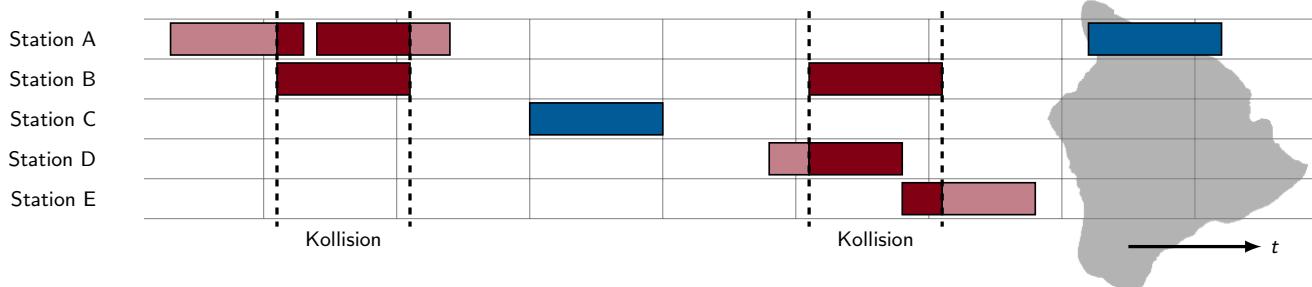
- ▶ Keine statische Aufteilung / Zuweisung von Zeitslots
- ▶ Stattdessen: **Zufälliger, konkurrierender** oder **dynamisch geregelter** Medienzugriff

Random Access (ALOHA)

- ▶ Entwickelt an der Universität von Hawaii (1971), c.f. Prof. Abramson
- ▶ Ursprünglich für kabellose Datenübertragungen
- ▶ Ziel: Verbindung von Oahu mit den anderen hawaiianischen Inseln

Funktionsweise

- ▶ Jede Station sendet an eine zentrale Station (vgl. „Basisstation“ in modernen WLANs), sobald Daten vorliegen
- ▶ Senden zwei Stationen gleichzeitig, kommt es zu Kollisionen
- ▶ Erfolgreich übertragene Nachrichten werden vom Empfänger auf anderer Frequenz quittiert („out-of-band“ Bestätigungsverfahren auf Link-Layer, keine Kollisionen zwischen Nachrichten und Bestätigungen)



Das Kanalmodell ist vergleichsweise einfach. Es existieren math. Beschreibungen für den sog. **ALOHA Random Access Channel**.

Erreichbarer Durchsatz mit ALOHA

Vereinfachende Annahmen:

- ▶ Mittlere bis beliebig große Anzahl an Knoten ($N > 15$)
- ▶ Gleiche, unabhängige und geringe Sendewahrscheinlichkeit auf allen Knoten
- ▶ Nachrichten konstanter Größe (Sendedauer T)

Modellierung:

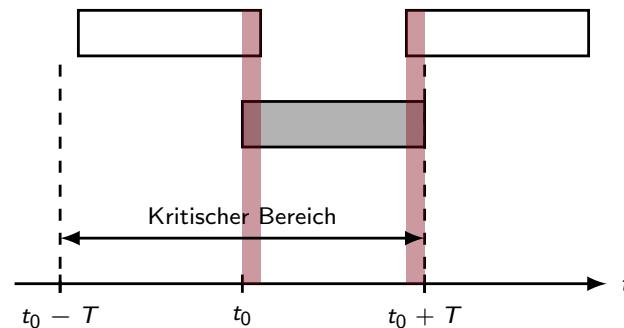
- ▶ Ob ein bestimmter Knoten i innerhalb des Zeitintervalls $[t, t + T]$ zu senden beginnt oder nicht entspricht einem Bernoulli-Experiment mit Erfolgs- bzw. Sendewahrscheinlichkeit p_i
- ▶ Da die Sendewahrscheinlichkeit für alle Knoten gleich ist, gilt $p_i = p \quad \forall i = 1, \dots, N$
- ▶ Da wir N Knoten haben, die jeweils unabhängig voneinander zu senden beginnen, wird dasselbe Bernoulli-Experiment N -mal wiederholt
- ▶ Das ist nichts anderes als eine **Binomialverteilung**, welche die Anzahl der Erfolge einer Serie gleichartiger und unabhängiger Versuche beschreibt
- ▶ Für sinnvoll großes N kann die Binomialverteilung durch eine **Poisson-Verteilung**² approximiert werden (s. Übung)
- ▶ Die mittlere erwartete Anzahl von Nachrichten pro Intervall ist gegeben als $Np = \lambda$

Das Ereignis X_t , dass im Intervall $[t, t + T]$ genau k Knoten senden, ist poisson-verteilt:

$$\Pr[X_t = k] = \frac{\lambda^k e^{-\lambda}}{k!}.$$

² Verteilung der seltenen Ereignisse

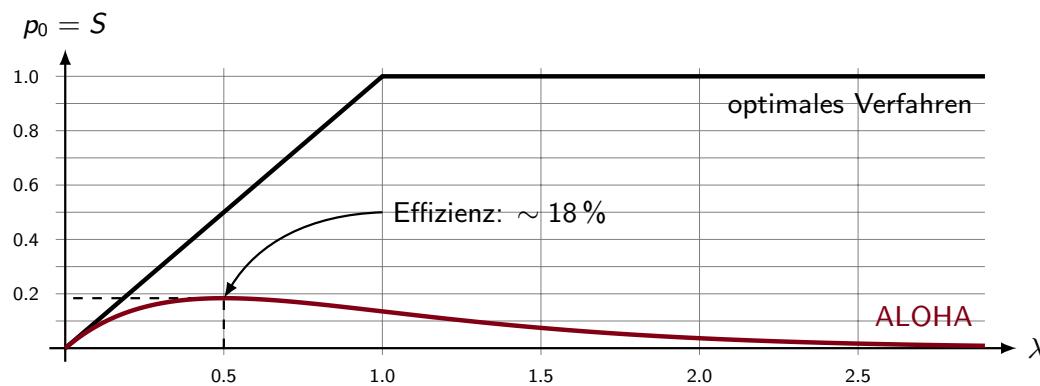
- ▶ Eine beliebige Station sende nun zum Zeitpunkt t_0 eine Nachricht
- ▶ Eine Kollision tritt genau dann auf, wenn eine andere Station im Intervall $(t_0 - T, t_0 + T]$ versucht, ebenfalls zu übertragen
- ▶ Die Übertragung ist also erfolgreich, wenn innerhalb des Intervalls $[t_0, t_0 + T]$ genau eine Übertragung stattfindet **und** im Intervall $(t_0 - T, t_0)$ keine Übertragung begonnen hat.



- ▶ Mit der Dichtefunktion $\Pr[X_t = k] = \frac{\lambda^k e^{-\lambda}}{k!}$ erhalten wir
- ▶ die Wahrscheinlichkeit p_0 für eine erfolgreiche Übertragung:

$$p_0 = \Pr[X_{t_0 - T} = 0] \cdot \Pr[X_{t_0} = 1] = e^{-\lambda} \cdot \lambda e^{-\lambda} = \lambda e^{-2\lambda}$$

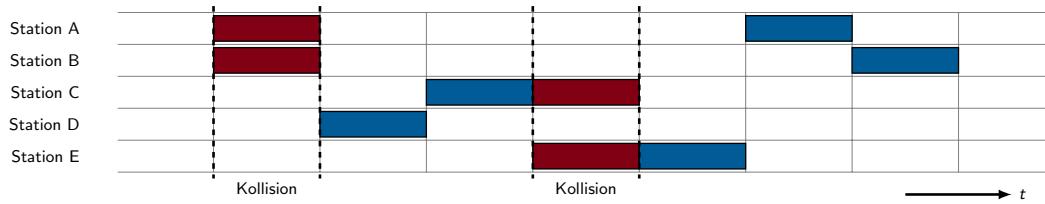
Die Erfolgswahrscheinlichkeit p_0 kann gegen die Senderate λ aufgetragen werden:



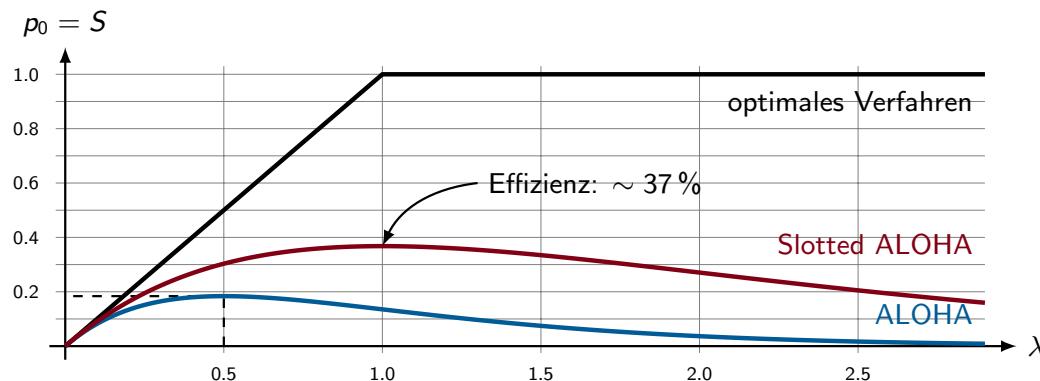
- ▶ Wir wissen, dass innerhalb eines beliebigen Intervalls $[t, t + T]$ höchstens eine Übertragung erfolgreich sein kann
- ▶ Dementsprechend entspricht die Anzahl S der erfolgreichen Nachrichten pro Intervall gleichzeitig der Wahrscheinlichkeit für eine erfolgreiche Übertragung
- ▶ Bei einem **optimalen** Verfahren würde die Anzahl erfolgreicher Nachrichten S linear mit der Senderate ansteigen, bis die maximale Anzahl von Nachrichten pro Zeitintervall erreicht ist (hier ist das genau eine Nachricht pro Intervall)
- ▶ Steigt die Senderate weiter, würde dies ein optimales Verfahren nicht beeinträchtigen

Variante: Slotted ALOHA

Stationen dürfen nicht mehr zu beliebigen Zeitpunkten mit einer Übertragung beginnen, sondern nur noch zu den Zeitpunkten $t = nT$, $n = 0, 1, \dots$



Kritischer Bereich ist nur noch T anstelle von $2T$ $\Rightarrow S = \lambda \cdot e^{-\lambda}$.



Carrier Sense Multiple Access (CSMA)

Eine einfache Verbesserung von Slotted ALOHA: „Listen Before Talk“

- ▶ Höre das Medium ab
- ▶ Beginne erst dann zu senden, wenn das Medium frei ist

Non-persistent CSMA:

1. Wenn Medium frei, übertrage im nächstmöglichen Intervall
2. Wenn belegt, warte eine feste Zeitspanne, dann (1)

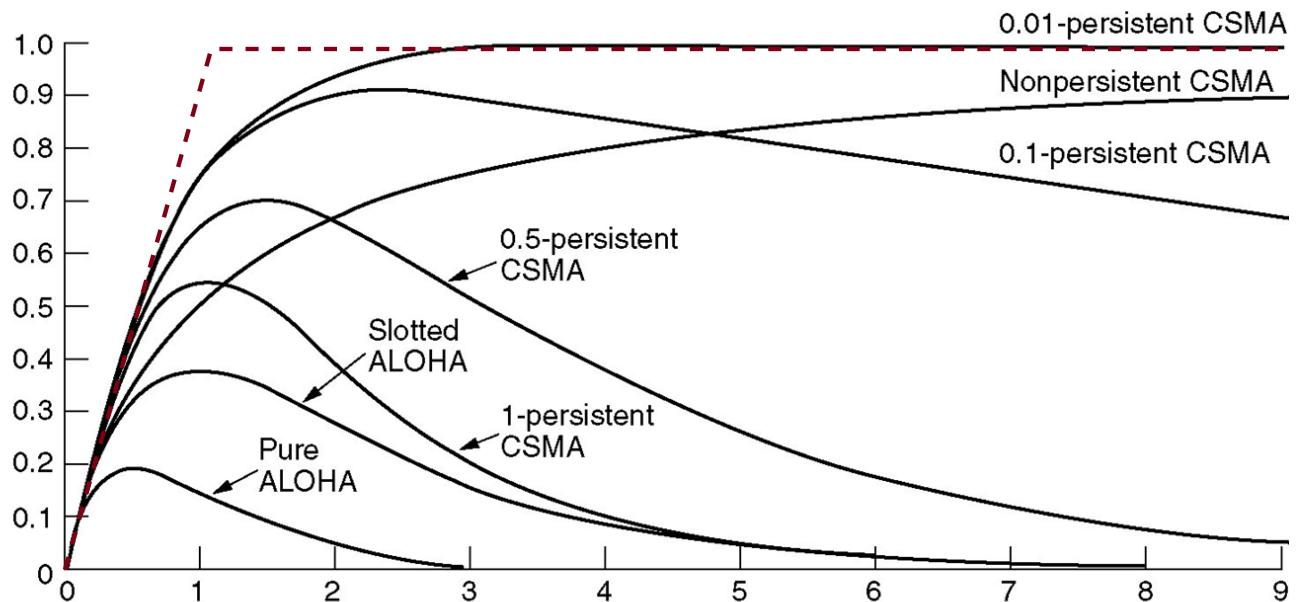
1-persistent CSMA:

1. Wenn Medium frei, übertrage im nächstmöglichen Intervall
2. Wenn Medium belegt, warte bis frei und übertrage im nächstmöglichen Intervall

p-persistent CSMA:

1. Wenn Medium frei, übertrage mit Wahrscheinlichkeit p oder verzögere mit Wahrscheinlichkeit $1 - p$ um eine Slotzeit
2. Wenn Medium belegt, warte bis frei, dann (1)
3. Wenn um eine Slotzeit verzögert, dann (1)

Alle bisherigen Verfahren im Vergleich



CSMA/CD (Collision Detection)

Ansatz

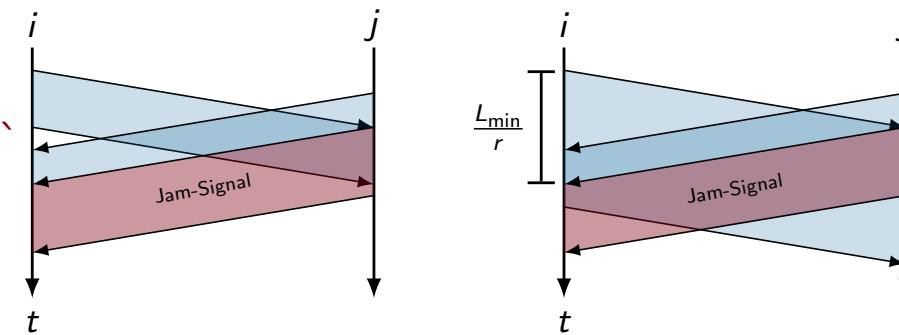
- ▶ Erkenne Kollisionen und wiederhole die Übertragung, wenn eine Kollision erkannt wird
- ▶ Verzichte auf das Senden von Bestätigungen
- ▶ Wird keine Kollision erkannt, gilt die Übertragung als erfolgreich

Problem: Der Sender muss die Kollision erkennen, während er noch überträgt

Voraussetzung für CSMA/CD [2]

Angenommen zwei Stationen i und j kommunizieren über eine Distanz d mittels CSMA/CD. Damit Kollisionen erkannt werden können, müssen Nachrichten folgende Mindestlänge L_{\min} aufweisen:

$$L_{\min} = \frac{2d}{\nu c} r$$



Wird 1-persistentes CSMA mit Kollisionserkennung verwendet, ergibt sich folgendes Problem:

- ▶ Die Kollision zerstört die Nachrichten beider in die Kollision verwickelten Stationen
- ▶ Mind. eine der Stationen sendet ein JAM-Signal
- ▶ Nachdem das Medium frei wird, wiederholen beide Stationen die Übertragung
⇒ Es kommt sofort wieder zu einer Kollision

Lösung: Warte „zufällige“ Zeit nach einer Kollision

Binary Exponential Backoff

Bei der k -ten Wiederholung einer Nachricht

- ▶ wählt der Sender zufällig $n \in \{0, 1, \dots, \min\{2^{k-1}, 1024\}\}$ aus und
- ▶ wartet n Slotzeiten vor einem erneuten Sendeversuch.

Die maximale Wiederholungszahl beträgt $k = 15$ (also 16 Sendeversuche).

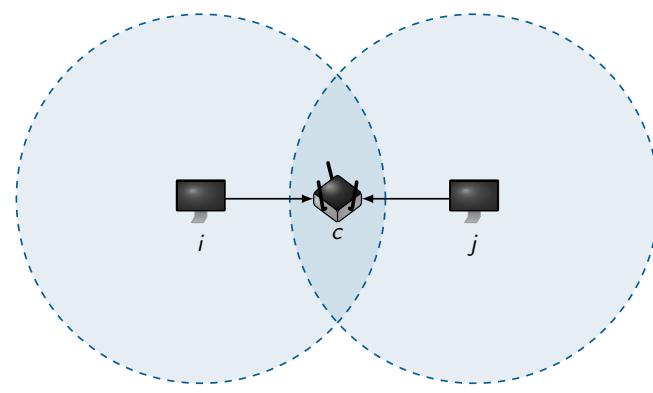
Durch die Wartezeiten, die

- ▶ zufällig gewählt und
- ▶ situationsabhängig größer werden,
- ▶ wird die Kollisionswahrscheinlichkeit bei Wiederholungen reduziert.

CSMA/CA (Collision Avoidance)

In Funknetzwerken funktioniert CSMA/CD nicht, da der Sender einer Nachricht eine Kollision auch bei ausreichender Nachrichtenlänge nicht immer detektieren kann.

„Hidden Station“:



- ▶ Knoten i und j senden gleichzeitig
- ▶ Knoten c erkennt die Kollision
- ▶ Weder i noch j bemerken die Kollision

CSMA/CA basiert auf p -persistentem CSMA, d. h.

1. Wenn Medium frei, übertrage mit Wahrscheinlichkeit p oder verzögere mit Wahrscheinlichkeit $1 - p$ um eine Slotzeit
2. Wenn Medium belegt, warte bis frei, dann (1)
3. Wenn um eine Slotzeit verzögert, dann (1)

Fallbeispiel: IEEE 802.11 DCF (Distributed Coordination Function)

- ▶ Festes Zeitintervall zwischen Rahmen (DCF Interframe Spacing).
- ▶ Wenn Medium mind. für DIFS idle ist, dann wähle unabhängig und gleichverteilt eine Anzahl von Backoff-Slots aus dem Intervall $\{0, 1, 2, \dots, \min\{2^{c+n} - 1, 255\}\}$.
- ▶ c ist abhängig vom PHY (z.B. $c = 4$), n ist der Retry Counter des Binary Exponential Backoffs.

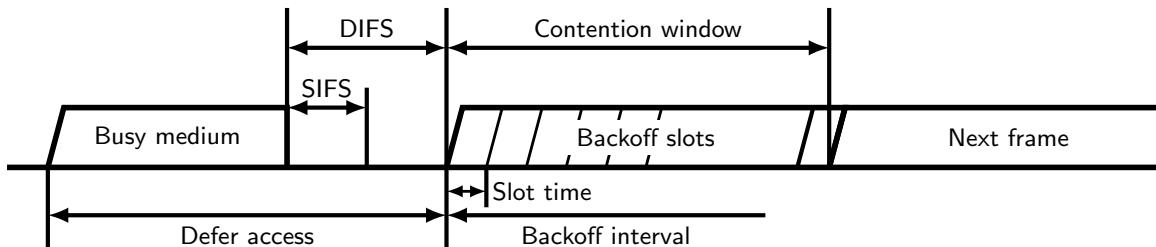


Abbildung: IEEE 802.11 DCF

- ▶ Medienzugriff hat durch festes c stets ein Contention Window.
- ▶ Ein Rahmen gilt in IEEE 802.11 als erfolgreich übertragen, wenn
 - ▶ im Fall von Unicasts der Empfänger eine Bestätigung schickt (Link-Layer Acknowledgements) oder
 - ▶ im Fall von Broadcasts die Übertragung eines Frames störungsfrei abgeschlossen wird.
- ▶ Da i.d.R. nicht gleichzeitig gesendet und das Medium geprüft werden kann (anders bei Ethernet), ist die zweite Bedingung praktisch bereits erfüllt, wenn ein Knoten zu senden beginnt.

Fallbeispiel: IEEE 802.11 DCF (Distributed Coordination Function)

Was passiert in der Praxis? Beispiel anhand handelsüblicher Hardware im Monitor Mode³:

- ▶ Wir deaktivieren Link-Layer Bestätigungen und prüfen, wie sich die Hardware verhält.
- ▶ Ohne Bestätigungen wird es keinen Exponential Back-Off geben, da Übertragungen (einmal begonnen) nicht mehr fehlschlagen (IEEE 802.11 macht kein Media Sensing während einer Übertragung läuft).
- ▶ Das Contention Window sollte aber $\{0, 1, 2, \dots, 15\}$ betragen und Backoff-Slots unabhängig und gleichverteilt daraus gezogen werden. \Rightarrow Ein zu sendendes Frame sollte (bei freiem Medium) im Mittel um 7.5 Slotzeiten verzögert werden.

Was wir nun genau tun:

- ▶ Wir können die Verzögerung zwischen aufeinander folgenden Frames einer Station mittels einer zweiten Station relativ genau messen.
- ▶ Aus den gemessenen Zeiten erstellen wir eine (empirische) kummulative Verteilungsfunktion (KVF).
- ▶ Diese EKVF gibt $\Pr[X \leq N]$ an, also die Wahrscheinlichkeit, dass die Anzahl der gewarteten Slotzeiten (die Größe des Contention Windows) kleiner oder gleich N Slotzeiten ist.
- ▶ Diese sollte einer Treppenfunktion zwischen 0 und 15 mit äquidistanten Inkrementen um jeweils 1 Slotzeit folgen.

³ Monitor Mode bezeichnet einen Operationsmodus von IEEE 802.11 Hardware, in dem die Netzwerkkarten alle eingehenden Frames vollständig unverarbeitet zugänglich machen, unabhängig davon, ob es sich um Daten-, Management- oder Control-Frames handelt und unabhängig davon, ob das Frame überhaupt an die jeweilige Station adressiert war. Umgekehrt können in diesem Modus auch beliebige Link-Layer Frames „von Hand gebaut“ und unverändert verschickt werden.

Fallbeispiel: IEEE 802.11 DCF (Distributed Coordination Function)

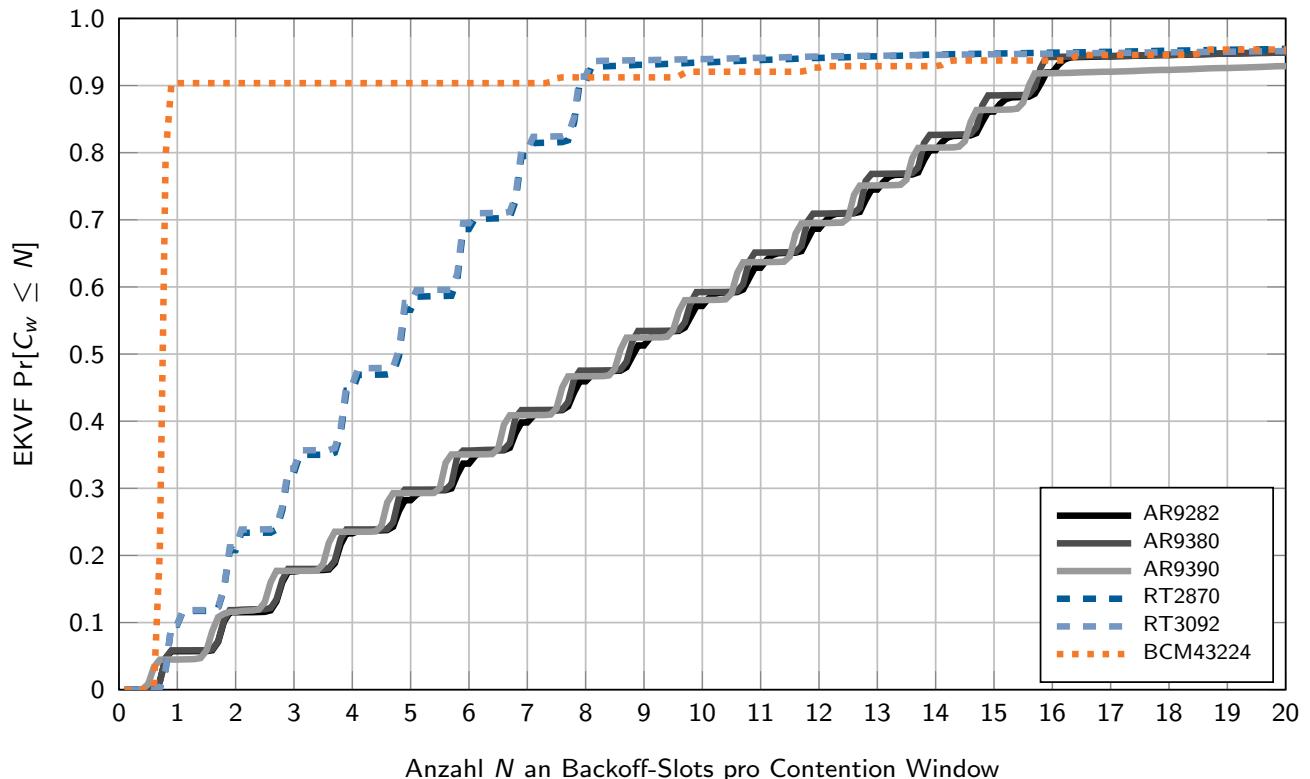


Abbildung: EKVF der zeitl. Abstände zwischen Frames einer Station gemessen in Vielfachen von Slotzeiten.

Fallbeispiel: IEEE 802.11 DCF (Distributed Coordination Function)

Was bedeutet das nun?

- ▶ Während eine dieser Broadcom-Karten sendet, haben alle anderen Sendepause.
- ▶ Hält sich ein Gerät nicht an die Vorgaben, kann es sich beim Senden auf Kosten anderer „Vorteile“ verschaffen, da kleinere Contention Phases
 - ▶ die Wahrscheinlichkeit erhöhen, die Contention Phase zu gewinnen und
 - ▶ natürlich die Idle-Time des Medium reduzieren.
- ▶ Gerade Letzteres (Idle-Times) ist bei IEEE 802.11 der limitierende Faktor, da die Zeit zwischen Frames im Verhältnis zur Serialisierungszeit sehr hoch ist.

Anmerkungen:

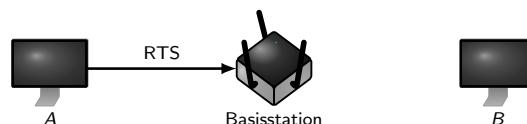
- ▶ Das Beispiel diskutiert IEEE 802.11 Hardware im Monitor Mode und ist daher für den (praxisrelevanten) Infrastructure Mode nur wenig aussagekräftig.
- ▶ In einer Masterarbeit haben wir aber kürzlich gezeigt, dass es dort aber nicht viel besser aussieht...
- ▶ Das Verhalten hängt nicht nur von der Hardware/Firmware, sondern auch vom Treiber ab.
- ▶ Nein, wir werden nicht von Atheros/Qualcomm bezahlt. :)
- ▶ Das dazu passende Paper gibts in Moodle.

Erweiterung: RTS/CTS (Request to Send / Clear to Send)

- ▶ Übertragungen werden i.d.R. von einer Basisstation gesteuert
- ▶ Bevor ein Knoten eine Nachricht überträgt, wird nach dem CSMA/CA-Prinzip ein RTS an die Basisstation geschickt
- ▶ Nur wenn die Basisstation mit einem CTS antwortet, darf die Übertragung beginnen

Beispiel:

1. A sendet RTS, welches von B aufgrund der Distanz nicht empfangen wird.



2. Basestation antwortet mit CTS, welches von A und B empfangen wird.



3. A darf senden, B muss eine im CTS definierte Zeitspanne abwarten, bevor überhaupt ein RTS gesendet werden darf.



Erweiterung: RTS/CTS (Request to Send / Clear to Send)

Vorteile:

- ▶ Kollisionen mit Hidden Stations werden vermieden, aber nicht gänzlich verhindert.
- ▶ Insgesamt weniger Kollisionen, auch ohne Hidden Stations.

Nachteile:

- ▶ Es können noch immer Kollisionen auftreten, z.B. wenn B das CTS nicht empfängt.
- ▶ RTS/CTS nimmt vorab Zeit in Anspruch, was die maximal erzielbare Datenrate reduziert.

Anmerkungen:

- ▶ RTS/CTS ist Bestandteil des sog. **Virtual Carrier Sensing**, da mit dem CTS das Medium für eine bestimmte Zeitspanne für eine Übertragung reserviert wird.
- ▶ Um die Verlustwahrscheinlichkeit von RTS/CTS-Nachrichten zu minimieren, werden diese mit der robustesten Kodierung übertragen, was i.d.R. der niedrigsten unterstützten Datenrate entspricht. Im Gegenzug sind RTS/CTS-Nachrichten sehr klein.
- ▶ Es ist streng genommen für RTS/CTS nicht notwendig, dass ein Netzwerk durch eine Basisstation kontrolliert wird. Es funktioniert auch im **ad-hoc Modus**⁴ oder (mit Einschränkungen) in Mesh-Netzwerken.
- ▶ Alle Geräte, unabhängig davon ob sie zum selben Service Set⁵ gehören oder nicht, sollten CTS-Nachrichten verarbeiten.

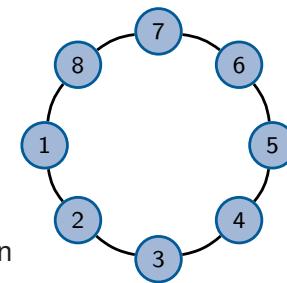
⁴ Bezeichnet eine Gruppe IEEE 802.11-fähiger Geräte, welche ohne Basisstation direkt miteinander kommunizieren

⁵ Bezeichnung für eine Gruppe miteinander kommunizierender IEEE 802.11-fähiger Geräte

Token Passing

Idee: Kollisionsfreie Übertragung durch Weitergabe eines **Tokens**

- ▶ Stationen werden zu einem physikalischen Ring zusammengeschaltet
- ▶ Ein Token zirkuliert im Ring
- ▶ Will eine Station senden, nimmt sie das Token vom Ring und darf danach als einzige Station im Ring übertragen
- ▶ Nachdem alle Nachrichten gesendet wurden (oder nach einer definierten Zeitspanne), wird das Token wieder auf den Ring gelegt



Empfang von Nachrichten:

- ▶ Die Nachricht zirkuliert wie das Token durch den Ring
- ▶ Der Empfänger markiert die Nachricht als gelesen und schickt sie weiter
- ▶ Trifft sie wieder beim Sender ein, so nimmt dieser sie vom Netz

Was ist, wenn das Token „verloren geht“?

- ▶ Es gibt eine **Monitor-Station**, z. B. die Station, die das erste Token erzeugt hat
- ▶ Diese Monitor-Station erzeugt bei Bedarf neue Tokens, entfernt endlos kreisende Pakete und entfernt doppelte Token
- ▶ Fällt die Monitor-Station aus, wird von den verbleibenden Stationen eine Neue gewählt

Vorteile:

- ▶ Sehr effizient, da keine kollisionsbedingten Wiederholungen
- ▶ Garantierte maximale Verzögerung (Determinismus)

Nachteile bzw. Schwierigkeiten:

- ▶ Geht das Token verloren, muss es durch ein Neues ersetzt werden
→ eine Station muss spezielle Aufgaben übernehmen (**Monitor-Station**).
- ▶ Fehlerhaftes Verhalten eines Knotens stört die gesamte Kommunikation im Ring.
- ▶ Übertragungsverzögerung u. U. größer als bei CSMA, da Sender auf Token warten muss.
- ▶ Zusammenschaltung der Stationen zu einem Ring ist u. U. aufwendig.

Einsatz heute:

- ▶ **Token Ring (IEEE 802.5)** wurde vollständig von Ethernet (IEEE 802.3) ersetzt und spielt in lokalen Netzwerken heute keine Rolle mehr.
- ▶ **FDDI (Fiber Distributed Data Interface)** ist ein Sammelbegriff für Glasfaserringe bis zu einer Länge von einigen hundert Kilometern. Diese werden z. B. als Backbone lokaler Zugangsanbieter im städtischen Maßstab eingesetzt.

Zusammenfassung

In diesem Teilkapitel haben wir einige **flexible** Zeitmultiplexverfahren kennengelernt, die Zugriff mehrerer Stationen auf ein gemeinsames Medium erlauben. Im Gegensatz zu **statischem** Zeitmultiplex wird die Kanalbandbreite nicht für inaktive Knoten reserviert.

Konkurrierender Zugriff:

- ▶ ALOHA und Slotted ALOHA
- ▶ CSMA (non-persistent, 1-persistent, p -persistent)
- ▶ CSMA/CD (Kollisionserkennung)
[IEEE 802.3 Ethernet](#)

Geregelter Zugriff:

- ▶ CSMA/CA (Kollisionsvermeidung)
[IEEE 802.11 WLAN](#)
- ▶ Token Passing (Kollisionsverhinderung)
[IEEE 802.5 Token Ring, Fiber Distributed Data Interface \(FDDI\)](#)

Übersicht

Problemstellung und Motivation

Darstellung von Netzwerken als Graphen

- Netztopologien
- Adjazenz- und Distanzmatrix
- Erzeugung von Baumstrukturen

Verbindungscharakterisierung, Mehrfachzugriff, Medienzugriffskontrolle

- Verbindungscharakterisierung
- Medienzugriff
- ALOHA und Slotted ALOHA
- CSMA, CSMA/CD, CSMA/CA
- Token Passing

Rahmenbildung, Adressierung und Fehlererkennung

- Erkennung von Rahmengrenzen und Codetransparenz
- Adressierung und Fehlererkennung

Verbindung auf Schicht 1 und 2

- Hubs, Bridges und Switches

Motivation

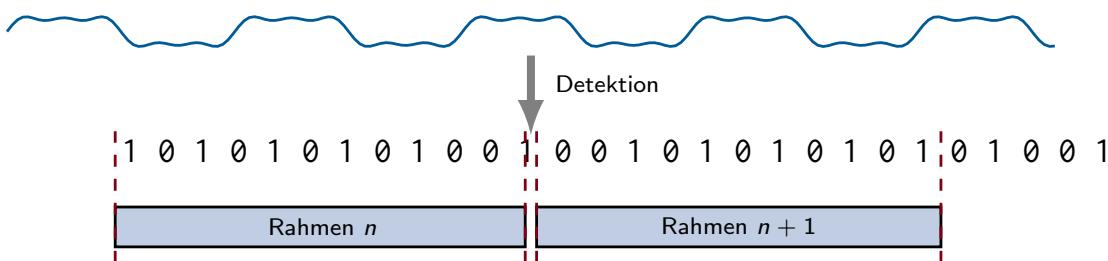
Bislang haben wir nur von **Nachrichten** gesprochen, ohne uns Gedanken über deren Format zu machen. Aus Sicht der physikalischen Schicht ist eine Nachricht lediglich eine Folge von Bits. Für eine Betrachtung der Sicherungsschicht reicht diese Vorstellung aber nicht mehr aus.

Im Folgenden wollen wir uns Gedanken machen,

- ▶ wie einzelne Nachrichten auseinander gehalten werden können,
- ▶ welche zusätzlichen Informationen Protokolle der Sicherungsschicht benötigen und
- ▶ wie Übertragungsfehler, die trotz Kanalkodierung auftreten, erkannt werden können.

Im Kontext der Sicherungsschicht bezeichnen wir Nachrichten fortan als **Rahmen** (engl. **Frame**).

Erkennung von Rahmengrenzen



Wie kann der Empfänger Rahmen erkennen, insbesondere wenn

- ▶ Rahmen unterschiedliche Größen haben und
- ▶ nicht ständig Nutzdaten auf der Leitung liegen (Idle-Perioden)?

Es gibt viele Möglichkeiten:

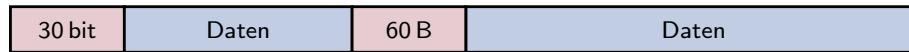
- ▶ Längenangabe der Nutzdaten
- ▶ Steuerzeichen (Start / Ende)
- ▶ Begrenzungsfelder und „Bit-Stopfen“
- ▶ Coderegelverletzung

Ziel aller Verfahren zur Rahmenbegrenzung ist die Erhaltung der **Codetransparenz**, d. h. die Übertragung beliebiger Zeichenfolgen zu ermöglichen.

Längenangabe der Nutzdaten

Idee:

- ▶ Am Anfang des Rahmens steht die Länge der nachfolgenden Nutzdaten (oder die Gesamtlänge des Rahmens).
- ▶ Voraussetzung: Das Längenfeld und damit der Beginn einer Nachricht muss eindeutig zu erkennen sein



Wie kann der Beginn eines Rahmens erkannt werden?

- ▶ Durch Steuerzeichen (Start / Ende)
- ▶ Durch Voranstellen von Begrenzungsfeldern
- ▶ Durch Verlust des Trägersignals zwischen den Rahmen (Coderegelverletzung, s. Kapitel 1)

Steuerzeichen

In Kapitel 1 haben wir bereits den **4B5B-Code** kennengelernt, welcher in Kombination mit Leitungscodes wie MLT-3 auf der physikalischen Schicht eingesetzt wird.

- ▶ Je 4 bit Eingabe werden auf 5 bit Ausgabe abgebildet
- ▶ Einem Rahmen werden die Startsymbole J/K vorangestellt
- ▶ Nach einem Rahmen werden die Endsymbole T/R eingefügt

Eingabe	Ausgabe	Bedeutung	Eingabe	Ausgabe	Bedeutung
0000	11110	Hex data 0	-	00000	Quiet (Signalverlust)
0001	01001	Hex data 1	-	11111	Idle (Pause)
0010	10100	Hex data 2	-	11000	Start #1 (J)
0011	10101	Hex data 3	-	10001	Start #2 (K)
0100	01010	Hex data 4	-	01101	End (T)
0101	01011	Hex data 5	-	00111	Reset (R)
⋮	⋮	⋮	-	11001	Set
1111	11101	Hex data F	-	00100	Halt

Beispiel:

Eingabe: 1011 0101 0110
Ausgabe: 11000 10001 10111 01011 01110 01101 00111

Steuerzeichen werden nicht nur auf Schicht 1/2 verwendet. Auf Schicht 6 (Darstellungsschicht) wird der **ASCII-Code (American Standard Code for Information Interchange)** verwendet (7 bit Codeworte):

ASCII (hex)	Zeichen						
00	NUL	20	SP	40	@	60	'
01	SOH	21	!	41	A	61	a
02	STX	22	"	42	B	62	b
03	ETX	23	#	43	C	63	c
04	EOT	24	\$	44	D	64	d
05	ENQ	25	%	45	E	65	e
06	ACK	26	&	46	F	66	f
07	BEL	27	,	47	G	67	g
08	BS	28	(48	H	68	h
09	TAB	29)	49	I	69	i
0A	LF	2A	*	4A	J	6A	j
0B	VT	2B	+	4B	K	6B	k
0C	FF	2C	,	4C	L	6C	l
0D	CR	2D	-	4D	M	6D	m
0E	SO	2E	.	4E	N	6E	n
0F	SI	2F	/	4F	O	6F	o
10	DLE	10	0	50	P	70	p
11	DC1	11	1	51	Q	71	q
12	DC2	12	2	52	R	72	r
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:

Was ist, wenn Steuerzeichen zufällig in den Nutzdaten vorkommen?

1. Im Fall des 4B5B-Codes kann das nicht passieren:
 - ▶ 4 bit Datenworte werden injektiv auf 5 bit Datenworte abgebildet
 - ▶ Einige der verbleibenden 5 bit Worte werden als Steuerzeichen verwendet
2. Der ASCII-Code ist lediglich eine Interpretationsvorschrift:
 - ▶ Einige Codeworte sind Textzeichen (Ziffern, Zahlen, ...), andere Steuerzeichen
 - ▶ Um ein Steuerzeichen als Datum übertragen zu können, wird dieses durch ein spezielles Steuerzeichen markiert: **Escape Character**
 - ▶ Soll dieses spezielle Steuerzeichen selbst übertragen werden, so wird es verdoppelt
 - ▶ Dieses Vorgehen bezeichnet man als **Character Stuffing**

Meist wird automatisch für **Codetransparenz** gesorgt, so dass sich der Benutzer nicht darum kümmern muss. Das trifft nicht auf Programmiersprachen zu:

```
System.out.println("Ein \" muss escaped werden");
```

Innerhalb des auszugebenden Strings müssen Anführungszeichen mittels einer Backslashs escaped werden.

Weitere Beispiele:

- ▶ Bash (Ctrl+C)
- ▶ Texteditoren (Emacs)

Begrenzungsfelder und Bit-Stopfen

Idee:

- ▶ Markiere Start und Ende einer Nachricht mit einer bestimmten Bitfolge
- ▶ Stelle sicher, dass die Markierung nicht zufällig in den Nutzdaten vorkommt („Bit-Stopfen“, engl. Bit Stuffing)

Beispiel:

- ▶ Start- / Endemarkierung sei 01111110
- ▶ Um das Auftreten der Markierung in Nutzdaten zu verhindern, füge nach fünf aufeinanderfolgenden 1-en eine 0 ein

Eingabe:	110010111110111111
Ausgabe:	01111110 110010111110101111101 01111110

- ▶ Empfänger entfernt nach fünf aufeinanderfolgenden 1-en die darauf folgende 0

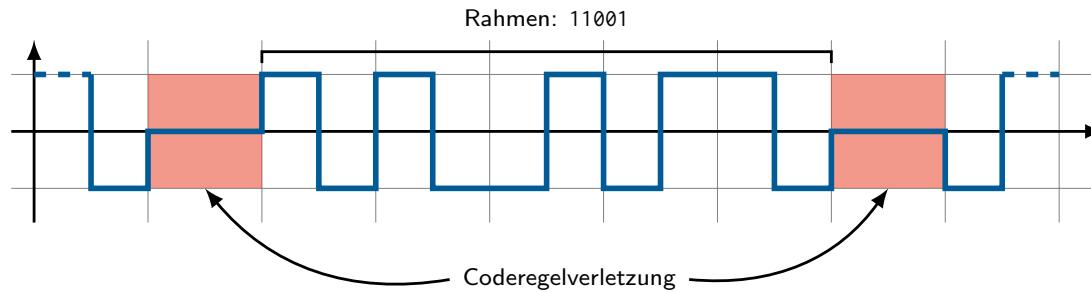
Coderegelverletzung

Viele Leitungscodes (z. B. RZ und Manchester) besitzen unabhängig von den zu übertragenden Daten bestimmte Signalwechsel.

Idee:

- ▶ Lasse bestimmte Signalwechsel aus
- ▶ Auf diese Art wird ein ungültiges (im Code nicht existierendes) Symbol erzeugt
- ▶ Dieses kann verwendet werden, um Start und Ende von Rahmen zu markieren

Beispiel: Manchester-Code



Fallbeispiele

IEEE 802.3a/i (Ethernet): 10 Mbit/s

- ▶ Als Leitungscode wird der Manchester-Code verwendet.
- ▶ Das Ende eines Frames wird durch Coderegelverletzung angezeigt.

IEEE 802.3u (FastEthernet): 100 Mbit/s

- ▶ Als Leitungscode wird MLT-3 in Kombination mit dem 4B5B-Code verwendet.
- ▶ Start und Ende von Rahmen werden durch Steuerzeichen des 4B5B-Codes markiert.

IEEE 802.3z (Gigabit Ethernet over Fiber): 1000 Mbit/s

- ▶ Als Leitungscode wird NRZ in Kombination mit dem 8B10B-Code verwendet.
- ▶ Start und Ende von Rahmen werden durch Steuerzeichen des 8B10B-Codes markiert.
- ▶ IEEE 802.3ab (Gigabit Ethernet over Copper) verwendet andere Leitungscodes, da die Dämpfung andernfalls zu groß wäre.

Zusätzlich wird bei all diesen Beispielen jedem Rahmen noch eine **Präambel** (s. Kapitel 1) vorangestellt. Diese dient allerdings nur der Takt synchronisierung zwischen Sender und Empfänger.

Übersicht

Problemstellung und Motivation

Darstellung von Netzwerken als Graphen

Netztopologien
Adjazenz- und Distanzmatrix
Erzeugung von Baumstrukturen

Verbindungscharakterisierung, Mehrfachzugriff, Medienzugriffskontrolle

Verbindungscharakterisierung
Medienzugriff
ALOHA und Slotted ALOHA
CSMA, CSMA/CD, CSMA/CA
Token Passing

Rahmenbildung, Adressierung und Fehlererkennung

Erkennung von Rahmengrenzen und Codetransparenz
Adressierung und Fehlererkennung

Verbindung auf Schicht 1 und 2

Hubs, Bridges und Switches

Adressierung und Fehlererkennung

Bislang wissen wir,

- ▶ wie ein binärer Datenstrom übertragen wird und
- ▶ wie der Empfänger Rahmengrenzen wiedererkennt.

Wir wissen aber noch nicht,

- ▶ wie Nutzdaten, die von Schicht 3 und höher kommen, von der Sicherungsschicht behandelt werden,
- ▶ wie der Empfänger eines Rahmens adressiert wird und
- ▶ wie aus den Nutzdaten und protokollspezifischen Informationen ein Rahmen entsteht.

Anmerkung: Alle folgenden Konzepte werden anhand der IEEE 802-Standards erklärt. Die wesentlichen Punkte sind mit kleinen Modifikationen auf andere Verfahren übertragbar.

Adressierung

In Direktverbindungsnetzen

- ▶ sind angeschlossene Knoten direkt erreichbar,
- ▶ es findet also keine Vermittlung (engl. [Routing](#)) zwischen Knoten statt.

Anforderungen an Adressen auf Schicht 2:

- ▶ [Eindeutige Identifizierung](#) der Knoten [innerhalb](#) des Direktverbindungsnetzes.
- ▶ Zumeist existiert eine [Broadcast-Adresse](#), welche alle Knoten im Direktverbindungsnetz anspricht.
- ▶ Zusätzlich kann es [Multicast-Adressen](#)⁶ geben, die bestimmte Gruppen von Knoten ansprechen.

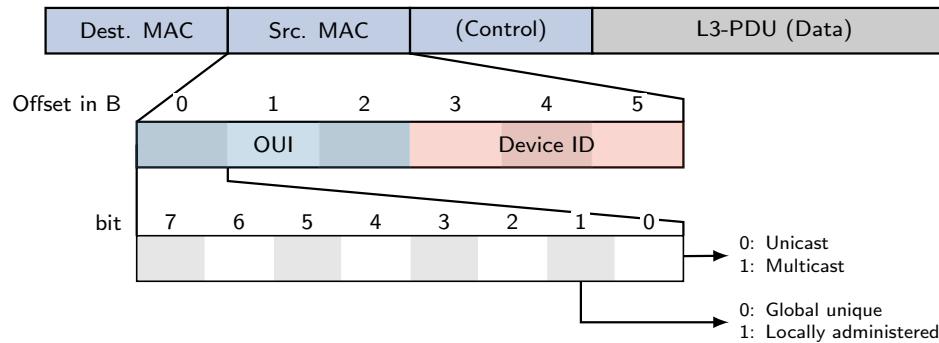
Adressen auf Schicht 2 bezeichnet man allgemein als [MAC-Adressen](#), wobei MAC für [Media Access Control](#) steht.

Beispiel:

Dest. MAC	Src. MAC	(Control)	L3-PDU (Data)
-----------	----------	-----------	---------------

⁶ Multicast-Adressen auf Schicht 2 werden häufig wie Broadcasts behandelt. Speziell im Einsatz mit IPv6 in geswitchten Netzwerken sind sie aber von großer Bedeutung.

MAC-Adressen aller IEEE 802-Standards (z.B. Ethernet, WLAN, Bluetooth) haben den folgenden Aufbau:



- ▶ Netzwerkkarten besitzen ab Werk eine MAC-Adresse. Diese ist meist im ROM ([Read Only Memory](#)) der Netzwerkkarte hinterlegt.
- ▶ Die Aufteilung in OUI und Device ID ermöglicht es den Herstellern von Netzwerkkarten, eindeutige MAC-Adressen zu vergeben.
- ▶ Vergeben werden die OUIs von der IANA (Internet Assigned Numbers Authority) [1].
- ▶ Daher ist möglich, den Hersteller einer Netzwerkkarte anhand deren MAC-Adresse zu identifizieren (z. B. 7c:6d:62 = Apple).
- ▶ Als Broadcast-Adresse ist **ff:ff:ff:ff:ff:ff** („all ones“) definiert.
- ▶ Ob es sich bei einer Adresse um eine Unicast- oder Multicast-Adresse handelt, bestimmt das lowest order Bit des ersten Oktetts.
- ▶ Für bestimmte Anwendungen ist es sinnvoll, auf die herstellerübergreifende Eindeutigkeit zu verzichten, z.B. bei virtualisierten Netzwerkadapters. Hierfür sind die sog. [lokal-administrierten](#) Adressen (zweites Bit des ersten Oktetts) vorgesehen.

Fehlererkennung

- ▶ Trotz Kanalkodierung können Übertragungsfehler (Bitfehler) auftreten.
- ▶ Es kann daher passieren, dass eine fehlerhafte Payload an höhere Schichten weitergeleitet wird.

Um die Wahrscheinlichkeit für derartige Fehler zu minimieren, werden **fehlererkennende Codes** eingesetzt (sog. **Prüfsummen**):



Im Gegensatz zur Kanalkodierung dient die Prüfsumme eines Schicht-2-Protokolls üblicherweise nicht der Fehlerkorrektur sondern lediglich der Fehlererkennung.

Cyclic Redundancy Check (CRC) [3]

Im Gegensatz zu **fehlerkorrigierenden** Codes (Kanalcodes, Kapitel 1), handelt sich bei CRC um eine Familie **fehlererkennender** Codes:

- ▶ Eine grosse Anzahl von Fehlern (Einbit-, Mehrbit-, Burstfehler) sollen erkannt werden.
- ▶ Die zugefügte Redundanz soll gering sein.
- ▶ Fehler sollen lediglich erkannt aber nicht korrigiert werden können.

Grundlagen:

- ▶ Ein Datenwort der Länge n bit lässt sich darstellen als Polynom

$$a(x) = \sum_{i=0}^{n-1} a_i x^i \text{ mit } a_i \in \{0,1\}.$$

- ▶ Alle Datenworte der Länge genau n bit bilden die Menge

$$F_q[x] = \left\{ a \mid a(x) = \sum_{i=0}^{n-1} a_i x^i, a_i \in \{0,1\} \right\}.$$

- ▶ Zusammen mit passend definierter Addition und Multiplikation entsteht ein sog. **endlicher Körper** $\langle F_q[x], +, \cdot \rangle$ mit $q = 2^n$ Elementen, auf dem die üblichen Regeln zur Addition und Multiplikation gelten.

Was heißt „passend definiert“?

Summe:

- ▶ Für die Summe zweier beliebiger $a, b \in F_q[x]$ erhalten wir

$$c(x) = a(x) + b(x) = \sum_{i=0}^{n-1} (a_i + b_i) x^i,$$

wobei für die Summe der Koeffizienten die Addition des GF(2)⁷ gilt, d.h. die Summe zweier Datenwörter entspricht einer bitweisen XOR-Verknüpfung.

Produkt:

- ▶ Das Produkt ist komplizierter, da für $d(x) = a(x) \cdot b(x)$ der Grad von $d(x)$ im Allgemeinen größer als $n - 1$ ist und damit $d(x) \notin F_q[x]$.
- ▶ Daher wählt man ein **Reduktionspolynom $r(x)$** mit $\text{grad}(r(x)) = n$ und definiert das Produkt von $a, b \in F_q[x]$ als

$$d(x) = (a(x) \cdot b(x)) \bmod r(x).$$

- ▶ Dies entspricht einer normalen Polynommultiplikation (wobei die Addition einer XOR-Verknüpfung entspricht) mit anschließender Modulo-Operation über $r(x)$.
- ▶ Die Modulo-Operation entspricht einer normalen Polynomdivision (XOR als Addition), wobei das Ergebnis der Divisionsrest ist.
- ▶ Dies sorgt dafür, dass $\text{grad}(d(x)) < n$ ist.

⁷ Galois Feld, Körper mit zwei Elementen (s. Diskrete Strukturen)

Anmerkungen:

- ▶ Wählt man für $r(x)$ ein **irreduzibles** Polynom, d.h. heißt $r(x)$ kann nicht als Produkt zweier $a, b \in F_q[x]$ dargestellt werden, so erhält man einen **endlichen Körper mit $q = 2^n$ Elementen**.
- ▶ Für CRC wählt man häufig $r(x) = p(x)(x + 1)$, wobei $p \in F_q[x]$ ein irreduzibles Polynom des Grads $n - 1$ ist:
 - ▶ Sowohl $p(x)$ als auch $x + 1$ sind Elemente von $F_q[x]$ und $r(x)$ ist als Produkt zweier solcher Elemente offensichtlich nicht irreduzibel.
 - ▶ Mit dieser Wahl von $r(x)$ ist $\langle F_q[x], +, \cdot \rangle$ **kein** endlicher Körper.
 - ▶ Diese Wahl von $r(x)$ ermöglicht es jedoch, alle ungeradzahligen Fehler zu erkennen.
- ▶ Die Wahl von $r(x)$ bestimmt also nicht nur die Länge der Prüfsumme, sondern auch maßgeblich die Fehlererkennungseigenschaften.

Zurück zu CRC

- ▶ CRC berechnet zu einem gegebenen Datenblock (z. B. L2-PDU) eine Checksumme fester Länge.
- ▶ Codewörter sind Polynome $a \in F_q[x]$.
- ▶ Der Grad n des Reduktionspolynoms $r(x)$ bestimmt
 - ▶ den maximalen Grad $n - 1$ aller möglichen Codewörter $a \in F_q[x]$ sowie
 - ▶ welche Arten von Bitfehlern (Einbit-, Mehrbit-, Burstfehler) erkannt werden können.
- ▶ Ethernet verwendet CRC32 mit dem Reduktionspolynom

$$r(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1.$$

Wie funktioniert CRC?

Angenommen wir haben ein Reduktionspolynom $r(x)$ des Grads n und eine Nachricht $m(x)$ der Länge $n - 1$ bit, die mittels CRC gesichert werden soll:

1. Hänge n Nullen an $m(x)$ an:

$$m'(x) = m(x) \cdot x^n.$$

2. Bestimme den Divisionsrest

$$c(x) = m'(x) \bmod r(x),$$

welcher der Checksumme entspricht.

3. Die zu sendende Nachricht besteht aus der Summe

$$s(x) = m'(x) + c(x).$$

Der Empfänger prüft die eingehende Nachricht $s'(x) = s(x) + e(x)$, welche möglicherweise einen Übertragungsfehler $e(x) \neq 0$ enthält:

1. Er bestimmt den Divisionsrest

$$c'(x) = s'(x) \bmod r(x) = (s(x) + e(x)) \bmod r(x).$$

2. Ist $c'(x) = 0$, so ist mit **hoher Wahrscheinlichkeit kein** Übertragungsfehler aufgetreten. Ist $c'(x) \neq 0$, so ist **sicher** ein Fehler aufgetreten.

Beispiel: Reduktionspolynom $r(x) = x^3 + x^2 + 1$, Daten $m(x) = x^7 + x^5 + x^2 + 1$

1. Koeffizienten bestimmen: $r(x) \hat{=} 1101$ und $m(x) \hat{=} 10100101$
2. $\text{grad}(r(x)) = 3 \Rightarrow$ Daten mit x^3 multiplizieren. Dies entspricht dem „Anhängen“ von 3 Nullen:
 $m'(x) = m(x) \cdot x^3 \hat{=} 10100101000$
3. Polynomdivision $m'(x)/r(x)$ ausführen und den Rest (Checksumme) $c(x)$ bestimmen.
4. Die zu sendende Nachricht ist $s(x) = m'(x) + c(x)$. Die Addition reduziert sich auf ein XOR, da wir auf GF(2) arbeiten.

$m'(x)$										$r(x)$				
1	0	1	0	0	1	0	1	0	0	0	:			
1	1	0	1								=			
0	1	1	1	0										
1	1	0	1											
0	0	1	1	1	1	0								
		1	1	0	1									
0	0	1	1	1	1	0								
		1	1	0	1									
0	0	1	1	0	0	0								
		1	1	0	1									
		0	0	0	1									
											$= c(x)$			

Übertragungsfehler steht:

$$\begin{array}{r}
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 = m'(x) \\
\oplus & & & & & & & & & & 0 & 0 & 1 = c(x) \\
\hline
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 = s(x)
\end{array}$$

Der Empfänger prüft die Nachricht, indem er $c'(x) = (s(x) + e(x))/r(x)$ bestimmt, wobei $e(x)$ für mögliche Übertragungsfehler steht:

- $c'(x) \neq 0$ besagt, dass sicher ein Fehler aufgetreten ist
- $c'(x) = 0$ besagt, dass mit hoher Wahrscheinlichkeit kein Fehler aufgetreten ist

Welche Fehler erkennt CRC?

Sei N die Länge der Checksumme, also $N = \text{grad}(r(x))$. Dann werden die folgenden Fehler erkannt:

- ▶ Alle 1 bit-Fehler
- ▶ Isolierte 2 bit-Fehler, d. h. Fehler an den Bitstellen i und j wobei $i > j$
- ▶ Einige Burst-Fehler, die länger sind als N

Abhängig von der konkreten Wahl des Reduktionspolynoms können auch entweder

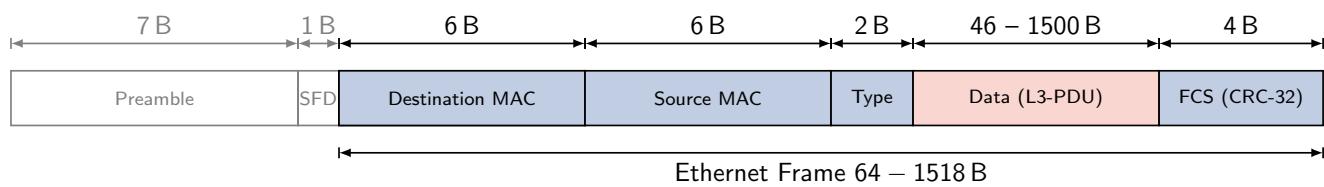
- ▶ alle **Burst-Fehler**, deren Länge kleiner ist als N oder
 - ▶ alle Fehlermuster, deren Anzahl der Fehlerbits ungerade ist
- erkannt werden.

Welche Fehler erkennt CRC nicht zuverlässig oder gar nicht?

- ▶ Fehler, die länger sind als N
- ▶ Fehler, die aus mehreren Bursts bestehen
- ▶ Alle Fehler, die ein Vielfaches des Reduktionspolynoms sind

Fallbeispiel: IEEE 802.3u (FastEthernet)

Frame **vor** der 4B5B-Kodierung:



- ▶ Präambel und **Start Frame Delimiter (SFD)** dienen der Takt synchronisation.
- ▶ Das erste Byte der Präambel wird durch das J/K-Symbol des 4B5B-Codes ersetzt (Start of Frame).
- ▶ Nach der **Frame Check Sequence (FCS)** wird das T/R-Symbol des 4B5B-Codes eingefügt (End of Frame).
- ▶ Zwischen J/K und T/R liegende Daten werden gemäß des 4B5B-Codes kodiert.
- ▶ Das Typfeld gibt die Art des Frames an (z. B. 0x0800 ≈ IPv4 Payload, 0x0806 ≈ ARP).
- ▶ Das Datenfeld muss (vor der Kodierung) mind. 46 B lang sein – andernfalls wird es bis zu diesem Wert **gepadded**.

Übersicht

Problemstellung und Motivation

Darstellung von Netzwerken als Graphen

Netztopologien
Adjazenz- und Distanzmatrix
Erzeugung von Baumstrukturen

Verbindungscharakterisierung, Mehrfachzugriff, Medienzugriffskontrolle

Verbindungscharakterisierung
Medienzugriff
ALOHA und Slotted ALOHA
CSMA, CSMA/CD, CSMA/CA
Token Passing

Rahmenbildung, Adressierung und Fehlererkennung

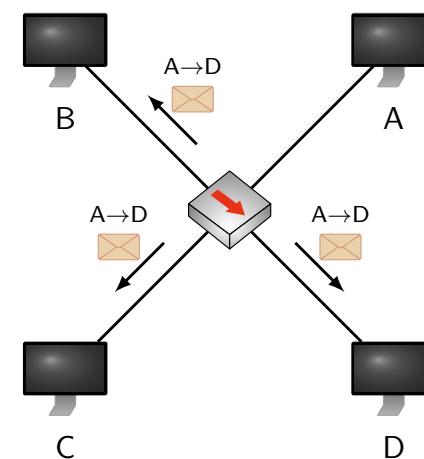
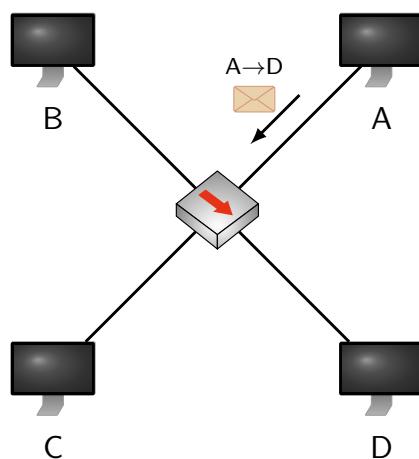
Erkennung von Rahmengrenzen und Codetransparenz
Adressierung und Fehlererkennung

Verbindung auf Schicht 1 und 2

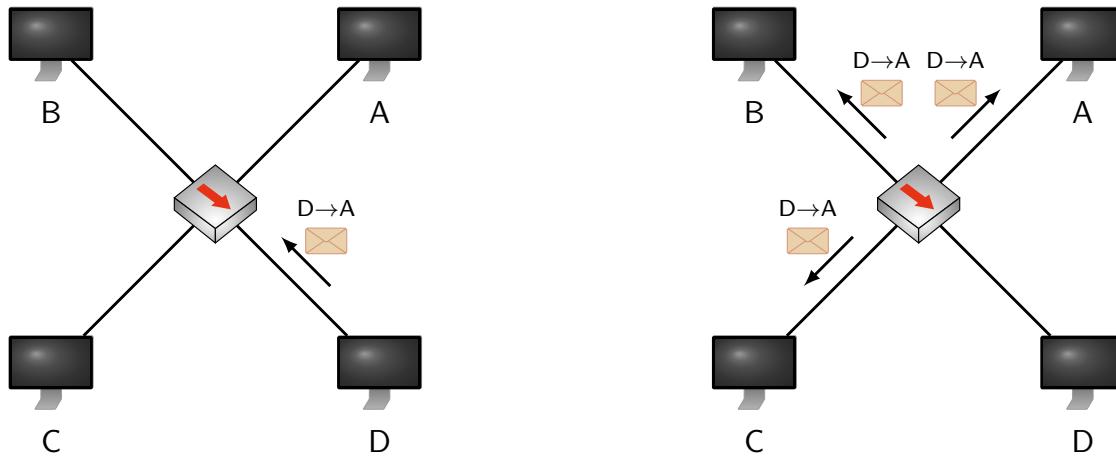
Hubs, Bridges und Switches

Verbindung auf Schicht 1: Hub [4]

- ▶ Knoten A sendet einen Rahmen an Knoten D
- ▶ Der Hub verbindet die einzelnen Links zu einem gemeinsamen Bus
- ▶ Der Rahmen erreicht **alle** Knoten
- ▶ Es darf folglich zu jedem Zeitpunkt **nur ein** Knoten senden, andernfalls treten **Kollisionen** auf



- ▶ Knoten D antwortet auf den Rahmen von A
- ▶ Auch die Antwort erreicht alle Knoten



Definition (Collision Domain)

Unter einer **Kollisions-Domäne** versteht man den Teil eines Direktverbindungsnetzes, innerhalb dem eine Kollision bei gleichzeitiger Übertragung mehrerer Knoten auftreten kann. Dieser wird häufig auch als **Segment** bezeichnet.

Sind Hubs mehr als nur Sternverteiler?

Man unterscheidet aktive und passive Hubs:

- ▶ **Aktive Hubs (Repeater)** verstärken die Signale auf der physikalischen Schicht, ohne dabei die in Rahmen enthaltenen Felder wie Adressen oder Checksummen zu prüfen
- ▶ **Passive Hubs** sind wirklich nur Sternverteiler – man könnte genauso gut die einzelnen Adern der Patchkabel verlöten

Kann man Hubs kaskadieren?

Ja, aber es gilt bei Ethernet mit Baumtopologie (802.3a/i) die **5-4-3-Regel**:

- ▶ Nicht mehr als 5 Abschnitte,
- ▶ verbunden durch 4 Repeater,
- ▶ wobei nur in 3 Abschnitten aktive Endgeräte enthalten sein dürfen.

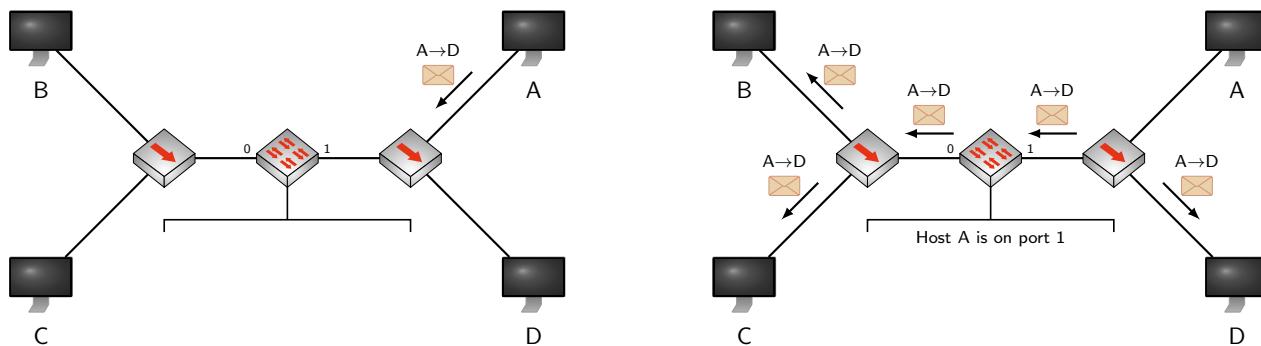
Jeder Abschnitt soll aufgrund der Dämpfung bei 802.3a (10BASE-2) nicht länger als 185 m sein, bei 802.3i (10BASE-T) nicht länger als 100 m zwischen Hub und Endgerät (Dämpfung).

Aufgrund einer sicheren Kollisionserkennung ergibt sich bei 100BASE-TX eine maximale Ausdehnung von 500 m (→ Übung).

Können Hubs unterschiedliche Medientypen miteinander verbinden?

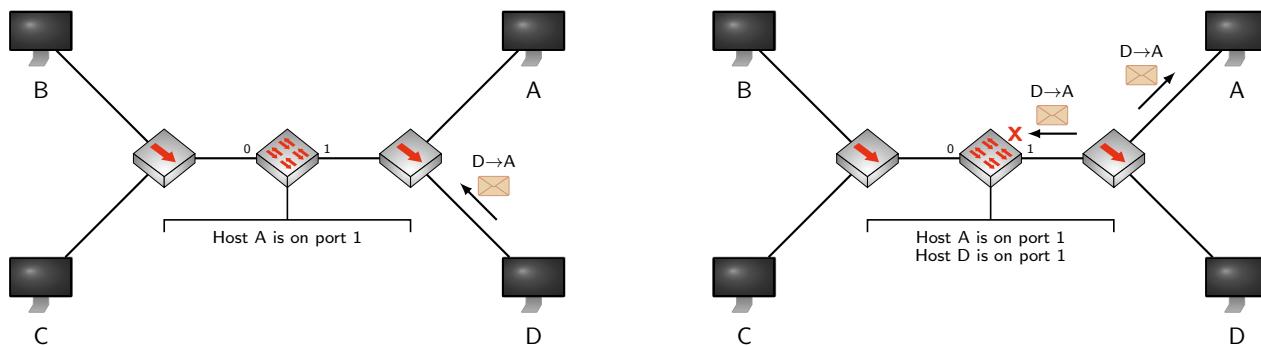
- ▶ Ja, wenn auf allen Abschnitten dasselbe Medienzugriffsverfahren genutzt wird (beispielsweise Verbindung Ethernet über BNC- und Patch-Kabel mit jeweils gleicher Datenrate).
- ▶ Unterschiedliche Zugriffsverfahren können nicht gekoppelt werden.

Verbindung auf Schicht 2: Switch [4]



- ▶ Zwei Gruppen von Hosts, die jeweils über Hubs verbunden sind, werden im obigen Beispiel durch ein **Switch** gekoppelt.
- ▶ Das Switch arbeitet zunächst wie ein Hub mit 2 Ports (Learning-Phase).
- ▶ Dabei merkt sich das Switch, über welchen Port ein Rahmen empfangen wurde.
- ▶ So ordnet es den Ports 0 und 1 die MAC-Adressen der Knoten zu, die an den jeweiligen Port angeschlossen sind.
- ▶ Ein Switch mit nur zwei Ports (was es heute wieder (!) im Kontext von Virtualisierung gibt), nennt man auch **Bridge**.

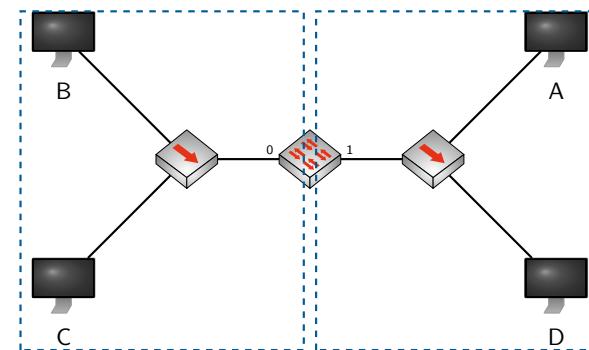
Verbindung auf Schicht 2: Switch



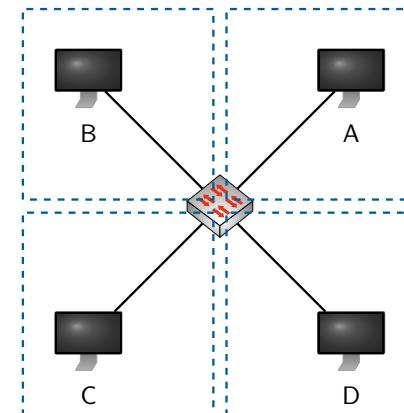
- ▶ Die Ziel-Adresse eingehender Rahmen wird mit den Einträgen in der **Switching-Table** verglichen.
- ▶ Ist ein Eintrag vorhanden, wird der Rahmen **nur** an den betreffenden Ziel-Port weitergeleitet.
- ▶ Ist kein Eintrag vorhanden, so wird der Rahmen an alle Ports weitergeleitet.
- ▶ Einträge erhalten einen Zeitstempel (Timestamp) und werden nach einem festen Zeitintervall invalidiert.

Verbindung auf Schicht 2: Switch

- ▶ Ein Switch bzw. eine Bridge unterbricht Kollisionsdomänen (auch als Segmentierung bezeichnet).
- ▶ Wenn ein Switch alle angeschlossenen Geräte kennt, darf in jedem der beiden Segmente jeweils ein Knoten zur selben Zeit senden.



- ▶ Ist pro Switchport genau ein Host angeschlossen, spricht man von Microsegmentation oder einem vollständig geswitchtem Netz (heute der Regelfall).
- ▶ In diesem Fall können jeweils zwei beliebige Hosts gleichzeitig miteinander kommunizieren.

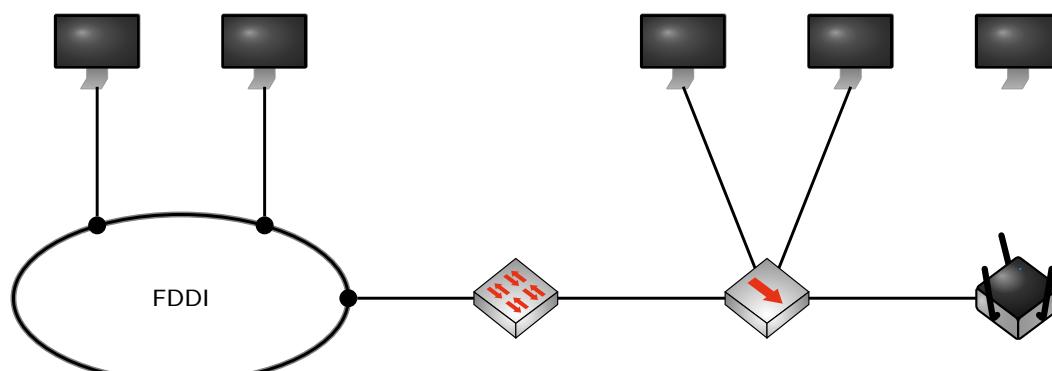


Switches können auch genutzt werden, um Netzsegmente mit unterschiedlichen Zugriffsverfahren zu koppeln:

- ▶ FDDI-Ethernet-Switch zwischen Token Passing und CSMA/CD
- ▶ WLAN Access Point zwischen CSMA/CD und CSMA/CA

Diese Kopplung ist transparent, d. h.

- ▶ angeschlossene Stationen bemerken nicht, dass ein Switch verwendet wird und
- ▶ im normalen Betrieb wird ein Host niemals direkt mit einem Switch kommunizieren.



Voraussetzung:

Die MAC-Adressen müssen „kompatibel“ sein, um Empfänger über ihre MAC-Adressen identifizieren zu können.

Anmerkungen

- ▶ Switches sind für Hosts **transparent**, d. h. ein Host weiß nicht, dass er über ein Switch mit anderen Hosts kommuniziert.
- ▶ Sender- und Empfänger-Adresse werden von Switches **nicht verändert**
- ▶ Switches schränken nicht die Erreichbarkeit innerhalb des Direktverbindungsnetzes ein
- ▶ Ein Broadcast (MAC-Adresse `ff:ff:ff:ff:ff:ff`) wird von allen Hosts empfangen (man spricht daher auch von **Broadcast-Domänen** im Unterschied zu einer Kollisions-Domäne)
- ▶ Switches benötigen zur Erfüllung ihrer grundlegenden Aufgaben **keine eigene** Adresse
- ▶ Weiterleitungsentscheidungen werden aber auf Basis der Ziel-Adresse und der aktuellen Switching-Tabelle getroffen

Ferner unterscheidet man zwischen zwei unterschiedlichen Switching-Arten:

- ▶ **Store-and-Forward:** Eingehende Rahmen werden vollständig empfangen und deren FCS geprüft. Falls der Ausgangsport belegt ist, kann eine begrenzte Anzahl von Rahmen gepuffert werden.
- ▶ **Cut-Through:** Beginne mit der Serialisierung des Rahmens, sobald der Ausgangsport bestimmt wurde. Die FCS wird in diesem Fall nicht geprüft.

Schleifen auf Schicht 2

Problembeschreibung

- ▶ Schleifen auf Schicht 1 bedeuten Kurzschluss.
- ▶ Schleifen auf Schicht 2 führen dazu, dass mehrere Kopien eines Rahmens erzeugt werden und im Netzwerk zirkulieren.

Wie entstehen Schleifen?

- ▶ Auch wenn Direktverbindungsnetze räumlich begrenzt sind, kann man schnell den Überblick verlieren und ungewollt Schleifen erzeugen.
- ▶ Manchmal erzeugt man Schleifen auch absichtlich, so dass redundante Pfade entstehen. Fällt eine Verbindung aus, kann der Verkehr umgeleitet werden.

Wie werden Schleifen vermieden?

- ▶ Switches unterstützen das sog. **Spanning Tree Protocol (STP)**
- ▶ Ziel ist die Deaktivierung redundanter Pfade, so dass alle Netzsegmente **schleifenfrei** erreichbar sind
- ▶ Fällt eine Verbindung aus, wird ggf. einer dieser Pfade reaktiviert

Zusammenfassung

Wir sollten wissen,

- ▶ wie Netzwerke als Graphen dargestellt werden können,
- ▶ was der Unterschied zwischen einem MST und einem SPT ist,
- ▶ welche unterschiedlichen Medienzugriffsverfahren es gibt,
- ▶ wie diese Kollisionen vermeiden oder mit ihnen umgehen,
- ▶ warum die maximale Länge eines Ethernet-Segments 500 m beträgt,
- ▶ wie Knoten in Direktverbindungsnetzen adressiert werden,
- ▶ wie MAC-Adressen bei Ethernet aufgebaut sind,
- ▶ wie mehrere Direktverbindungsnetze zu einem größeren miteinander verbunden werden können,
- ▶ worin der Unterschied zwischen Hubs, Bridges und Switches besteht,
- ▶ wie Switches lernen, an welchem Port, welche Geräte angeschlossen sind und wie Weiterleitungsentscheidungen getroffen werden und
- ▶ was eine Kollisions-Domäne bzw. eine Broadcast-Domäne ist.

Bibliography I

- [1] D. Eastlake and J. Abley. IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters, 2013. <http://tools.ietf.org/html/rfc7042>.
- [2] E. Stein. *Taschenbuch Rechnernetze und Internet*, chapter Konzepte: Lokale Netzwerke, pages 191–218. Fachbuchverlag Leipzig, 2. edition, 2004. Auszug s. Moodle/SVN.
- [3] E. Stein. *Taschenbuch Rechnernetze und Internet*, chapter Fehlererkennung durch CRC, pages 86–87. Fachbuchverlag Leipzig, 2. edition, 2004. Auszug s. Moodle/SVN.
- [4] E. Stein. *Taschenbuch Rechnernetze und Internet*, chapter Netzaufbau, pages 200–203. Fachbuchverlag Leipzig, 2. edition, 2004. Auszug s. Moodle/SVN.

Grundlagen Rechnernetze und Verteilte Systeme

Kapitel 3 – Vermittlungsschicht

Worum geht es in diesem Kapitel?

Motivation

Vermittlungsarten

- Leitungsvermittlung
- Nachrichtenvermittlung
- Paketvermittlung

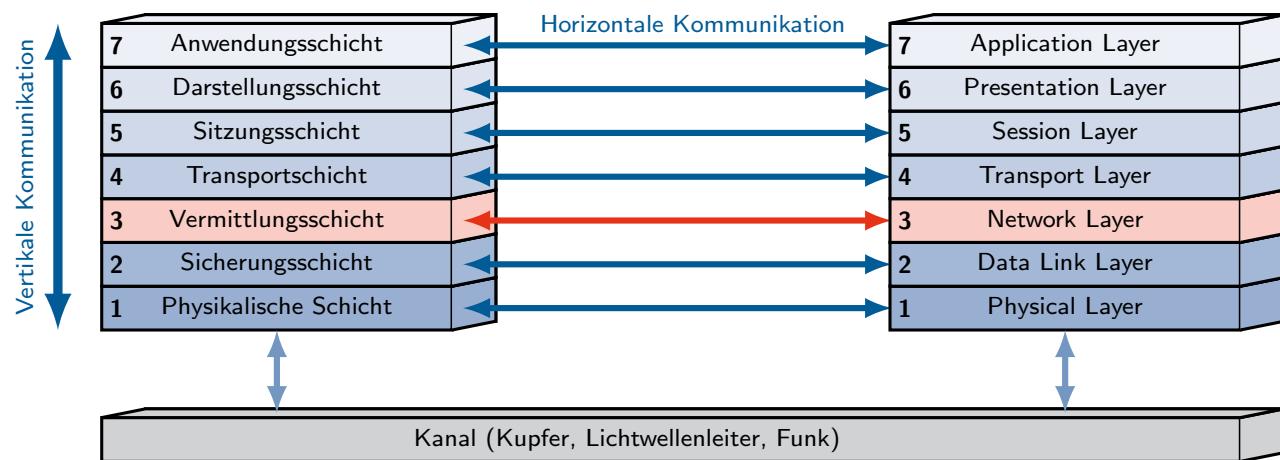
Adressierung im Internet

- Internet Protocol version 4 (IPv4)
- Internet Protocol version 6 (IPv6)

Wegwahl (Routing)

- Routing Table und Longest Prefix Matching
- Dynamisches Routing
- Routing Information Protocol (RIP)
- Autonome Systeme

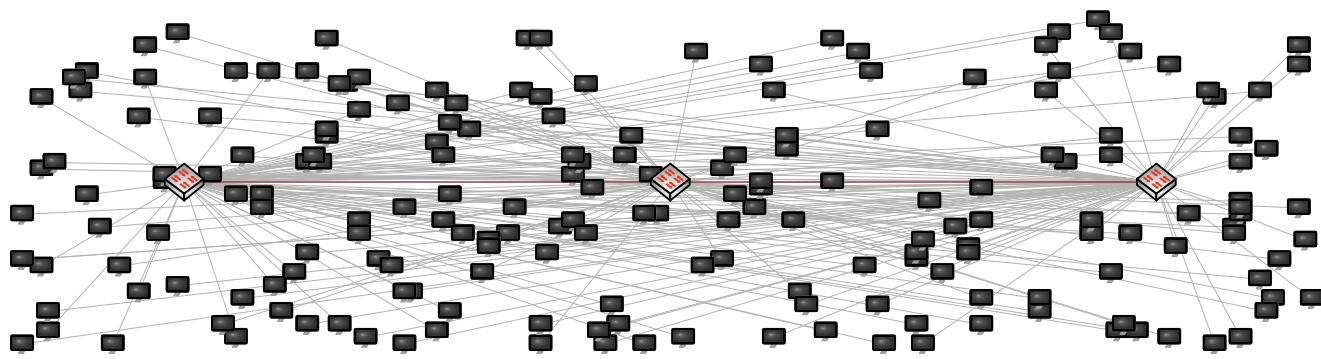
Einordnung im ISO/OSI-Modell



Motivation

Sind Direktverbindungsnetze wie Ethernet skalierbar?

- ▶ Alle angeschlossenen Hosts sind direkt bzw. über wenige Switches erreichbar
- ▶ MAC-Adressen bieten keine logische Struktur zur Adressierung
- ▶ Gruppierung von Geräten in kleinere Netze (**Subnetze**) durch MAC-Adressen nicht unterstützt



Aufgaben der Vermittlungsschicht:

- ▶ Kopplung unterschiedlicher Direktverbindungsnetze
- ▶ Strukturierte Aufteilung in kleinere Subnetze
- ▶ Logische und global eindeutige Adressierung von Geräten
- ▶ Wegwahl zwischen Geräten über mehrere **Hops** hinweg

Worum geht es in diesem Kapitel?

Motivation

Vermittlungsarten

Leitungsvermittlung
Nachrichtenvermittlung
Paketvermittlung

Adressierung im Internet

Internet Protocol version 4 (IPv4)
Internet Protocol version 6 (IPv6)

Wegwahl (Routing)

Routing Table und Longest Prefix Matching
Dynamisches Routing
Routing Information Protocol (RIP)
Autonome Systeme

Vermittlungsarten

Es gibt drei grundlegende Vermittlungsarten:

- ▶ **Leitungsvermittlung**
„Reserviere eine dedizierte Leitung zwischen Sender und Empfänger“
- ▶ **Nachrichtenvermittlung**
„Wähle für jede Nachricht individuell einen Weg“
- ▶ **Paketvermittlung**
„Teile eine Nachricht in mehrere kleinere Pakete auf und versende jedes Paket unabhängig von den anderen“

Im Folgenden charakterisieren wir diese drei Vermittlungsarten anhand des Beispielnetzwerks



mit $n = 2$ Vermittlungsknoten (i und j) hinsichtlich der Gesamtdauer T einer Übertragung von l Datenbits über die Distanz d und motivieren so die Vorteile der Paketvermittlung.

Leitungsvermittlung

Während einer verbindungsorientierten Übertragung können drei Phasen unterschieden werden:

1. Verbindungsaufbau

- ▶ Austausch von **Signalisierungsnachrichten** zum Aufbau einer **dedizierten Verbindung** zwischen Sender und Empfänger.
- ▶ Dieser Schritt beinhaltet die Wegwahl, welche vor Beginn der Datenübertragung durchgeführt wird.

2. Datenaustausch

- ▶ Kanal steht den Kommunikationspartnern zur **exklusiven Nutzung** bereit.
- ▶ Auf die Adressierung des Kommunikationspartners kann während der Übertragung weitgehend verzichtet werden (Punkt-zu-Punkt-Verbindung).

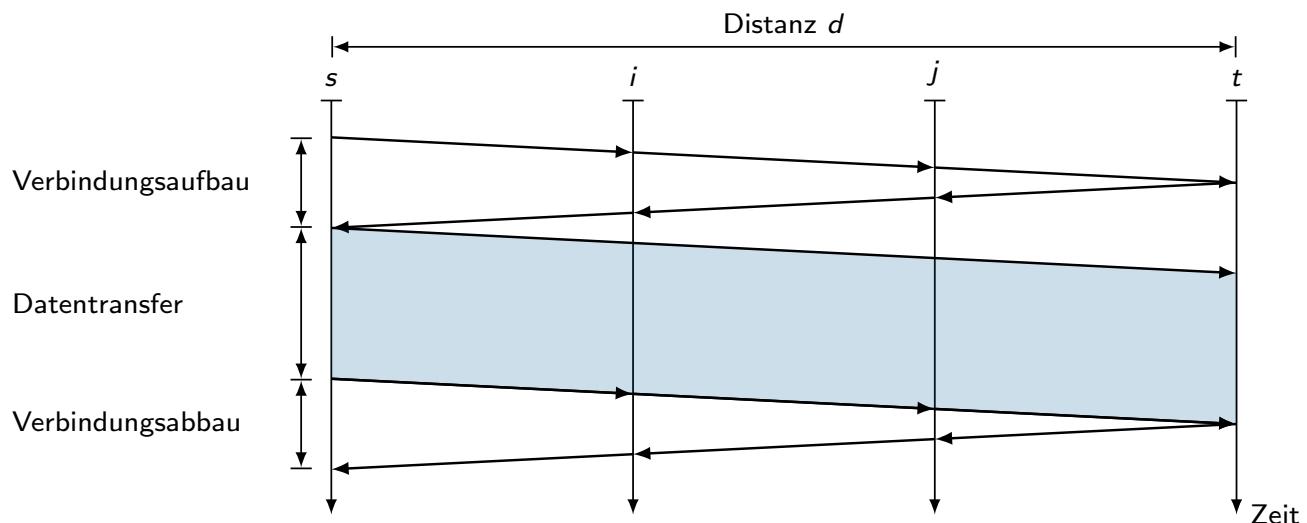
3. Verbindungsabbau

- ▶ Austausch von Signalisierungsnachrichten zum Abbau der Verbindung.
- ▶ Die durch die Verbindung belegten Ressourcen werden für nachfolgende Verbindungen freigegeben.

Übertragungszeit bei Leitungsvermittlung

Wir nehmen an, dass

- ▶ die Serialisierungszeit von Signalisierungsnachrichten vernachlässigbar klein ist,
- ▶ die Verarbeitungszeiten und Wartezeiten in jedem Knoten vernachlässigbar klein sind und dass
- ▶ der Sender s einen Datenblock der Länge L an einem Stück übertragen möchte.



$$T_{LV} = 2t_p + t_s + 2t_p = t_s + 4t_p = \frac{L}{r} + \frac{4d}{\nu c}$$

Vorteile der Leitungsvermittlung

- ▶ Gleichbleibende Güte der dedizierten Verbindung nach dem Verbindungsauflbau
- ▶ Schnelle Datenübertragung ohne Notwendigkeit, weitere Vermittlungsentscheidungen treffen zu müssen

Nachteile der Leitungsvermittlung

- ▶ Ressourcenverschwendungen, da Leitung zur exklusiven Nutzung reserviert wird
- ▶ Verbindungsauflbau kann komplex sein und benötigt u. U. weit mehr Zeit, als die Ausbreitungsverzögerungen vermuten lassen (z. B. Einwahl ins Internet mittels Modem)
- ▶ Hoher Aufwand bei der Schaltung physikalischer Verbindungen

Einsatz in heutigen Netzwerken

- ▶ Leitungsvermittlung wird häufig durch Paketvermittlung ersetzt (z. B. Voice over IP)
- ▶ In vielen Vermittlungsnetzen wird Leitungsvermittlung zumindest virtualisiert in Form von **Virtual Circuits** unterstützt (z. B. Frame Relay, ATM, MPLS)

Nachrichtenvermittlung

Veränderungen bei der Nachrichtenvermittlung:

- ▶ Aufbau und Abbau einer dedizierten Verbindung entfallen
- ▶ Der gesamten Nachricht der Länge L wird ein **Header** der Länge L_H vorangestellt



- ▶ Der Header beinhaltet insbesondere Adressinformationen, die geeignet sind, Sender und Empfänger auch über mehrere Zwischenstationen hinweg eindeutig zu identifizieren
- ▶ Die so entstehende PDU wird als Ganzes übertragen

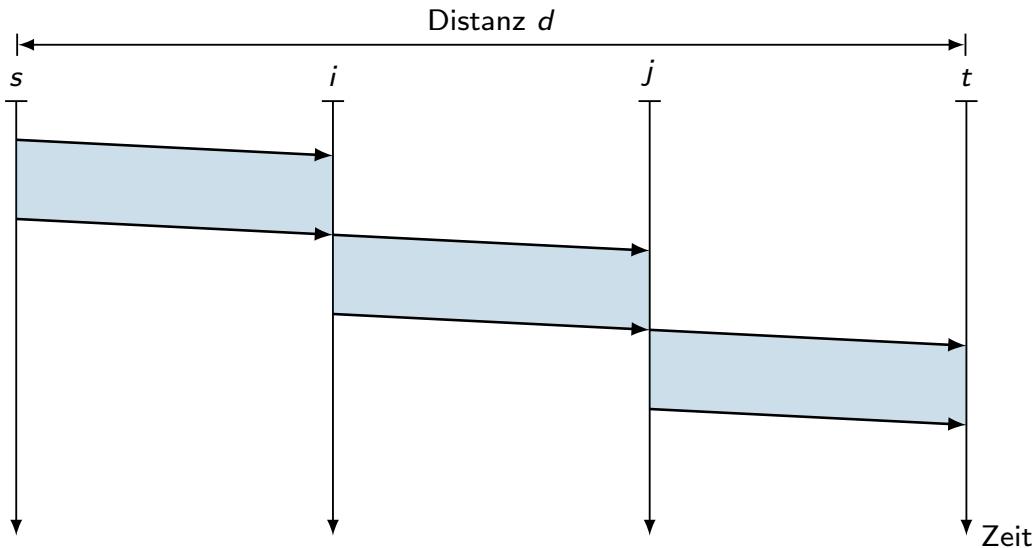
Hoffnung: Zeitersparnis, da die Phasen zum Aufbau und Abbau der Verbindung entfallen.

Analogie: Post / DHL / Paketdienste

- ▶ Absender verpackt Ware und versieht das Paket mit Adressinformationen (Header)
- ▶ Die Adressen identifizieren Absender und Empfänger weltweit eindeutig und haben eine logische Struktur, die eine effiziente Zuordnung im Transportnetz der Post erlaubt

Übertragungszeit bei Nachrichtenvermittlung

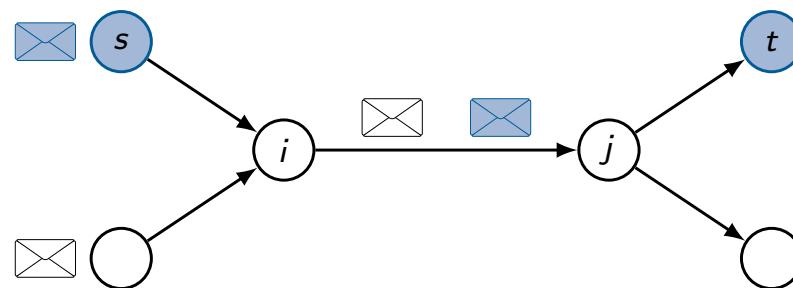
Erinnerung: Hier $n = 2$ Vermittlungsknoten i und j .



$$T_{NV} = (n + 1) \cdot t_s + t_p = (n + 1) \cdot \frac{L_H + L}{r} + \frac{d}{\nu c}$$

Multiplexing auf Nachrichtenebene

- ▶ Das Wegfallen fest vorgegebener Pfade ermöglicht die gemeinsame Nutzung von Teilstrecken
- ▶ Dies entspricht dynamischem **Zeitmultiplex** (Time Division Multiplex, TDM)



Vorteile:

- ▶ Flexibles Zeitmultiplex von Nachrichten
- ▶ Bessere Ausnutzung der Kanalkapazität
- ▶ Keine Verzögerung beim Senden des ersten Pakets durch Verbindungsaufbau

Nachteile:

- ▶ Pufferung von Nachrichten, wenn (i,j) ausgelastet
- ▶ Verlust von Nachrichten durch begrenzten Puffer möglich (Stausituation → Kapitel 4)
- ▶ Mehrfache Serialisierung der ganzen Nachricht

Paketvermittlung

Unterschiede zur Nachrichtenvermittlung:

- ▶ Nachrichten werden nicht mehr als Einheit übertragen sondern in kleinere **Pakete** unterteilt:



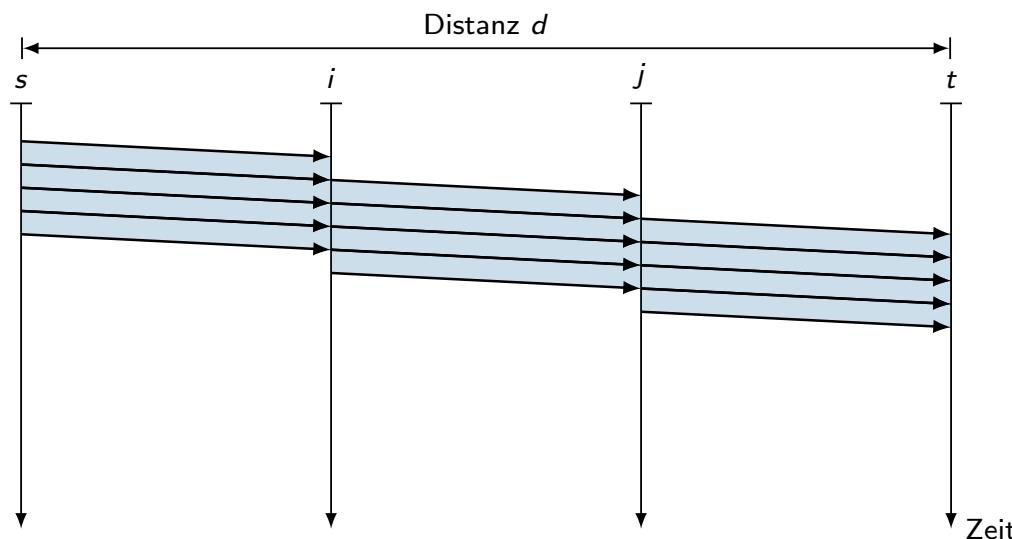
- ▶ Jedem Paket wird ein eigener Header vorangestellt, der alle Informationen zur Weiterleitung und ggf. auch zur Reassembly enthält:



- ▶ Pakete werden **unabhängig** voneinander vermittelt, d. h. Pakete derselben Nachricht können über unterschiedliche Wege zum Empfänger gelangen.
- ▶ Im Allgemeinen müssen die einzelnen Pakete nicht gleich groß sein, es gibt aber Anforderungen an die minimale (p_{\min}) und maximale (p_{\max}) Paketgröße.

Übertragungszeit bei Paketvermittlung

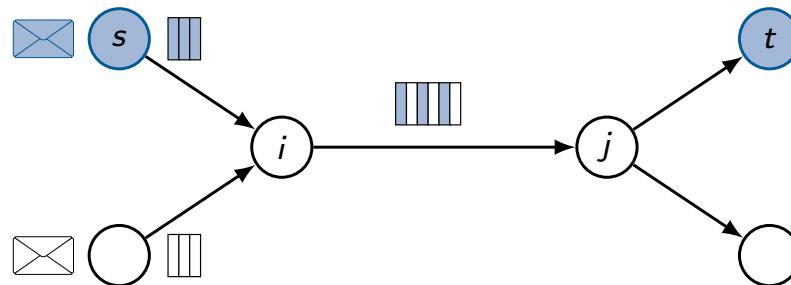
- ▶ Erinnerung: Hier $n = 2$ Vermittlungsknoten i und j .
- ▶ Vereinfachend nehmen wir an, dass die Nachrichtenlänge ein Vielfaches von p_{\max} ist.
(\Rightarrow alle Pakete haben dieselbe Größe p_{\max}).



$$T_{PV} = \frac{1}{r} \left(\left\lceil \frac{L}{p_{\max}} \right\rceil \cdot L_h + L \right) + \frac{d}{\nu c} + n \cdot \frac{L_h + p_{\max}}{r}$$

Multiplexing auf Paketebene

- ▶ Durch die Vermittlung kleiner Pakete statt langer Nachrichten werden Engpässe fairer genutzt
- ▶ Gehen Pakete verloren, müssen nur Teile einer größeren Nachricht wiederholt werden



Vorteile:

- ▶ Flexibles Zeitmultiplex einzelner Pakete
- ▶ Pufferung kleiner Pakete statt ganzer Nachrichten

Nachteile:

- ▶ Verlust von Paketen durch begrenzten Puffer möglich
- ▶ Jedes Paket benötigt seinen eigenen Header (Overhead)
- ▶ Empfänger muss Pakete wieder zusammensetzen

Vergleich der drei Verfahren

$$\text{Leitungsvermittlung: } T_{LV} = \frac{L}{r} + 4 \frac{d}{\nu c}$$

$$\text{Nachrichtenvermittlung: } T_{NV} = (n+1) \frac{L_h + L}{r} + \frac{d}{\nu c}$$

$$\text{Paketvermittlung: } T_{PV} = \frac{1}{r} \left(\left\lceil \frac{L}{p_{\max}} \right\rceil \cdot L_h + L \right) + \frac{d}{\nu c} + n \cdot \frac{L_h + p_{\max}}{r}$$

Zahlenbeispiel:

- ▶ Die Distanz zwischen Sender und Empfänger beträgt $d = 1000 \text{ km}$
- ▶ Für Glasfaserleitungen beträgt $\nu = 0.9$
- ▶ Es kommen $n = 5$ Vermittlungsknoten zum Einsatz
- ▶ Die Datenrate beträgt auf allen Teilstrecken $r = 1 \text{ Mbit/s}$
- ▶ Die Länge der zu sendenden Nachricht beträgt $L = 5 \text{ MiB}$
- ▶ Die maximale Paketgröße betrage $p_{\max} = 1480 \text{ B}$
- ▶ Die Headergröße pro Nachricht / Paket betrage $L_h = 20 \text{ B}$

Es ergeben sich folgende Zahlenwerte:

$$T_{LV} \approx 42 \text{ s}, \quad T_{NV} \approx 5033 \text{ s} \text{ und } T_{PV} \approx 43 \text{ s.}$$

Wo werden die Verfahren eingesetzt?

Leitungsvermittlung:

- ▶ Analoge Telefonverbindungen (POTS)
- ▶ Interneteinwahl („letzte Meile“)
- ▶ Standortvernetzung von Firmen
- ▶ **Virtuelle Kanäle** (engl. **Virtual Circuits**) in unterschiedlichen Arten von Vermittlungsnetzen (Frame Relay, ATM, MPLS, ...)

Nachrichtenvermittlung:

- ▶ Kaum praktische Anwendung auf Schicht 3
- ▶ Aber: Nachrichtenvermittlung existiert aus Sicht höherer Schichten (ab Schicht 5 aufwärts)

Paketvermittlung:

- ▶ In den meisten modernen Datennetzen
- ▶ Zunehmend auch zur Sprachübertragung (Voice over IP)
- ▶ Digitales Radio / Fernsehen
- ▶ Viele Peripherieschnittstellen an Computern (PCI, USB, Thunderbolt)

Übersicht

Motivation

Vermittlungsarten

Leitungsvermittlung
Nachrichtenvermittlung
Paketvermittlung

Adressierung im Internet

Internet Protocol version 4 (IPv4)
Internet Protocol version 6 (IPv6)

Wegwahl (Routing)

Routing Table und Longest Prefix Matching
Dynamisches Routing
Routing Information Protocol (RIP)
Autonome Systeme

Adressierung im Internet

Die Sicherungsschicht (Schicht 2) bietet

- ▶ mehr oder weniger fairen Medienzugriff bei von mehreren Hosts geteilten Medien,
- ▶ einen „ausreichenden“ Schutz vor Übertragungsfehler und
- ▶ Adressierung innerhalb eines Direktverbindungsnetzes.

Die Vermittlungsschicht (Schicht 3) ergänzt dies um

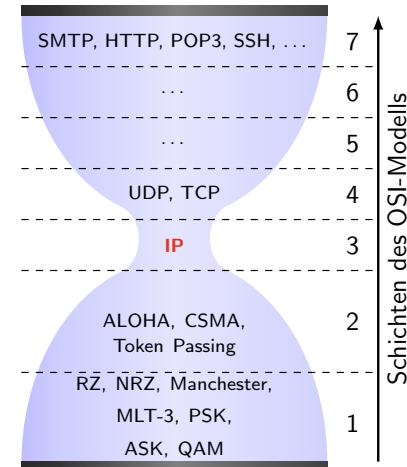
- ▶ Möglichkeiten zur global eindeutigen **und** strukturierten / logischen Adressierung sowie
- ▶ Verfahren zur Bestimmung von (möglichst) optimalen Pfaden.

Wir beschränken uns in diesem Teilkapitel auf die Betrachtung von

- ▶ **IPv4 (Internet Protocol v4, 1981)** bzw.
- ▶ seinem Nachfolger **IPv6 (1998)**.

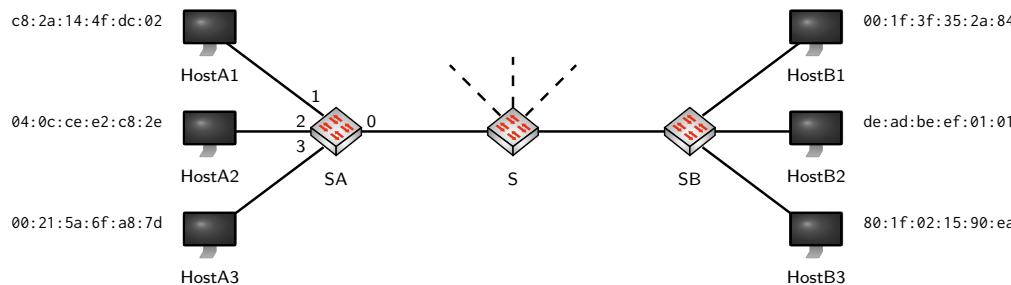
Beispiele für alternative Protokolle der Netzwerkschicht:

- ▶ IPX (Internetwork Packet Exchange, 1990)
- ▶ DECnet Phase 5 (1987)
- ▶ AppleTalk (1983)



Internet Protocol v4 (IPv4) [13]

Wir betrachten das folgende Beispielnetz, welches auf einem aktuellen Ethernet-Standard basiert:



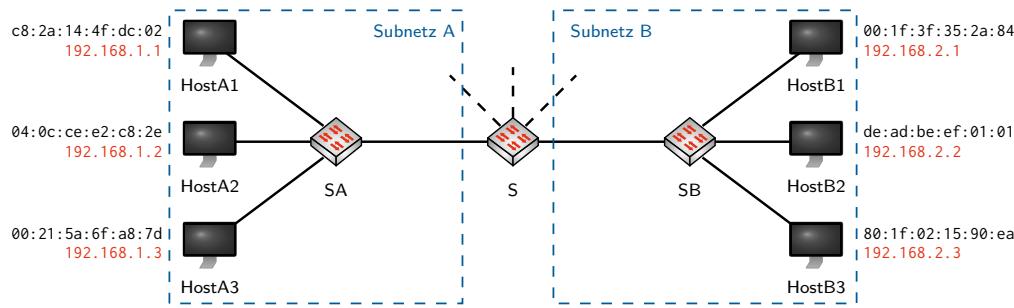
Wie viele Einträge enthält die Switching-Tabelle von Switch SA?

- ▶ Genau einen Eintrag für jeden bekannten Host
- ▶ Die meisten MAC-Adressen werden auf Port 0 abgebildet
- ▶ Eine Zusammenfassung von Einträgen ist i. A. nicht möglich, da MAC-Adressen nicht die Position eines Knotens innerhalb eines Netzwerks wiederspiegeln

Port	MAC
0	00:1f:3f:35:2a:84
0	de:ad:be:ef:01:01
0	80:1f:02:15:90:ea
1	c8:2a:14:4f:dc:02
2	04:0c:ce:e2:c8:2e
3	00:21:5a:6f:a8:7d
:	:

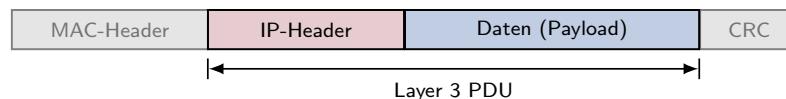
⇒ Es erscheint sinnvoll, von physikalischen Adressen zu abstrahieren

Wir betrachten das Beispielnetz mit einem Router (R) in der Mitte:



- ▶ Jedem Host ist eine **IP-Adresse** zugewiesen. Jede IP-Adresse ist in vier Gruppen zu je einem Byte, durch Punkte getrennt, dargestellt (**Dotted Decimal Notation**).
- ▶ In diesem Beispiel identifiziert das 4. Oktett einen Host innerhalb eines Netzes.
- ▶ Die ersten drei Oktette identifizieren das Netzwerk, in dem sich der Host befindet.
- ▶ Der Router R trifft Weiterleitungsentscheidungen auf Basis der Ziel-IP-Adresse.

⇒ Jedes Paket muss mit einer Absender- und Ziel-IP-Adresse (im IP-Header) versehen werden:



IP-Header

- ▶ Der IP-Header beinhaltet nicht nur Quell- und Ziel-Adresse.
- ▶ Die Verwendung der wichtigsten Felder wird später in diesem Kapitel anhand von Beispielen genauer erläutert werden.

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																									
0 B	Version		IHL		TOS										Total Length																																										
4 B	Identification										Flags		Fragment Offset																																												
8 B	TTL					Protocol					Header Checksum																																														
12 B	Source Address																																																								
16 B	Destination Address																																																								
20 B	Options / Padding (optional)																																																								

Abbildung: IPv4-Header (minimale Länge: 20 B)

Version

- ▶ Gibt die verwendete IP-Version an.
- ▶ Gültige Werte sind 4 (IPv4) und 6 (IPv6).

IHL (IP Header Length)

- ▶ Gibt die Länge des IP Headers inkl. Optionen in Vielfachen von 32 bit an.
- ▶ Wichtig, da der IPv4-Header durch Optionsfelder variable Länge hat.

TOS (Type of Service)

- ▶ Dient der Klassifizierung und Priorisierung von IP-Paketen (z. B. Hinweis auf zeitsensitive Daten wie Sprachübertragungen).
- ▶ Möglichkeit zur Staukontrolle (**Explicit Congestion Notification**) auf Schicht 3 (optional).

Total Length

- ▶ Gibt die Gesamtlänge des IP-Pakets (Header + Daten) in Bytes an.
- ▶ Die Maximallänge eines IP-Pakets beträgt damit 65 535 B.
- ▶ Der Sender passt die Größe ggf. an, um Fragmentierung zu vermeiden.
- ▶ Die maximale Paketlänge, so dass keine Fragmentierung notwendig ist, bezeichnet man als **Maximum Transmission Unit (MTU)**. Diese ist abhängig von Schicht 2/1 und beträgt bei FastEthernet 1500 B.

Identification

- ▶ Für jedes IP-Paket (zufällig) gewählter 16 bit langer Wert.
- ▶ Dient der Identifikation zusammengehörender Fragmente (**IP-Fragmentierung** → später).

Flags

- ▶ Bit 16: Reserviert und wird auf 0 gesetzt.
- ▶ Bit 17: **Don't Fragment (DF)**. Ist dieses Bit 1, so darf das IP-Paket nicht fragmentiert werden.
- ▶ Bit 18: **More Fragments (MF)**. Gibt an, ob weitere Fragmente folgen (1) oder dieses Paket das letzte Fragment ist (0). Wurde das Paket nicht fragmentiert, wird es ebenfalls auf 0 gesetzt.

Fragment Offset

- ▶ Gibt die absolute Position der Daten in diesem Fragment bezogen auf das unfragmentierte Paket in ganzzahligen Vielfachen von 8 B an.
- ▶ Ermöglicht zusammen mit dem Identifier und MF-Bit die Reassemblierung fragmentierter Pakete in der richtigen Reihenfolge.

TTL (Time to Live)

- ▶ Leitet ein Router ein IP-Paket weiter, so dekrementiert er das TTL-Feld um 1.
- ▶ Erreicht das TTL-Feld den Wert 0, so verwirft ein Router das Paket und sendet eine Benachrichtigung an den Absender (ICMP Time Exceeded → später).
- ▶ Dieser Mechanismus beschränkt die Pfadlänge im Internet und verhindert endlos kreisende Pakete infolge von **Routing Loops**.

Protocol

- ▶ Identifiziert das Protokoll auf Schicht 4, welches in der Payload (Daten) des IP-Pakets enthalten ist.
- ▶ Relevant u. a. für das Betriebssystem, um Pakete dem richtigen Prozess zuordnen zu können.
- ▶ Gültige Werte sind beispielsweise 0x06 (TCP) und 0x08 (UDP).

Header Checksum

- ▶ Einfache, auf Geschwindigkeit optimierte Prüfsumme, welche nur den IP-Header (ohne Daten) schützt.
- ▶ Die Checksumme ist so ausgelegt, dass die Dekrementierung des TTL-Felds einer Inkrementierung der Checksumme entspricht. Es ist also keine Neuberechnung der Checksumme bei der Weiterleitung von Paketen notwendig.
- ▶ Es ist lediglich Fehlererkennung aber keine Korrektur möglich.

Source Address

- ▶ IP-Adresse des Absenders.

Destination Address

- ▶ IP-Adresse des Empfängers.

Options / Padding

- ▶ IP unterstützt eine Reihe von Optionen (z. B. Zeitstempel, Route Recording, ...), welche als optionale Felder an den IP-Header angefügt werden können.
- ▶ Nicht alle diese Optionen sind 4 B lang. Da die Länge des IP-Headers jedoch ein Vielfaches von 4 B betragen muss, werden kürzere Optionen ggf. durch Padding auf den nächsten gültigen Wert ergänzt.

Einschub: Host-Byte-Order und Network-Byte-Order

Es gibt zwei unterschiedliche Byte-Orders:

1. Big Endian: „Niederwertigstes Byte steht an höchstwertigster Adresse“
Intuitive Schreibweise entspricht der Reihenfolge im Speicher, z. B. die Dezimalzahl 256 in hexadezimaler Schreibweise als 0x0100.
2. Little Endian: „Niederwertigstes Byte steht an niederwertigster Adresse“
Kontraintuitiv, da die Daten im Speicher „verkehrt herum“ abgelegt werden, z. B. die Dezimalzahl 256 in hexadezimaler Schreibweise als 0x0001.

x86-kompatible Computer verwenden intern Little Endian. Bei der Kommunikation mit anderen Computern stellt das aber möglicherweise ein Problem dar. Deswegen: Konvertierung in Network Byte Order.

Network Byte Order

Vor dem Versenden von Daten müssen Daten aus der **Host Byte Order** (Little Endian bei x86) in **Network Byte Order** (Big Endian) konvertiert werden. Analoges gilt beim Empfang von Daten.

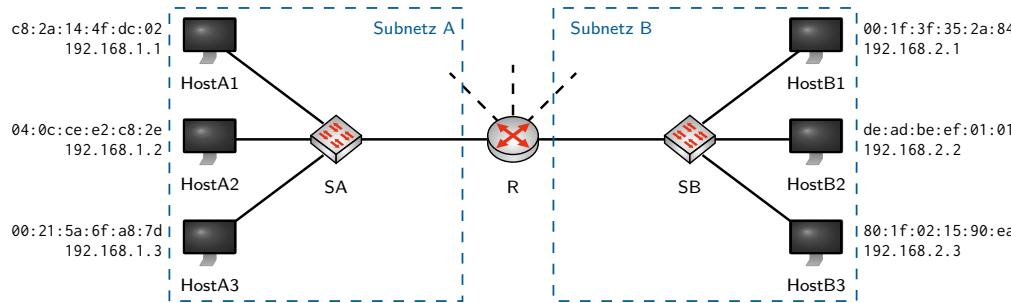
- ▶ Dies betrifft z. B. die Felder Total Length, Identification, Flags+Fragment Offset und Header Checksum aus dem IP-Header.
- ▶ Nicht betroffen sind 1 B lange Felder. Quell- und Zieladresse liegen gewöhnlich als Array von 1 B langen Werten vor, weswegen auch diese kein Problem darstellen.

Die Konvertierung zwischen beiden Formaten erfolgt für 16 bit lange Werte in C beispielsweise durch die Funktionen htons() und ntohs():

- ▶ htons(0x0001) = 0x0100 „Host to Network Short“
- ▶ ntohs(0x0100) = 0x0001 „Network to Host Short“

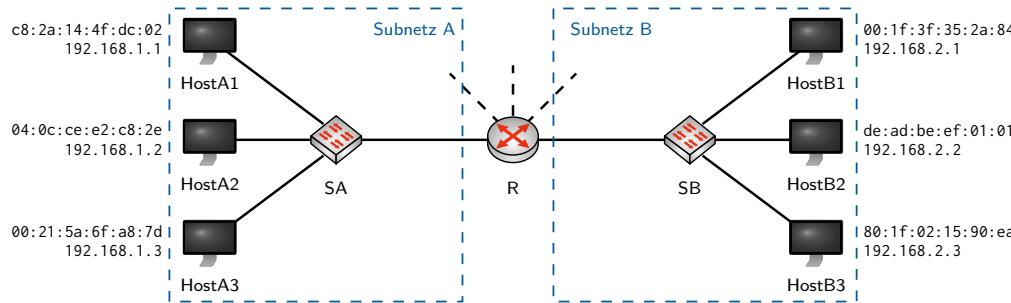
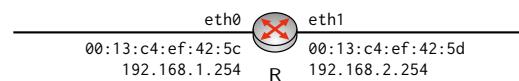
Für 32 bit lange Werte gibt es analog htonl() und ntohl() („l“ für den Datentyp long).

Zurück zu unserem Beispiel:



- ▶ Innerhalb von Netz A erreichen sich alle Hosts weiterhin direkt über SA.
- ▶ Will HostA1 nun ein Paket an HostB2 schicken, so geht das nicht mehr direkt:
 - ▶ Die Weiterleitung von A nach B erfolgt über R **auf Basis von IP-Adressen**
 - ▶ HostA1 muss also dafür sorgen, dass R das Paket erhält
 - ▶ Dazu trägt HostA1 als Ziel-MAC-Adresse die Adresse des lokalen Interfaces von R ein
 - ▶ Als Ziel-IP-Adresse wird die IP-Adresse von HostB2 verwendet

⇒ R muss adressierbar sein und verfügt daher auf jedem Interface über eine eigene MAC-Adresse und (wie wir gleich sehen werden) auch über eine eigene IP-Adresse:

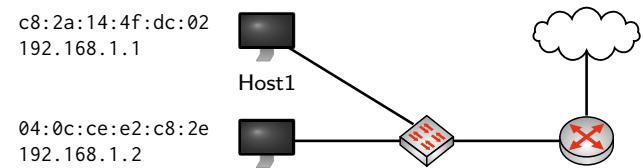


Offene Probleme:

- ▶ Angenommen der Sender kennt nur die IP-Adresse des Ziels. Wie kann die MAC-Adresse des Next-Hops bestimmt werden? ([Adressauflösung](#))
- ▶ Woher weiß HostA1, dass er Netz B über R erreichen kann? ([Routing Table](#))
- ▶ Angenommen das Ziel eines Pakets befindet sich nicht in einem direkt an R angeschlossenen Netz. Woher kennt R die richtige Richtung? ([Routing Table](#))
- ▶ Wie wird diese „Routing Table“ erzeugt? ([statisches Routing, Routing Protokolle](#))
- ▶ Was ist, wenn R mehrere Wege zu einem Ziel kennt? ([Longest Prefix Matching; Border Gateway Protocol Path Selection](#))
- ▶ Wie funktioniert die Unterteilung einer IP-Adresse in Netz- und Hostanteil? ([Classful Routing, Classless Routing, Subnetting](#))
- ▶ Woher kennt der Sender überhaupt die IP des Ziels? ([DNS → Schicht 7](#))

Adressauflösung [14]

- ▶ Host1 will eine Nachricht an Host2 senden
- ▶ Die IP-Adresse von Host2 (192.168.1.2) sei ihm bereits bekannt
- ▶ Wie erhält Host1 die zugehörige MAC-Adresse?



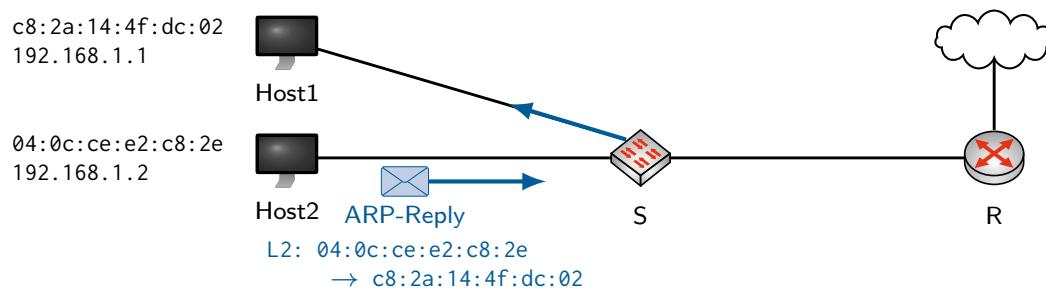
Address Resolution Protocol (ARP)

1. Host1 sendet einen ARP Request: „Who has 192.168.1.2? Tell 192.168.1.1“
2. Host2 antwortet mit einem ARP Reply: „192.168.1.2 is at 04:0c:ce:e2:c8:2e“

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																
0 B	Hardware Type																Protocol Type																															
4 B	Hardware Addr. Length				Protocol Addr. Length				Operation																																							
8 B	Sender Hardware Address (first 32 bit)																																															
12 B	Sender Hardware Address (last 16 bit)																Sender Protocol Address (first 16 bit)																															
16 B	Sender Protocol Address (last 16 bit)																Target Hardware Address (first 16 bit)																															
20 B	Target Hardware Address (last 32 bit)																																															
24 B	Target Protocol Address																																															

Abbildung: ARP-Paket für IPv4 über Ethernet

Beispiel:



Hinweis: L2: xx:xx:xx:xx:xx:xx → yy:yy:yy:yy:yy:yy stellt Absender- und Ziel-MAC auf Schicht 2 dar.

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0 B	0x0001 (Ethernet)																0x8000 (IPv4)															
4 B	0x06					0x04				0x0001 (Request)																						
8 B	0xc82a144f																															
12 B	0xdC02 (Sender Hardware Address)									0xc80a8							0x040ccce2															
16 B	0x0101 (Sender Protocol Address)									0x0000							0xc82a															
20 B	0x00000000 (Target Hardware Address)																0x144fdc02															
24 B	0xc0a80102 (Target Protocol Address)																0xc0a80101															

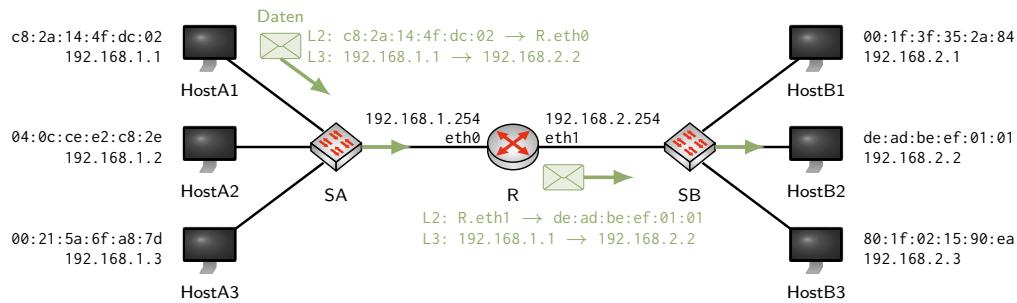
(a) ARP Request

(b) ARP Reply

- ▶ Der ARP-Request wird an die MAC-Broadcastadresse ff:ff:ff:ff:ff:ff geschickt, weswegen der Switch S den Rahmen an alle angeschlossenen Hosts weiterleitet.
- ▶ Der ARP-Reply wird als MAC-Unicast versendet (adressiert an Host1).
- ▶ Die Rollen Sender / Target sind zwischen Request und Reply vertauscht (vgl. Inhalte der grünen und roten Felder).

Was ist nun, wenn das Ziel **nicht** im selben Netz liegt (z. B. HostA1 an HostB2)?

- ▶ Jeder Host sollte ein **Default Gateway** kennen, an das er alle Pakete schickt, deren Zieladressen nicht im eigenen Netz liegen.
- ▶ Die Zugehörigkeit einer Adresse zum jeweiligen Netz erkennt ein Host durch Vergleich der eigenen Adresse mit der Zieladresse.
- ▶ Im Moment gehen wir noch davon aus, dass die ersten 3 Oktette einer IP-Adresse das Netz identifizieren
⇒ 192.168.1.1 und 192.168.2.2 liegen in unterschiedlichen Netzen.



1. HostA1 erkennt, dass 192.168.2.2 nicht im eigenen Netz liegt. Sein Default-Gateway ist 192.168.1.254.
2. HostA1 löst die MAC-Adresse zu 192.168.1.254 auf.
3. HostA1 sendet das Datenpaket an R: Dabei adressiert er R mittels der eben bestimmten MAC-Adresse (Schicht 2). Als Ziel-IP-Adresse (Schicht 3) verwendet er die IP-Adresse von HostB2.
4. R akzeptiert das Paket, bestimmt das ausgehende Interface und leitet das Paket weiter an HostB2. Dabei adressiert R wiederum HostB2 anhand seiner MAC-Adresse (erfordert ggf. einen weiteren ARP-Schritt).

Merkel:

- ▶ MAC-Adressen dienen zur Adressierung **innerhalb** eines (Direktverbindungs-)Netzes und werden beim Routing verändert.
- ▶ IP-Adressen dienen der End-zu-End-Adressierung **zwischen** mehreren (Direktverbindungs-)Netzen und werden beim Routing nicht verändert.

Anmerkungen

- ▶ Das Ergebnis einer Adressauflösung wird i. d. R. im **ARP-Cache** eines Hosts zwischengespeichert, um nicht bei jedem zu versendenden Paket erneut eine Adressauflösung durchführen zu müssen.
- ▶ Die Einträge im ARP-Cache altern und werden nach einer vom Betriebssystem festgelegten Zeit invalidiert (5-10 Minuten).
- ▶ Den Inhalt des ARP-Caches kann man sich unter Linux, OS X und Windows mittels des Befehls arp -a anzeigen lassen.
- ▶ ARP-Replies können auch als MAC-Broadcast verschickt werden, so dass alle Hosts innerhalb einer Broadcast-Domain den Reply erhalten. Abhängig vom Betriebssystem werden derartige „unaufgeforderten ARP-Replies“ (engl. **unsolicited ARP replies**) häufig ebenfalls im ARP-Cache gespeichert.

Fragen: Was würde passieren, wenn ...

- ▶ zwei Hosts innerhalb derselben Broadcast-Domain identische MAC-Adressen aber unterschiedliche IP-Adressen haben?
- ▶ ein Host absichtlich auf ARP-Requests antwortet, die nicht an ihn gerichtet waren?
- ▶ ein Host unsinnige ARP-Replies via MAC-Broadcasts verschickt?

Internet Control Message Protocol (ICMP) [15]

Das IP-Protokoll unterstützt das Senden von Paketen an Ziel-Hosts. Dabei kann es zu Fehlern kommen, z. B.

- ▶ ein Paket gerät in eine Routing-Schleife,
- ▶ ein Router kennt keinen Weg zum Zielnetz,
- ▶ der letzte Router zum Ziel kann die MAC-Adresse des Empfängers nicht auflösen ...

Das [Internet Control Message Protocol \(ICMP\)](#) dient dazu,

- ▶ in derartigen Fällen den Absender über das Problem zu benachrichtigen und
- ▶ stellt zusätzlich Möglichkeiten bereit, um z. B.
 - ▶ die Erreichbarkeit von Hosts zu prüfen („Ping“) oder
 - ▶ Pakete umzuleiten ([Redirect](#)).

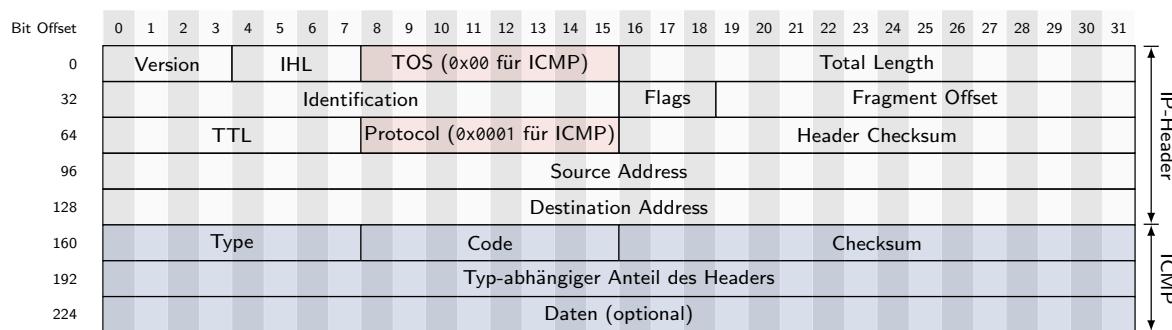


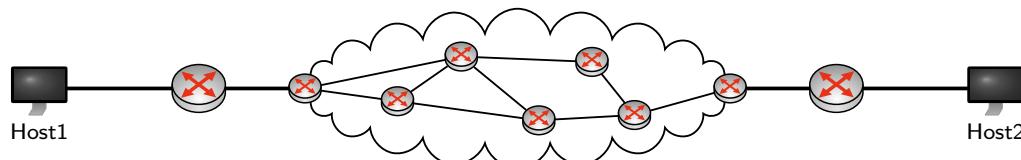
Abbildung: Allgemeiner ICMP-Header mit vorangestelltem IP-Header

ICMP Echo Request / Echo Reply

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	Type = 0x08	Code = 0x00	Checksum	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	Type = 0x00	Code = 0x00	Checksum
	Identifier		Sequence Number		Identifier		Sequence Number
	Daten (optional)				Daten (optional)		

(a) ICMP Echo Request

(b) ICMP Echo Reply



„Ping“ von Host1 nach Host2:

- ▶ Host1 wählt einen zufälligen Identifier (16 bit Wert), die Sequenznummer wird für jeden gesendeten Echo-Request um eins inkrementiert.
- ▶ Der Echo Request wird von Routern wie jedes IP-Paket weitergeleitet.
- ▶ Erhält Host2 den Echo Request, so antwortet er mit einem Echo Reply. Dabei werden Identifier, Sequenznummer und Daten aus dem Request kopiert und zurückgeschickt.
- ▶ Sollte die Weiterleitung zu Host2 fehlschlagen, so wird eine ICMP-Nachricht mit dem entsprechenden Fehlercode an Host1 zurückgeschickt.

Frage: Wozu dient der Identifier?

ICMP Time Exceeded

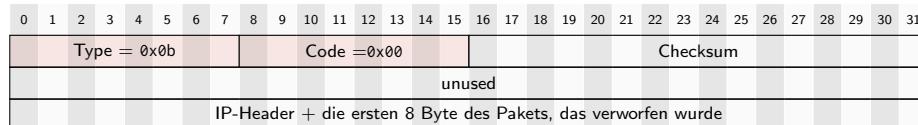
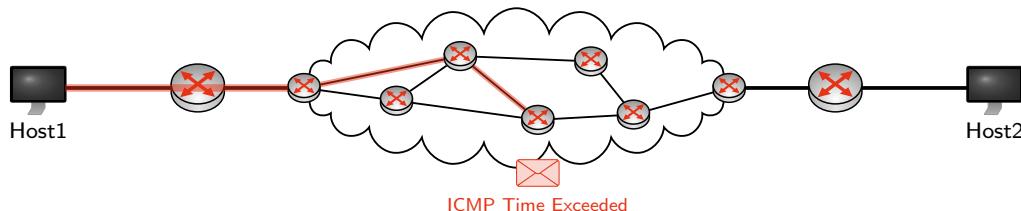


Abbildung: ICMP TTL-Exceeded

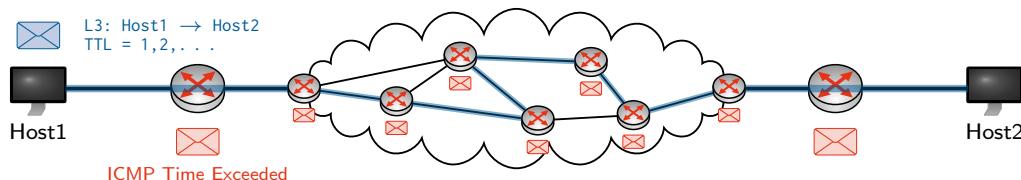


- ▶ Der IP-Header besitzt ein TTL-Feld, welches bei der Weiterleitung eines Pakets durch den jeweiligen Router um eins dekrementiert wird.
- ▶ Erreicht es den Wert 0, so wird das betreffende Paket verworfen.
- ▶ Der Router generiert ein ICMP Time Exceeded und schickt es an den Absender des verworfenen Pakets zurück.

Da der Header und die ersten 8 B des verworfenen Pakets an den Absender zurückgeschickt werden, kann dieser genau bestimmen, welches Paket verworfen wurde.

Frage: Welche Informationen sind in diesen ersten 8 B enthalten, wenn das verworfene Paket ein ICMP Echo Request war?

ICMP Traceroute



- ▶ Obwohl Pakete zwischen Host1 und Host2 (in Hin- und Rückrichtung) jeweils unterschiedliche Wege nehmen können, werden in der Praxis meist nur einer oder sehr wenige genutzt.
- ▶ Welchen Pfad nehmen Pakete von Host1 nach Host2?

Traceroute: Host1 sendet z. B. ICMP Echo Requests an Host2

- ▶ wobei das TTL-Feld zu Beginn auf 1 gesetzt wird und
- ▶ danach schrittweise um jeweils 1 erhöht wird.

⇒ Router entlang des Pfads von Host1 nach Host2 werden schrittweise die Pakete verwerfen und jeweils ein TTL Exceeded an Host1 zurücksenden. Anhand der IP-Quelladresse dieser Fehlernachrichten kann Host1 den Pfad hin zu Host2 nachvollziehen.

⇒ 3. Programmieraufgabe: „Implementiere Traceroute“

Beispiel:

```
slacky: moepi$ traceroute -n www.net.in.tum.de
traceroute to www.net.in.tum.de (131.159.15.49), 64 hops max, 52 byte packets
 1  192.168.1.1  2.570 ms  1.808 ms  2.396 ms
 2  82.135.16.28  15.036 ms  14.359 ms  13.760 ms
 3  212.18.7.189  14.118 ms  13.801 ms  13.845 ms
 4  82.135.16.102  20.062 ms  20.137 ms  20.251 ms
 5  80.81.192.222  22.707 ms  23.049 ms  23.215 ms
 6  188.1.144.142  31.068 ms  36.542 ms  30.823 ms
 7  188.1.37.90  30.815 ms  30.671 ms  30.808 ms
 8  129.187.0.150  30.272 ms  30.602 ms  30.845 ms
 9  131.159.252.1  30.885 ms  30.551 ms  30.992 ms
10  131.159.252.150  30.886 ms  30.955 ms  30.621 ms
11  131.159.252.150  30.578 ms  30.699 ms *
```

Fragen / Probleme:

- ▶ Ein Router hat mehrere IP-Adressen. Welche wählt er als Absender-Adresse der Fehlerbenachrichtigungen? Wählt er immer dieselbe Adresse?
- ▶ Was ist, wenn es tatsächlich mehrere gleichzeitig genutzte Pfade oder Pfadabschnitte gibt?
- ▶ Müssen Router überhaupt Fehlerbenachrichtigungen versenden?
- ▶ Angenommen der Pfad von A → B ist symmetrisch. Warum werden sich die Ausgaben von Traceroute dennoch unterscheiden?

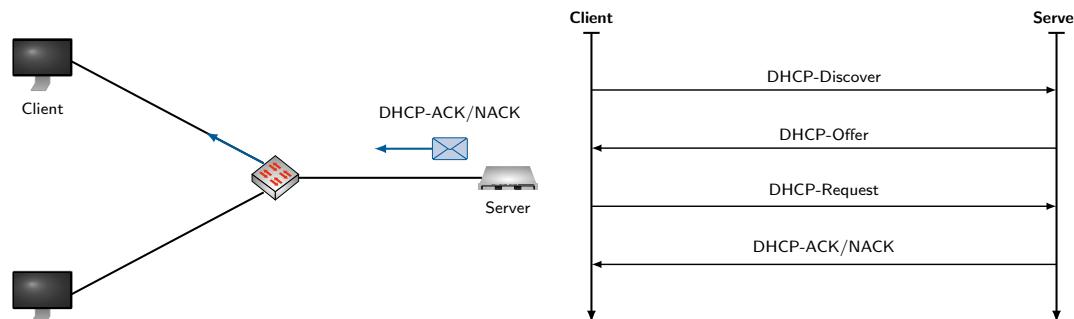
Dynamic Host Configuration Protocol (DHCP) [15]

Woher bekommen Hosts eigentlich ihre IP-Adresse?

- ▶ Statische Konfiguration von Hand
- ▶ Dynamisch von einem **DHCP-Server** zugewiesene IP-Adresse

Ablauf:

1. Client sendet DHCP-Discover (Layer 2 Broadcast)
2. DHCP-Server antwortet mit DHCP-Offer, wodurch er dem Client eine IP-Adresse anbietet
3. Client antwortet mit DHCP-Request, wodurch er die angebotene Adresse anfordert
4. DHCP-Server antwortet mit DHCP-ACK, wodurch er die angeforderte Adresse zur Nutzung freigibt, oder mit DHCP-NACK, wodurch er die Nutzung der Adresse untersagt



Anmerkungen

- ▶ Die vom DHCP-Server zugewiesene Adresse wird auch als **Lease** bezeichnet und ist in ihrer Gültigkeit zeitlich begrenzt
- ▶ Clients erneuern ihr Lease in regelmäßigen Abständen beim DHCP-Server.
- ▶ Gerade in kleineren (privaten) Netzwerken übernimmt häufig ein Router die Rolle des DHCP-Servers.
- ▶ Zusammen mit der IP-Adresse und Subnetzmaske können DHCP-Server viele weitere Optionen ausliefern:
 - ▶ DNS-Resolver zur Namensauflösung (→ Kapitel 5)
 - ▶ Hostname und Domänen-Suffix (→ Kapitel 5)
 - ▶ Statische Routen, insbesondere einen Default-Gateway
 - ▶ Maximum Transmission Unit
 - ▶ NTP-Server zur Zeitsynchronisation
 - ▶ ...
- ▶ Aus Redundanzgründen werden manchmal mehrere DHCP-Server pro Netzwerk verwendet, wobei
 - ▶ sich die Adressbereiche der beiden Server nicht überschneiden dürfen oder
 - ▶ zusätzliche Mechanismen zur Synchronisation der Adresspools notwendig sind.
- ▶ Ein versehentlich ins Netzwerk eingebrachter DHCP-Server (z. B. durch einen achtlos angeschlossenen Router oder Access Point) kann beträchtliche Auswirkungen auf das Netzwerk haben und ist häufig schwer zu finden:
 - ▶ Im Netz eines Lehrstuhls gab es einst einen solchen Rogue-DHCP-Server,
 - ▶ der trotz intensiver Suche nicht gefunden werden konnte.
 - ▶ Über die MAC-Adressen der vom Server stammenden DHCP-Nachrichten stellte sich am Ende heraus, dass es sich um ein **Nokia-Smartphone** im WLAN handelte!
 - ▶ Die Anzahl der Verdächtigen hatte sich damit stark eingeschränkt (;

Adressklassen (Classful Routing)

Zu Beginn des Kapitels hatten wir in einem Beispiel beschrieben, dass

- ▶ die ersten drei Oktette einer IP-Adresse das Netz identifizieren und
- ▶ das vierte Oktett Hosts innerhalb eines Netzes adressiert.

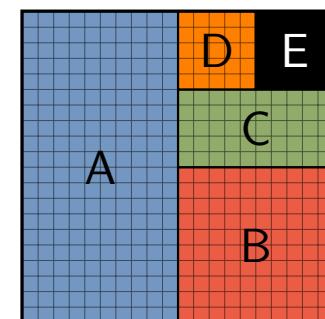
Diese Aufteilung war lediglich ein Beispiel, entsprechend der Adress-Klasse C. Historisch ist der IP-Adressraum in folgende fünf **Klassen** unterteilt:

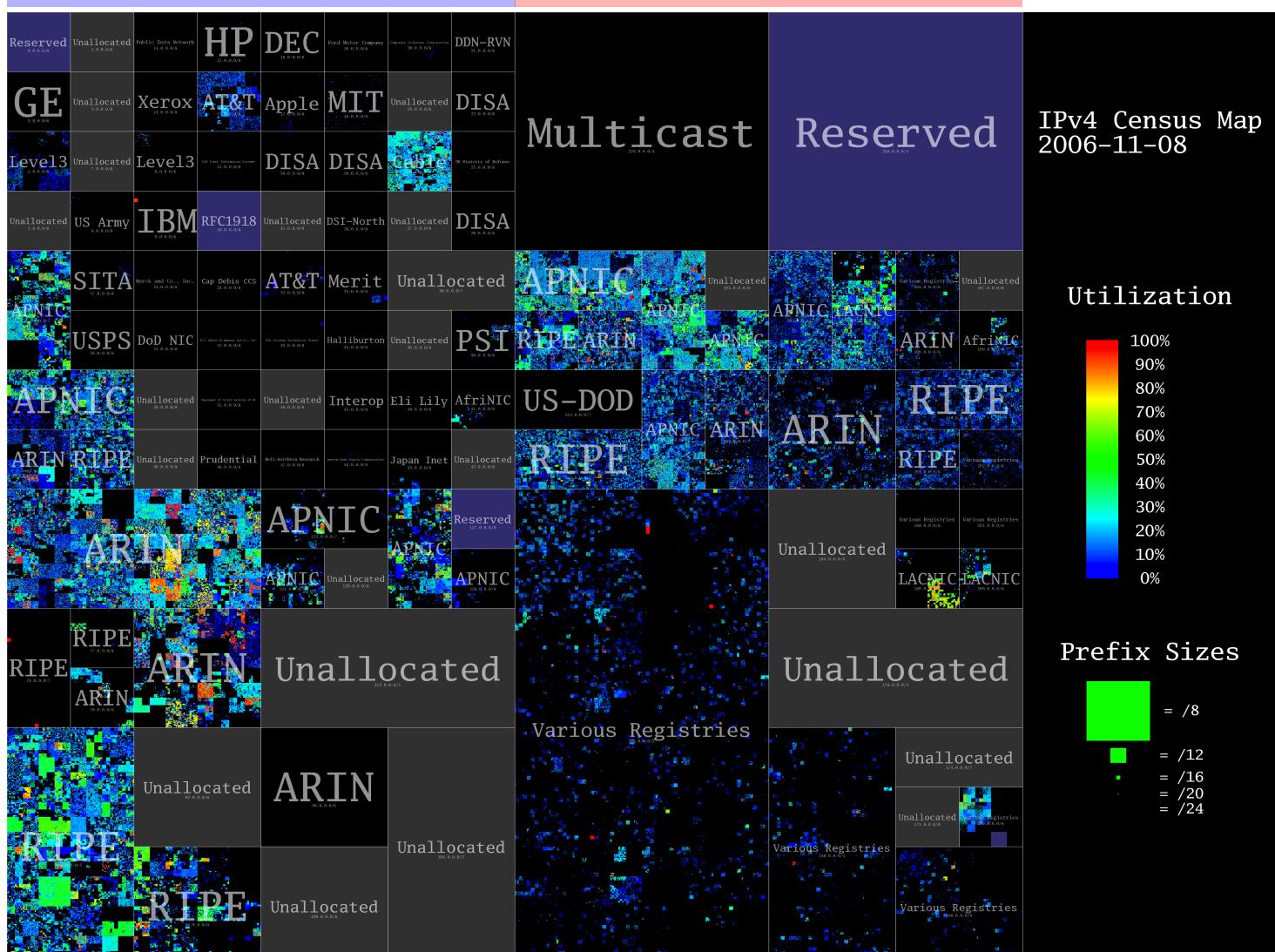
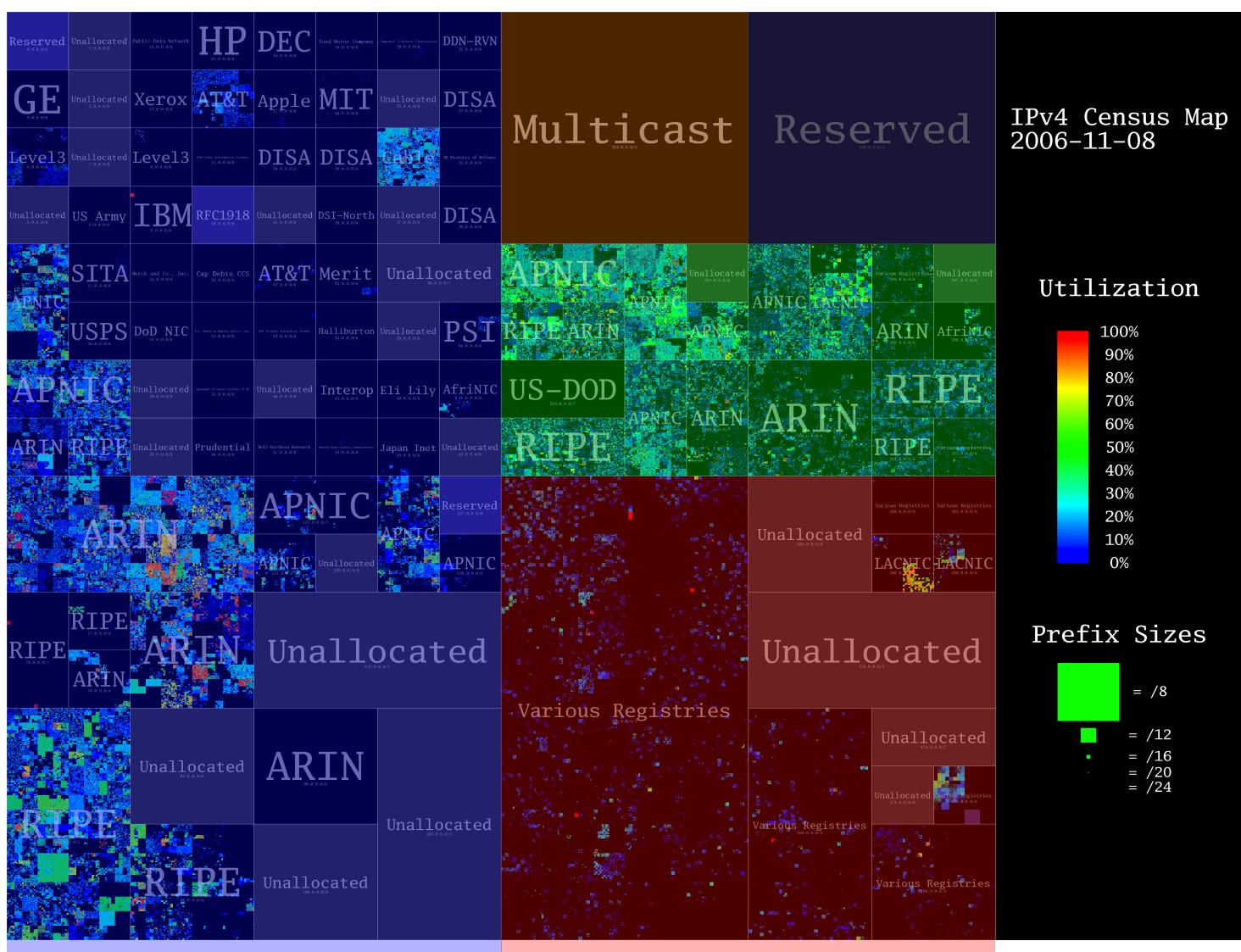
Klasse	1. Oktett	Erste Adresse	Letzte Adresse	Netz-/Hostanteil	Anzahl Netze	Adressen pro Netz
A	0xxxxxxx	0.0.0.0	127.255.255.255	N.H.H.H	$2^7 = 128$	$2^{24} = 16777216$
B	10xxxxxx	128.0.0.0	191.255.255.255	N.N.H.H	$2^{14} = 16384$	$2^{16} = 65536$
C	110xxxxx	192.0.0.0	223.255.255.255	N.N.N.H	$2^{21} = 2097152$	$2^8 = 256$
D	1110xxxx	224.0.0.0	239.255.255.255		reserviert für Multicast	
E	1111xxxx	240.0.0.0	255.255.255.255		reserviert	

Grafische Darstellung:

- ▶ IP-Adressraum als Quadrat
- ▶ Farbige Flächen markieren die fünf Adressklassen
- ▶ Die einzelnen Klasse-A-Netze sind als Drahtgitter angedeutet

→ Nächste Folie zeigt die tatsächliche **Verteilung und Ausnutzung** des Adressraums [1] (selbe Farbkodierung).





Verschwendung des Adressraums

- ▶ Als IPv4 1981 eingeführt wurde, konnte man sich nicht vorstellen, dass $\sim 2^{32}$ Adressen aufgeteilt in die Klassen A, B und C nicht ausreichend würden.
- ▶ Es wurden große Adresseblöcke an Firmen, Behörden und Bildungseinrichtungen vergeben, z. B. ganze Klasse-A Netze an HP, IBM, AT&T, Apple, MIT, Generel Electric, US Army, ...

Folge:

- ▶ Ineffiziente Aufteilung und Nutzung des Adressraums
- ▶ Große Netze mit internen Routern \Rightarrow weitere Unterteilung in **Subnetze** notwendig

Situation heute:

- ▶ Vergabe des letzten IPv4 Adressblocks am 3.2.2011 durch die **IANA**¹ an eine der fünf **Regional Internet Registries (RIRs)**, das **APNIC**²
- ▶ IPv4-Adressraum praktisch aufgebraucht
- ▶ Immer noch schleppende Einführung von IPv6

¹ Internet Assigned Numbers Authority

² Asia-Pacific Network Information Center (APNIC)

Subnetting (Classless Routing)

Bereits 1993 wurde mit **CIDR**³ ein Verfahren zur Unterteilung von IP-Netzen eingeführt:

- ▶ Zusätzlich zur IP-Adresse erhält ein Interface eine ebenfalls 32 bit lange **Subnetzmaske**
- ▶ Die Subnetzmaske unterteilt die IP-Adresse in einen **Netzanteil** und einen **Hostanteil**
- ▶ Eine logische 1 in der Subnetzmaske bedeutet Netzanteil, eine logische 0 Hostanteil
- ▶ UND-Verknüpfung von IP-Adresse und Subnetzmaske ergibt die Netzadresse
- ▶ Die übliche Klassenzugehörigkeit hat damit nur noch im Sprachgebrauch eine Bedeutung

Beispiel 1:

IP-Adresse	11000000 . 10101000 . 00000000 . 10110010	192.168.0.178
Subnetz Maske	11111111 . 11111111 . 11111111 . 00000000	255.255.255.0
Netzadresse	11000000 . 10101000 . 00000000 . 00000000	192.168.0.0
Broadcastadresse	11000000 . 10101000 . 00000000 . 11111111	192.168.0.255

- ▶ 24 bit Netzanteil, 8 bit Hostanteil $\Rightarrow 2^8 = 256$ Adressen
- ▶ Netzadresse: 192.168.0.0
- ▶ Broadcastadresse: 192.168.0.255
- ▶ Nutzbare Adressen für Hosts: $2^8 - 2 = 254$

³ Classless Inter-Domain Routing

Beispiel 2:

IP-Adresse	11000000 . 10101000 . 00000000 . 10110010		192.168.0.178
Subnetz Maske	11111111 . 11111111 . 11111111 . 10000000		255.255.255.128
		Netz	Host
Netzadresse	11000000 . 10101000 . 00000000 . 10000000		192.168.0.128
Broadcastadresse	11000000 . 10101000 . 00000000 . 11111111		192.168.0.255

- ▶ 25 bit Netzanteil, 7 bit Hostanteil $\Rightarrow 2^7 = 128$ Adressen
- ▶ Netzadresse: 192.168.0.128
- ▶ Broadcastadresse: 192.168.0.255
- ▶ Nutzbare Adressen für Hosts: $2^7 - 2 = 126$

Beispiel 3:

IP-Adresse	11000000 . 10101000 . 00000000 . 10110010		192.168.0.178
Subnetz Maske	11111111 . 11111111 . 11111111 . 11000000		255.255.255.192
		Netz	Host
Netzadresse	11000000 . 10101000 . 00000000 . 10000000		192.168.0.128
Broadcastadresse	11000000 . 10101000 . 00000000 . 10111111		192.168.0.191

- ▶ 26 bit Netzanteil, 6 bit Hostanteil $\Rightarrow 2^6 = 64$ Adressen
- ▶ Netzadresse: 192.168.0.128
- ▶ Broadcastadresse: 192.168.0.191
- ▶ Nutzbare Adressen für Hosts: $2^6 - 2 = 62$

Supernetting

Dasselbe Prinzip funktioniert auch in die andere Richtung:

- ▶ Zusammenfassung mehrerer **zusammenhängender** kleinerer Netze zu einem größeren Netz
- ▶ Wird häufig von Routern angewendet, um die Anzahl der Einträge in der Routingtabelle zu reduzieren

Beispiel:

IP-Adresse	11000000 . 10101000 . 00000000 . 10110010		192.168.0.178
Subnetz Maske	11111111 . 11111111 . 11111110 . 00000000		255.255.254.0
		Netz	Host
Netzadresse	11000000 . 10101000 . 00000000 . 00000000		192.168.0.0
Broadcastadresse	11000000 . 10101000 . 00000001 . 11111111		192.168.1.255

- ▶ 23 bit Netzanteil, 9 bit Hostanteil $\Rightarrow 2^9 = 512$ Adressen
- ▶ Netzadresse: 192.168.0.0
- ▶ Broadcastadresse: 192.168.1.255
- ▶ Nutzbare Adressen für Hosts: $2^9 - 2 = 510$

Präfixschreibweise:

Anstelle die Subnetzmaske auszuschreiben, wird häufig nur die Länge des Netzanteils (Anzahl führender Einsen in der Subnetzmaske) angegeben, z. B. 192.168.0.0/23.

Frage: Können die beiden Netze 192.168.1.0/24 und 192.168.2.0/24 zu einem größeren Netz zusammengefasst werden?

Antwort: Nein, denn es gibt keine passende Subnetzmaske:

- ▶ 192.168.0.0/23 enthält die Netze 192.168.{0,1}.0/24
- ▶ 192.168.2.0/23 enthält die Netze 192.168.{2,3}.0/24
- ▶ 192.168.1.0/23 ist **keine gültige** Netzadresse, denn UND-Verknüpfung der Adresse 192.168.1.0 mit der Subnetzmaske 255.255.254.0 ergibt 192.168.0.0 als Netzadresse!
⇒ 192.168.1.0 ist eine Hostadresse im Netz 192.168.0.0/23.

Frage: Können die beiden Netze 192.168.0.0/24 und 192.168.2.0/24 zu einem größeren Netz zusammengefasst werden?

Antwort: Nein, denn die beiden Netze sind nicht benachbart:

- ▶ Das Netz 192.168.0.0/22 würde zwar beide Subnetze enthalten, zusätzlich aber auch die beiden Netze 192.168.{1,3}.0/24.

Frage: Können die vier Netze 192.168.{4,5,6,7}.0/24 zu einem größeren Netz zusammengefasst werden?

Antwort: Ja, das Netz 192.168.4.0/22 umfasst genau diese vier Netze.

Besondere Adressbereiche

Einige Adressbereiche sind für bestimmte Zwecke reserviert:

1. 0.0.0.0 / 8: Hosts in diesem Netzwerk
 - ▶ Verwendung z.B. als Quelladresse bei DHCP, wenn ein Client noch keine IP-Adresse besitzt.
 - ▶ Serveranwendungen erwarten eingehende Verbindungen auf der Adresse 0.0.0.0, was soviel bedeutet wie „jede Adresse, die verfügbar ist“.
 - ▶ Andere Adressen innerhalb dieses Bereichs dürfen als Zieladresse für bestimmte Hosts innerhalb des lokalen Netzes verwendet werden.
 - ▶ Adressen aus diesem Bereich werden grundsätzlich nicht geroutet.
2. 127.0.0.0 / 8: „Loopback-Adressen“
 - ▶ Adressen in diesem Bereich identifizieren den lokalen Rechner (Localhost), z.B. 127.0.0.1.
 - ▶ Diese Adressen werden grundsätzlich nicht geroutet.
 - ▶ Pakete mit dieser Zieladresse würden niemals gesendet sondern vor dem Sendevorgang „geloopt“.
3. 10.0.0.0 / 8, 172.16.0.0 / 12, 192.168.0.0 / 16: Private Adressbereiche
 - ▶ Dürfen innerhalb lokaler Netzwerke ohne Registrierung durch die IANA verwendet werden.
 - ▶ Adressen innerhalb dieser Bereiche dürfen zwischen privaten Netzen geroutet werden.
 - ▶ Mehrfache Vergabe in unterschiedlichen privaten Netzen möglich.
 - ▶ Dürfen nicht in öffentliche Netze geroutet werden.
4. 169.254.0.0 / 16: Automatic Private IP Addressing (APIPA)
 - ▶ Block zur automatischen Adressvergabe (APIPA).
 - ▶ Routing wie bei den privaten Adressbereichen.
5. 255.255.255.255 / 32: Global Broadcast
 - ▶ Identifiziert im Prinzip **alle** Hosts.
 - ▶ Wird niemals geroutet.

Übersicht

Motivation

Vermittlungsarten

Leitungsvermittlung
Nachrichtenvermittlung
Paketvermittlung

Adressierung im Internet

Internet Protocol version 4 (IPv4)
Internet Protocol version 6 (IPv6)

Wegwahl (Routing)

Routing Table und Longest Prefix Matching
Dynamisches Routing
Routing Information Protocol (RIP)
Autonome Systeme

Internet Protocol version 6 (IPv6)

IPv6 wurde Ende 1995 [5] als Nachfolger für IPv4 vorgeschlagen und mit RFC 2460 [6] 1998 standardisiert.

- ▶ Damit dürfte es älter als einige Vorlesungsteilnehmer sein ...
- ▶ ... und hat IPv4 noch lange nicht obsolet gemacht.

Hauptgrund dafür ist, dass

- ▶ in erster Linie die Adressknappheit treibende Kraft für die Verbreitung von IPv6 ist und
- ▶ diese erst in den vergangenen Jahren zum Problem wurde.

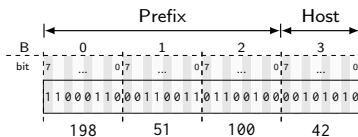
Die wesentlichen Änderungen gegenüber IPv4 umfassen:

- ▶ Vergrößerung des Adressraums von 2^{32} auf 2^{128} .³
- ▶ Vereinfachung des Headerformats (effizientere Verarbeitung auf Routern).
- ▶ Änderungen bei der IP-Fragmentierung.
- ▶ Flexibilität durch sog. **Extension Header** bei gleichzeitiger Vereinfachung des Headerformats.
- ▶ **Stateless Address Autoconfiguration (SLAAC)** mittels ICMPv6.
- ▶ Möglichkeit für **Stateful Autoconfiguration** durch DHCPv6.
- ▶ Nativer Einsatz von **Multicast**, beispielsweise um alle Router in einem Segment zu adressieren.

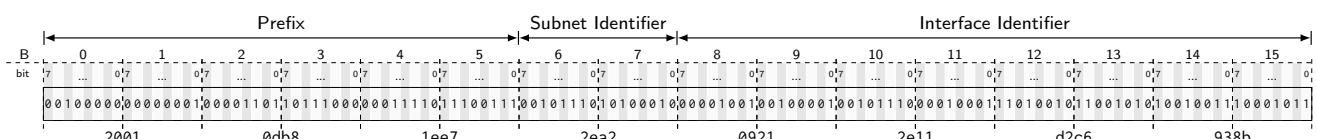
³ $2^{128} = 340.282.366.920.938.463.463.374.607.431.768.211.456$ Adressen

Adressformat

- ▶ IPv4-Adressen werden üblicherweise in **Dot-Decimal-Notation** dargestellt, d.h. je eine 8 bit lange Gruppe wird als Dezimalzahl im Bereich 0–255 getrennt durch Punkte notiert.
- ▶ Diese Schreibweise würde bei IPv6 insgesamt 16 Gruppen ergeben – und merken könnte sich das auch niemand mehr.



- ▶ Stattdessen werden IPv6-Adressen in Gruppen zu je 16 bit getrennt durch Doppelpunkte (**colon-separated**) in hexadezimaler Schreibweise dargestellt.



Achtung:

- ▶ Bitte IPv6-Adressen nicht mit MAC-Adressen verwechseln, nur weil beide hexadezimal notiert werden.
- ▶ Bitte das Präfix, das es auch bei IPv4 gibt, und den Subnet Identifier auseinander halten.

Hinweise zur Notation [9]⁴

Wir betrachten die folgende Adresse:

2001:0db8:0000:0000:0001:0000:0000:0001

1. Führende Nullen in den einzelnen Blöcken werden weggelassen:

✓ 2001:db8:0:0:1:0:0:1

2. Höchstens eine Gruppe konsekutiver Blöcke, die nur aus Nullen bestehen, darf wie folgt abgekürzt werden:

✓ 2001:db8::1:0:0:1

3. Gibt es mehrere Möglichkeiten für Fall 2), so wählt man die längste Sequenz von Nullen. Bei mehreren gleich lange Sequenzen wählt man die erste Möglichkeit. **Falsch** wäre also:

✗ 2001:db8:0:0:1::1
✗ 2001:db8::1::1

4. Ein einzelner 0 Block darf **nicht** mit :: abgekürzt werden:

✓ 2001:db8:0:1:1:1:1:1
✗ 2001:db8::1:1:1:1

Anderes Beispiel: Die Loopback-Adresse (vgl. 127.0.0.1 bei IPv4) lässt sich wie folgt kürzen:

0000:0000:0000:0000:0000:0000:0001/128 → ::1/128

⁴ Weitere Details finden sich in RFC 5952.

Headerformat

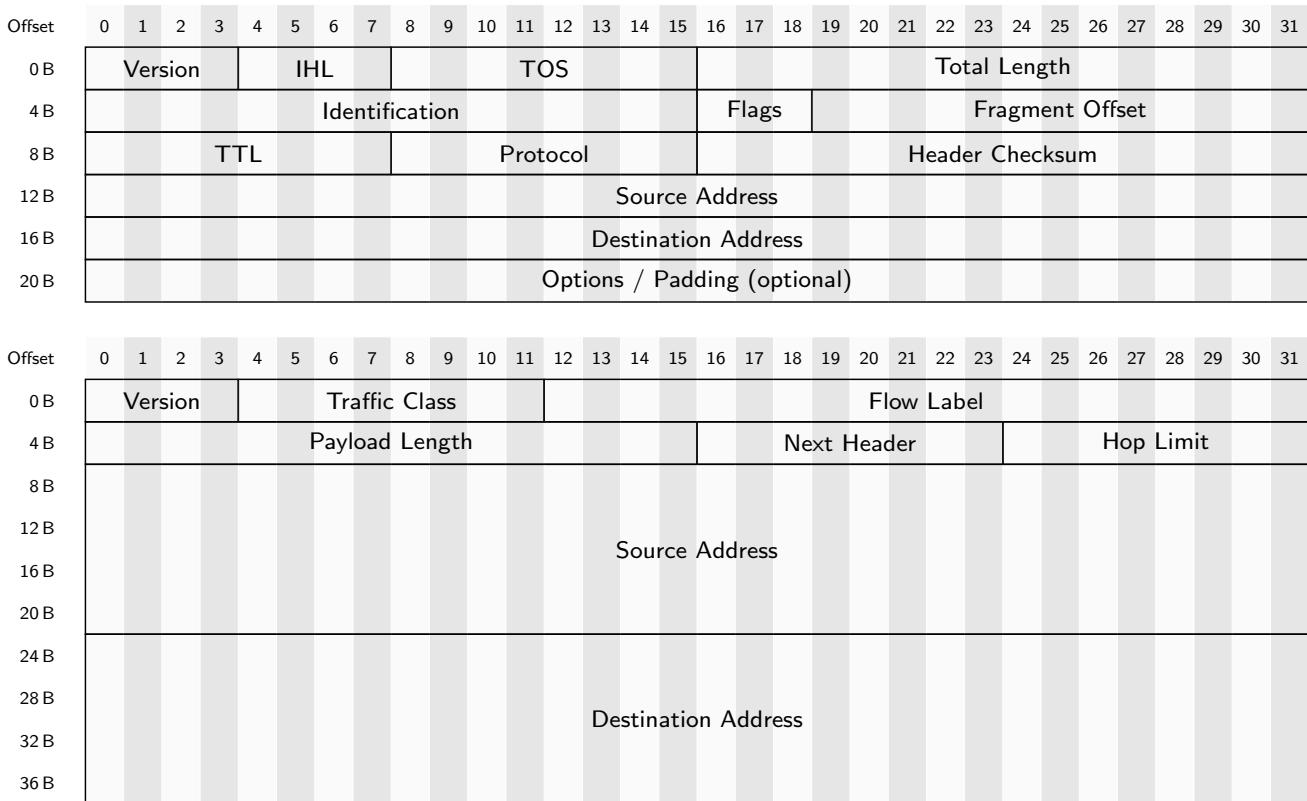


Abbildung: IPv4-Header (oben) und IPv6-Header (unten) im Vergleich

Version

- ▶ Gibt die verwendete IP-Version an.
- ▶ Gültige Werte sind 4 (IPv4) und 6 (IPv6).

Traffic Class

- ▶ Äquivalent zum TOS-Feld des IPv4-Headers.
- ▶ Wird zur Verkehrsriorisierung / Quality of Service (QoS) verwendet.

Flow Label

- ▶ Ursprünglich vorgesehen für Echtzeitanwendungen.
- ▶ Wird heute in erster Linie von Routern verwendet, um zusammengehörende Pakete (**Flows**) auf Schicht 3 zu erkennen.
- ▶ Pakete, die zum selben Flow gehören sollen ggf. gleich behandelt werden, z.B. im Fall mehrerer möglicher Pfade zum Ziel alle über denselben Pfad geroutet werden.

Payload Length

- ▶ Gibt die Länge der auf den IPv6-Header folgenden Daten (inkl. Extension Header, mehr dazu gleich).
- ▶ Angabe in Vielfachen von 1 B.
- ▶ Der IPv6-Header inkl. seiner Extension Header muss immer ein Vielfaches von 8 B sein.

Next Header

- ▶ Gibt den Typ des nächsten Headers an, der am Ende des IPv6-Headers folgt.
- ▶ Dies kann entweder ein L4-Header (z.B. TCP oder UDP), ein ICMPv6-Header oder ein sog. **IPv6 Extension Header** sein (mehr dazu gleich).

Hop Limit

- ▶ Entspricht dem TTL-Feld des IPv4-Headers.
- ▶ Wird beim Weiterleiten eines Pakets durch einen Router um jeweils 1 dekrementiert.
- ▶ Erreicht der Wert 0, wird das Paket verworfen und ein ICMPv6 Time Exceeded an den ursprünglichen Sender des Pakets zurückgeschickt.

Source Address

- ▶ 128 bit lange IPv6-Quelladresse.

Destination Address

- ▶ 128 bit lange IPv6-Zieladresse.

IPv6-Extension-Header

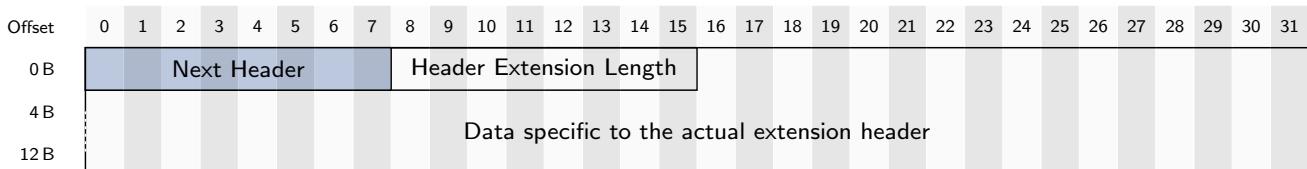


Abbildung: IPv6-Extension Header

Next Header

- ▶ Gibt den Typ des nächsten Headers an, der am Ende des IPv6-Headers folgt.
- ▶ Dies kann entweder ein L4-Header (z.B. TCP oder UDP), ein ICMPv6-Header oder ein weiterer IPv6 Extension Header sein.

Header Extension Length

- ▶ Länge des aktuellen Extension Headers in vielfachen von 8 B abzüglich der ersten 8 B.
- ▶ Die minimale Länge eines Extension Headers beträgt daher 8 B und die nutzbare Payload des Extension Headers mind. 6 B.

Extension Header Payload

- ▶ Daten spezifisch für den entsprechenden Extension Header.
- ▶ Da die Länge der Extension Header explizit angegeben ist, können Knoten, die den entsprechenden Extension Header nicht kennen, dennoch das Paket verarbeiten.

Beispiel:

- ▶ Rahmen haben auf Schicht 2 häufig eine maximale Größe, z.B. 1514 B für die L2-PDU (ohne CRC-C checksumme) bei IEEE 802.3u (100 Mbit/s Ethernet).
- ▶ Diese gibt auch die maximale Größe einer L3-PDU vor, welche als **Maximum Transmission Unit (MTU)** bezeichnet wird.
- ▶ Überschreitet eine L3-PDU diese Grösse, muss die L3-SDU **fragmentiert** und in Form unabhängiger Pakete versendet werden.
- ▶ Der Empfänger muss die einzelnen **Fragmente** im Anschluss **reassemblieren**.

IPv6 verfügt zu diesem Zweck über einen eigenen Extension Header, den **Fragment Extension Header**:

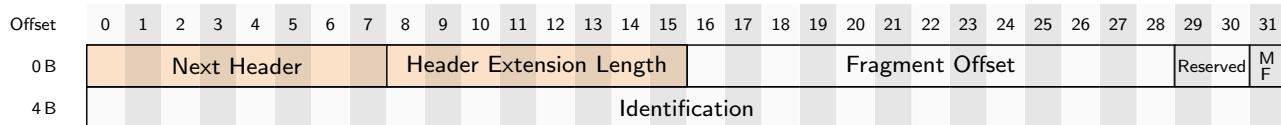


Abbildung: Fragment Extension Header

Next Header / Header Extension Length

- ▶ Generischer Extension-Header

Fragment Offset

- ▶ Offset der fragmenierten L3-SDU in Vielfachen von 8 B.
- ▶ Bei IPv6 erfolgt die Fragmentierung *ausschließlich* am Sender.
Frage: Woher kennt der Sender die maximale MTU auf dem Weg von sich selbst bis hin zum Ziel?
- ▶ Bei IPv4 können Pakete, falls nicht explizit über das DF-Bit untersagt, bei Bedarf auch von Routern fragmentiert werden.
Frage: Können Fragmente noch einmal fragmentiert werden?

Reserved

- ▶ Hat derzeit beim Fragmentation Header keine Verwendung.

More Fragments (MF)

- ▶ Gibt an, ob auf das aktuelle Paket weitere Fragmente folgen oder ob es sich um das letzte Fragment (gemäß des Fragment Offsets) handelt.

Identification

- ▶ 32 bit langer, vom Sender zufällig gewählter Wert, welcher alle Fragmente identifiziert, die zu einer L3-SDU reassembliert werden sollen.

Frage: Warum wird **nicht** das Flow Label des IPv6 Headers verwendet?

IPv6-Adressen zur besonderen Verwendung [3]

- ▶ ::1 / 128 – Loopback-Adresse
 - ▶ Adressiert den Localhost, d.h. Pakete mit dieser Zieladresse verlassen den lokalen Rechner garnicht erst, sondern werden über das sog. [Loopback-Interface](#) sofort wieder zugestellt (vgl. 127.0.0.1 bei IPv4).
 - ▶ Werden nicht geroutet.
- ▶ :: / 128 – nicht-spezifizierte Adresse
 - ▶ Analogon zu 0.0.0.0 bei IPv4.
 - ▶ Wird nicht geroutet.
- ▶ fe80:: / 10 – Link-Local Adressen
 - ▶ Jedes IPv6 Interface benötigt eine Link-Local-Adresse.
 - ▶ Die Link-Local-Adresse wird aus dem Interface-Identifier generiert.
 - ▶ Hat nur innerhalb des lokalen Links Gültigkeit und wird daher nicht geroutet.
- ▶ fc00: / 7 – Unique-Local Unicast-Adressen
 - ▶ Global eindeutige Adressen, die allerdings nur für lokale Kommunikation (z.B. innerhalb eines Firmennetzes) vorgesehen sind.
 - ▶ Dürfen wie private IPv4-Adressen lokal aber nicht im Internet geroutet werden.
- ▶ ff00:: / 8 – Multicast-Adressen
 - ▶ Adressen, die eine bestimmte Gruppe von Hosts adressieren (mehr dazu gleich).
 - ▶ Werden geroutet.
- ▶ (fast) der Ganze Rest – Globale Adressen
 - ▶ Global eindeutige Adressen, die für den Einsatz in öffentlichen Netzen vorgesehen sind.
 - ▶ Werden geroutet.

IPv6-Multicast

Grundsätzlich unterscheidet man sowohl auf Schicht 3/2 die vier folgenden Adressierungsarten:

1. Unicast
 - ▶ Pakete (Rahmen), die an ein einzelnes Ziel adressiert sind.
 - ▶ Alle anderen Knoten im Netzwerk verwerfen derartige Pakete (Rahmen) bzw. leiten sie lediglich zum Ziel weiter.
2. Broadcast
 - ▶ Pakete (Rahmen), die an alle Stationen im Netzwerk adressiert sind.
 - ▶ Adressierung erfolgt mittels spezieller Broadcast-Adressen.
 - ▶ Auf Schicht 3 sind Broadcasts zumeist auf das lokale Netzsegment begrenzt (vgl. Broadcast-Domain).
3. Multicast
 - ▶ Pakete (Rahmen), die an eine bestimmte Gruppe von Knoten adressiert sind.
 - ▶ Adressierung erfolgt mittels spezieller Multicast-Adressen.
 - ▶ Auf Schicht 2 werden Multicasts von Switches häufig wie Broadcasts behandelt.
 - ▶ Auf Schicht 3 gibt es spezielle Protokolle⁵, die Multicast-Adressierung auch über das lokale Netzsegment hinaus ermöglichen.
4. Anycast⁶
 - ▶ Pakete, die an eine beliebige Station einer bestimmten Gruppe adressiert sind.
 - ▶ Beispiel: Die wichtigsten Nameserver (→ Kapitel 5).

⁵ [Protocol Independent Multicast \(PIM\)](#) ermöglicht das Routing von Multicast-Pakete sowohl für IPv4 als auch IPv6. Details sind Inhalt weiterführender Veranstaltungen.

⁶ Wird im Rahmen der Veranstaltung nicht weiter behandelt.

Multicast ist elementarer Bestandteil von IPv6. Die folgenden Adressen bzw. Präfixe sind daher für bestimmte Multicast-Gruppen reserviert:

- ▶ **ff02::1 – All Nodes**
 - ▶ Adressiert alle Knoten auf dem lokalen Link.
- ▶ **ff02::2 – All Routers**
 - ▶ Adressiert alle Router auf dem lokalen Link.
- ▶ **ff02::1:2 – All DHCP-Agents**
 - ▶ Adressiert alle DHCP-Server auf dem lokalen Link.
- ▶ **ff02::1:ff00:0/104 – Solicited-Node Address**
 - ▶ Die Solicited-Node Adresse wird im **Neighbor Discovery Protocol** (mehr dazu gleich) verwendet, welches u.a. zur Adressauflösung dient.
 - ▶ Die Solicited-Node Adresse zu einer IPv6 Adresse wird aus dem Präfix ff02::1:ff00:0/104 und den letzten 24 bit der ursprünglichen IPv6 Adresse generiert.
 - ▶ Die Solicited-Node Adresse für 2001:0db8:1ee7:2ea2:0921:2e11:d2c6:938b ist somit ff02::1:ffc6:938b.
 - ▶ IPv6-Multicasts werden auch auf Schicht 2 mittels Multicast-Adressen versendet.
 - ▶ Switches müssen Multicast-Rahmen nur an diejenigen Ports weiterleiten, an denen ein Mitglied der entsprechenden Multicast-Gruppe angeschlossen ist.
 - ▶ In großen L2-Netzen können so unnötige Broadcasts vermieden werden.
 - ▶ Knoten, für die eine Nachricht nicht von Interesse ist, bekommen diese somit erst garnicht.

Mapping von Multicast IPv6 Adressen auf MAC-Adressen [4]

- ▶ IPv6-Pakete mit einer Zieladresse aus dem Präfix **ff00::/8** werden mit der zugehörigen Ethernet Multicast-Adresse versendet.
- ▶ Um Multicasts auf Schicht 3 auch auf Schicht 2 abilden zu können, muss es einen Zusammenhang zwischen den verwendeten Adressen beider Schichten geben.
- ▶ Die ersten 2 Oktette der MAC-Adresse werden auf **33:33** gesetzt.
 - ▶ letztes Bit des ersten Oktetts ist gesetzt → Multicast
 - ▶ vorletzte Bit des ersten Oktetts ist gesetzt → locally administered
 - ▶ siehe Kapitel 2
- ▶ Die letzten 4 Oktette der Ethernetadresse werden die letzten 4 Oktette der IPv6 Multicastadresse.

Beispiel:

$$\text{ff02::1:ffc6:938b} \mapsto \text{33:33:ff:c6:93:8b}$$

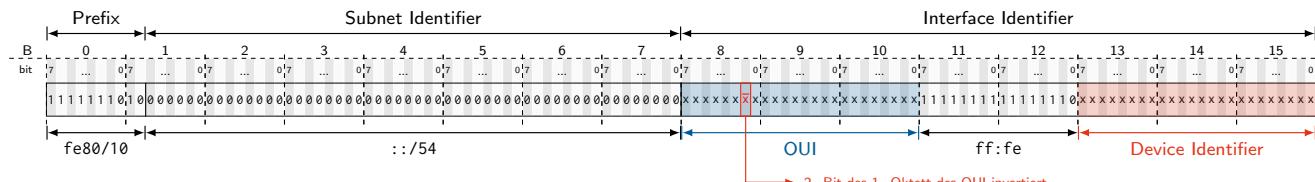
RFC 7042, Abschnitt 2.3.1:

(*Historical note: It was the custom during IPv6 design to use “3” for unknown or example values, and 3333 Coyote Hill Road, Palo Alto, California, is the address of PARC (Palo Alto Research Center, formerly “Xerox PARC”). Ethernet was originally specified by the Digital Equipment Corporation, Intel Corporation, and Xerox Corporation. The pre-IEEE [802.3] Ethernet protocol has sometimes been known as “DIX” Ethernet from the first letters of the names of these companies.*)

Stateless Address Autoconfiguration (SLAAC) [16]

IPv6 erlaubt eine automatische Konfiguration von Hosts innerhalb eines einzelnen Subnetzes.

Ein Host generiert sich die für ein Interface benötigte link-local IPv6-Adresse wie folgt:



- ▶ Das Präfix ist fe80::/10.
- ▶ Der Subnet Identifier (die folgenden 54 bit) werden auf 0 gesetzt.
- ▶ Die verbleibenden 64 bit stellen den **Interface Identifier** dar, welcher aus der MAC-Adresse des jeweiligen Interfaces als modifizierter EUI-64 Identifier generiert wird:
 - ▶ Die ersten 24 bit sind der OUI der MAC-Adresse.
 - ▶ Die nachfolgenden 16 bit werden mit ff:fe „gestopft“.
 - ▶ Die restlichen 24 bit werden mit dem Device Identifier der MAC-Adresse aufgefüllt.
- ▶ Dabei das vorletzte Bit des ersten Oktett des OUI (global/local-Bit) invertiert:
 - ▶ Bei MAC-Adressen bedeutet eine 0 an dieser Bitstelle eine global eindeutige und eine 1 eine lokal administrierte Adresse.
 - ▶ Bei IPv6 ist es genau andersrum.
 - ▶ Durch die Invertierung wird erreicht, dass eine manuell konfigurierte IPv6-Adresse wie 2001:db8::1 nicht einen Interface Identifier enthält, der auf eine global eindeutige MAC-Adresse hinweist.
 - ▶ Andernfalls müsste man von Hand Adressen wie 2001:db8::200:0:0:1 vergeben ...

Auch globale Adressen können über SLAAC konfiguriert werden:

- ▶ Um eine globale Adresse konfigurieren zu können, muss der Host zunächst wissen, welche IPv6 Präfixe von den lokalen Routern bedient werden.
- ▶ Präfix Informationen können von den Routern über das **Neighbor Discovery Protocol** (später) in Form von **Router Advertisements** versendet werden.
- ▶ Der Host kann sich über das /64 Präfix und dem modifizierten EUI-64 Identifier selbstständig eine Adresse erzeugen.
- ▶ SLAAC heißt **stateless**, da die Adressen nicht von einem Server vergeben werden
 - ▶ Globale Adressen können auch über DHCPv6 vergeben werden

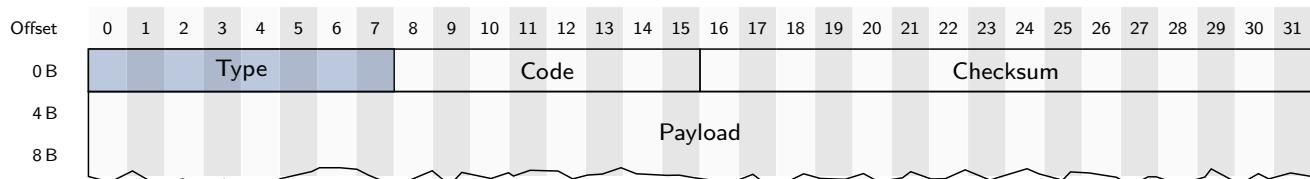
Frage: Welche Konsequenzen hat die Erzeugung des Interface Identifiers aus der MAC-Adresse einer Netzwerkkarte hinsichtlich der Privacy?

- ▶ Da MAC-Adressen i.d.R. eindeutig sind, kann ein Host durch die in seine IPv6-Adresse eingebettete MAC-Adresse unabhängig von Standard, Anschluss oder Provider verfolgt werden.
- ▶ Abhilfe schaffen die IPv6 Privacy Extensions [11]:
 - ▶ Erzeugung und regelmäßige Erneuerung von zufälligen Device Identifiern und damit einhergehende Wechsel der globalen IPv6 Adresse.
 - ▶ Zur Unterscheidung wird das vorletzte Bit des ersten Oktetts des „Device Identifiers“ auf 0 gesetzt.

Internet Control Message Protocol v6 (ICMPv6) [2]

Genereller Aufbau einer ICMPv6-Nachricht:

- ▶ Die ersten 32 bit sind immer vorhanden.
- ▶ Gesamtlänge der Nachricht hängt von Type und Code der ICMPv6-Nachricht ab.
- ▶ Die Länge, innerhalb der für IPv6 geltenden Grenzen, auf 1 B beliebig.



- ▶ **Type** gibt die Art der ICMP-Nachricht an, z.B. Echo, Time Exceeded usw.
- ▶ **Code** präzisiert die Art (den Type) der Nachricht, z.B. ob es sich bei einem ICMP Echo um einen Request oder eine Response handelt.
- ▶ **Checksum** ist eine (vergleichsweise einfache) fehlererkennende Checksumme:
 - ▶ Die Checksumme berücksichtigt das gesamte ICMPv6-Paket (inkl. Payload).
 - ▶ Zur Berechnung wird ein sog. **Pseudo-IPv6-Header** verwendet.
 - ▶ Der Pseudo-Header enthält die Absender- und Ziel-IP sowie das Next-Header-Feld.
- ▶ **Payload** der ICMPv6-Nachricht (Beispiele folgen).

Internet Control Message Protocol v6 (ICMPv6) [2]

Genereller Aufbau einer ICMPv6-Nachricht:

- ▶ Die ersten 32 bit sind immer vorhanden.
- ▶ Gesamtlänge der Nachricht hängt von Type und Code der ICMPv6-Nachricht ab.
- ▶ Die Länge, innerhalb der für IPv6 geltenden Grenzen, auf 1 B beliebig.



Warum ein „Pseudo-Header“?

- ▶ Wie wir aus Kapitel 1/2 wissen, können Kanalkodierung und Checksummen nicht garantieren, dass eine Nachricht selbst bei gültigen Checksummen korrekt übertragen wurde.
- ▶ Weitere Checksummen auf höheren Schichten können dies ebenfalls nicht, aber die Wahrscheinlichkeit für einen unbemerkt Fehlern sinkt hier selbst bei einfachen Checksummen drastisch.
- ▶ Gleichzeitig würde eine Checksumme über Headerfelder, welche sich von Hop zu Hop ändern, eine gewisse Belastung für Router (oder auf Schicht 2 für Switches) darstellen.
- ▶ Daher wird in diesem Fall die Checksumme nur über den Teil der Nachricht gebildet, welcher sich gewöhnlich nicht (oder nur sehr selten) auf dem Weg vom Sender zum Empfänger ändert.
- ▶ Beachten Sie beispielsweise den Ausschluss des Hopcounts, welcher sich ja bei jedem Hop ändert.

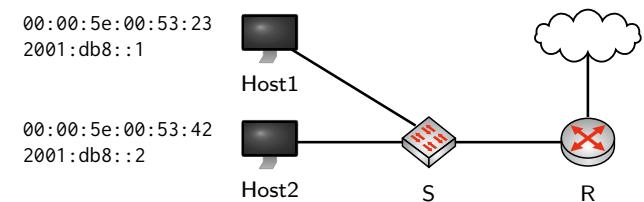
Neighbor Discovery Protocol (NDP)[12]

IPv6 bietet mit seiner **Neighbor Discovery**, welche Bestandteil von ICMPv6 ist, eine Reihe von Funktionalitäten, für die bei IPv4 nicht oder nur unvollständig standardisiert sind und eigene Protokolle notwendig gemacht haben. Funktionen der Neighbor Discovery sind insbesondere:

- ▶ Adressauflösung, Duplicate Address Detection und Neighbor Unreachability Detection: **Neighbor Solicitations** und **AdVERTISEMENTS**.
- ▶ Automatisches Auffinden von Routern innerhalb des lokalen Netzsegments, Adress-Präfixen und Parameter Konfiguration: **Router Discovery** und **Router AdVERTISEMENTS**.
- ▶ Umleitung zu anderen Gateways: **Redirects**.

Beispiel: Adressauflösung bei IPv6⁷

- ▶ Host1 will eine Nachricht an Host2 senden
- ▶ Die IP-Adresse von Host2 (2001:db8::2) sei ihm bereits bekannt
- ▶ Wie erhält Host1 die zugehörige MAC-Adresse?



⁷ Wir betrachten nur den Fall der Adressauflösung (vgl. ARP bei IPv4). Beim Einsatz von Neighbor Solicitations und Neighbor AdVERTISEMENTS zu anderen Zwecken sind i.A. andere Adressen und Optionen erforderlich.

Neighbor Solicitation (Request)



- ▶ **ICMPv6 Header**
 - ▶ ICMPv6 **Type** und **Code** (0x87 und 0x00 für eine Neighbor Solicitation Nachricht) sowie die ICMPv6 Checksumme.
- ▶ **Neighbor Discovery Body**
 - ▶ Die ersten 32 bit sind reserviert, so dass die Nachricht insgesamt wieder ein Vielfaches von 8 B lang wird.
 - ▶ Im Anschluss folgt Ziel-IPv6-Adresse, zu der die entsprechende MAC-Adresse gesucht wird.
- ▶ **Neighbor Discovery Options**
 - ▶ Neighbor Discovery Pakete können selbst wiederum Optionen enthalten.
 - ▶ **Type** und **Length** geben den Typ (1 für **Source Link Layer Address**) und Gesamtlänge der Option in Vielfachen von 8 B an.
 - ▶ Im Fall eines Neighbor Solicitation Pakets folgt L2-Adresse des anfragenden Knotens (**Source Link Address**).
 - ▶ Je nach Typ des NDP-Pakets können weitere Optionen folgen, wobei Empfänger unbekannte Optionen ignorieren müssen.

Neighbor Advertisement (Reply)



► ICMPv6 Header

- ICMPv6 Type und Code (0x88 und 0x00 für ein Neighbor Advertisement) sowie die ICMPv6 Checksumme.

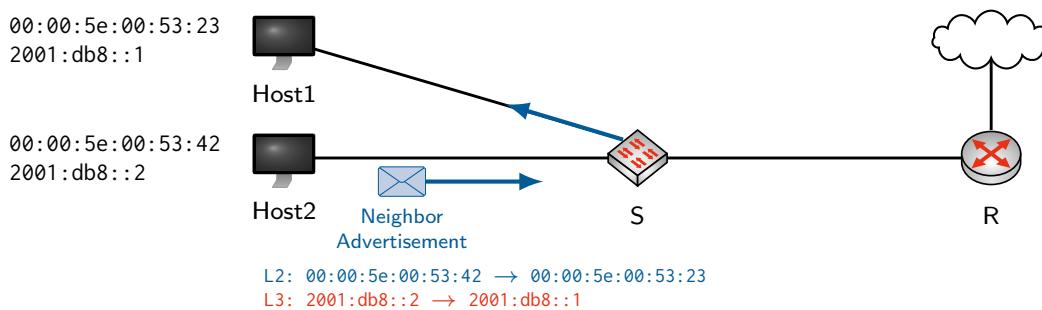
► Neighbor Discovery Body

- Die drei höchstwertigen Bit des ersten Oktetts haben folgende Bedeutungen:
 - Router-Flag R wird gesetzt, wenn der antwortende Knoten ein Router ist.
 - Solicited Flag S gibt an, ob das Advertisement infolge einer Solicitation geschickt wird.
 - Override Flag O wird gesetzt, wenn das Advertisement eine möglicherweise gecached Link-Layer Adresse beim Empfänger aktualisieren soll.

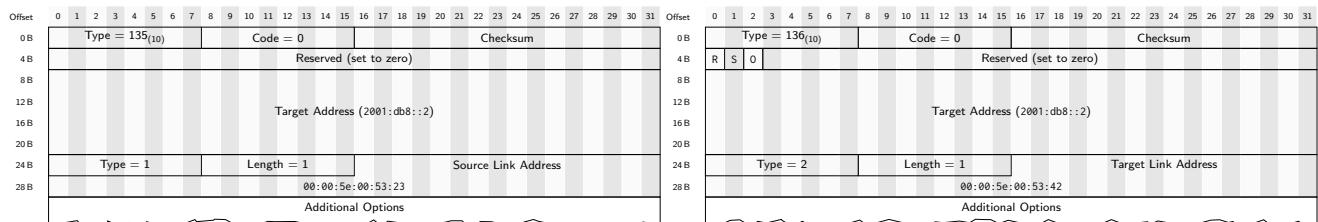
► Neighbor Discovery Options

- Type und Length geben den Typ (2 für Target Link Layer Address) und Gesamtlänge der Option in Vielfachen von 8 B an.
- Im Fall eines Neighbor Advertisements folgt die L2-Adresse des angefragten Knotens (Target Link Address).
- Je nach Typ des NDP-Pakets können weitere Optionen folgen, wobei Empfänger unbekannte Optionen ignorieren müssen.

Beispiel:



- Host1 sendet eine „Neighbor Solicitation for 2001:db8::2 from 00:00:5e:00:53:23“ an die zur bekannten IPv6 Adresse gehörende Solicited-Node Adresse (Multicast).
- Host2 empfängt diese Nachricht und antwortet mit einer „Neighbor Advertisement 2001:db8::2 (sol, ovr) is a 00:00:5e:00:53:42“ Nachricht (Unicast).



Übersicht

Motivation

Vermittlungsarten

Leitungsvermittlung
Nachrichtenvermittlung
Paketvermittlung

Adressierung im Internet

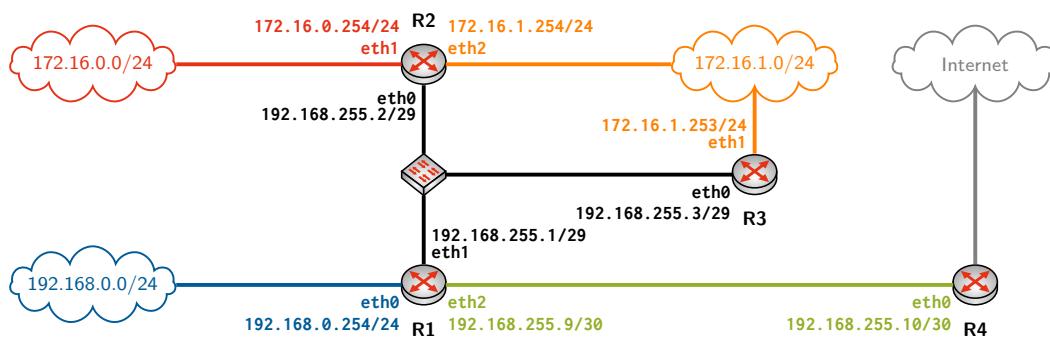
Internet Protocol version 4 (IPv4)
Internet Protocol version 6 (IPv6)

Wegwahl (Routing)

Routing Table und Longest Prefix Matching
Dynamisches Routing
Routing Information Protocol (RIP)
Autonome Systeme

Statisches Routing

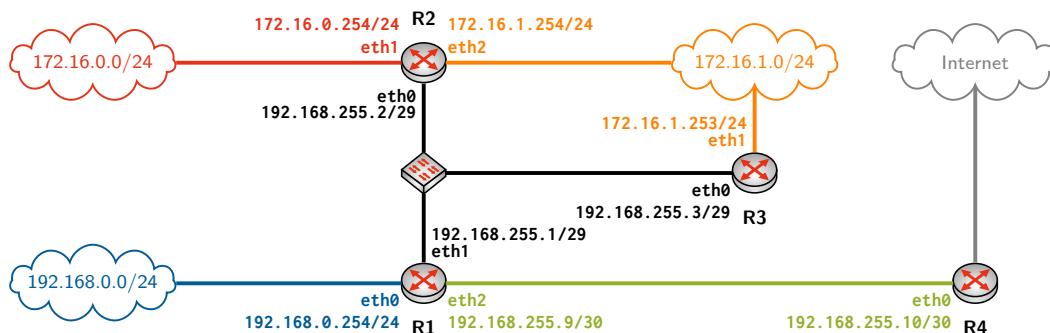
Wir betrachten im Folgenden das unten abgebildete Beispielnetzwerk:



- ▶ Die Farben der Links und Interface-Adressen verdeutlichen die einzelnen Subnetze
- ▶ Das Netzwerk 192.168.255.0/29 (schwarz) verfügt über 6 nutzbare Hostadressen
- ▶ Das Netzwerk 192.168.255.8/30 (grün) ist ein **Transportnetz** mit nur 2 nutzbaren Hostadressen
- ▶ Die übrigen Netze sind /24 Netze mit jeweils 254 nutzbaren Hostadressen

Frage: Wie entscheidet R1, an welchen **Next-Hop** ein Paket weitergeleitet werden soll?

Routing Table



Definition: Routing Table

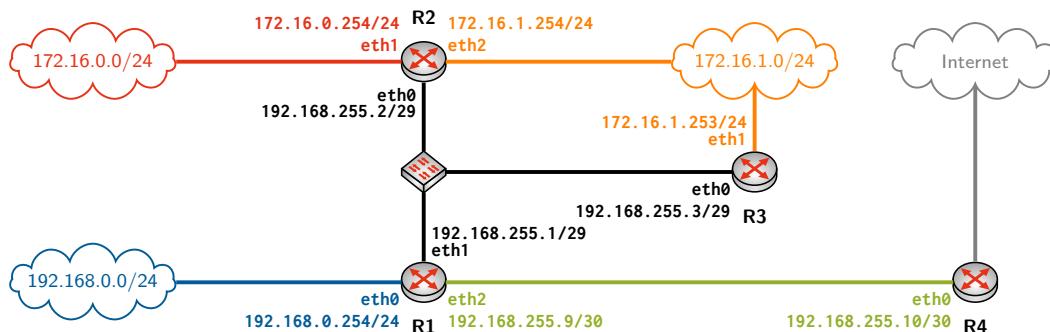
In der **Routing-Tabelle** speichert ein Router (oder Host)

- ▶ die Netzadresse eines Ziels,
- ▶ die Länge des Präfixes,
- ▶ den zugehörigen Next-Hop (auch Gateway genannt),
- ▶ das Interface, über welches dieser Next-Hop erreichbar ist, und
- ▶ die **Kosten** bis zum Ziel.

Hinweise:

- ▶ Bei IPv4 wird häufig anstatt der Präfixlänge die Subnetzmaske (oder **Genmask**) angegeben. Bei einem Präfix von N bit handelt es sich bei IPv4 um einen Block von vier Oktetten, wobei genau die erste N bit dieses Blocks 1 und alle übrigen 0 sind.
- ▶ Die Kosten werden fälschlicherweise auch als **Metrik** bezeichnet. Die Metrik hingegen ist, woraus die Kosten berechnet werden (z.B. Hop-Count, Bandbreite, Verzögerung etc.).

Routing Table



Beispiel: Routing-Tabelle für R1

Destination	NextHop	Costs	Iface
192.168.255.8/30	0.0.0.0	0	eth2
192.168.255.0/29	0.0.0.0	0	eth1
192.168.0.0/24	0.0.0.0	0	eth0
172.16.1.0/24	192.168.255.3	1	eth1
172.16.0.0/23	192.168.255.2	1	eth1
0.0.0.0/0	192.168.255.10	0	eth2

- ▶ Die Netze 172.16.{0..1}.0/24 wurden zusammengefasst
- ▶ Die Route 0.0.0.0 wird auch als **Default Route** bezeichnet
- ▶ Interessant: R1 kennt zwei (eigentlich sogar drei) Routen zum Netz 172.16.1.0/24 !

Longest Prefix Matching

1. R1 berechnet das logische AND aus der Zieladresse des Pakets und den Subnetzmasken (welche aus der Präfixlänge hervorgehen) in seiner Routingtabelle.
2. Das Ergebnis wird mit dem Eintrag in der Spalte „Destination“ verglichen.
3. Stimmt das Ergebnis damit überein, werden Gateway und zugehöriges Interface bestimmt.
4. Nachdem die MAC-Adresse des Gateways ggf. via ARP aufgelöst wurde, wird das Paket mit einem neuen Ethernet-Header versehen und weitergeleitet.

Beispiel: R1 erhalte ein Paket mit der Zieladresse 172.16.1.23.

Destination	NextHop	Costs	Iface
192.168.255.8/30	0.0.0.0	0	eth2
192.168.255.0/29	0.0.0.0	0	eth1
192.168.0.0/24	0.0.0.0	0	eth0
172.16.1.0/24	192.168.255.3	1	eth1
172.16.0.0/23	192.168.255.2	1	eth1
0.0.0.0/0	192.168.255.10	0	eth2

IP-Adresse	10101100 . 00010000 . 00000001 . 00010111	172.16.1.23
Subnetz Maske	11111111 . 11111111 . 11111111 . 11111100	255.255.255.252
Netzadresse	10101100 . 00010000 . 00000001 . 00010100	172.16.1.20

⇒ kein Match, da $172.16.1.20 \neq 192.168.255.8$

Longest Prefix Matching

IP-Adresse	10101100 . 00010000 . 00000001 . 00010111	172.16.1.23
Subnetz Maske	11111111 . 11111111 . 11111111 . 11111100	255.255.255.248
Netzadresse	10101100 . 00010000 . 00000001 . 00010000	172.16.1.16

⇒ kein Match, da $172.16.1.16 \neq 192.168.255.0$

IP-Adresse	10101100 . 00010000 . 00000001 . 00010111	172.16.1.23
Subnetz Maske	11111111 . 11111111 . 11111111 . 00000000	255.255.255.0
Netzadresse	10101100 . 00010000 . 00000001 . 00000000	172.16.1.0

⇒ kein Match, da $172.16.1.0 \neq 192.168.0.0$

IP-Adresse	10101100 . 00010000 . 00000001 . 00010111	172.16.1.23
Subnetz Maske	11111111 . 11111111 . 11111111 . 00000000	255.255.255.0
Netzadresse	10101100 . 00010000 . 00000001 . 00000000	172.16.1.0

⇒ Match, da $172.16.1.0 = 172.16.1.0 \Rightarrow$ Gateway ist 192.168.255.3

Definition: Longest Prefix Matching

Die Routingtabelle wird von längeren Präfixen (spezifischeren Routen) hin zu kürzeren Präfixen (weniger spezifische Routen) durchsucht. Der erste passende Eintrag liefert das Gateway (Next-Hop) eines Pakets. Diesen Prozess bezeichnet man als **Longest Prefix Matching**.

Beachte:

Destination	NextHop	Costs	Iface
192.168.255.8/30	0.0.0.0	0	eth2
192.168.255.0/29	0.0.0.0	0	eth1
192.168.0.0/24	0.0.0.0	0	eth0
172.16.1.0/24	192.168.255.3	1	eth1
172.16.0.0/23	192.168.255.2	1	eth1
0.0.0.0/0	192.168.255.10	0	eth2

- ▶ Der Eintrag für 172.16.0.0/23 liefert ebenfalls einen Match, ist aber weniger spezifisch als der für 172.16.1.0/24 (1 bit kürzeres Präfix).
- ▶ Die Default Route 0.0.0.0/0 liefert immer einen Match (logisches AND mit der Maske 0.0.0.0).
- ▶ Es ist nicht garantiert, dass das Gateway/Next-Hop der Default Route („Gateway of last resort“) eine Route zum Ziel kennt (→ ICMP Destination Unreachable / Host Unreachable).
- ▶ Routen zu direkt verbundenen Netzen (also solchen, zu denen ein Router selbst gehört) können automatisch erzeugt werden. Der NextHop ist in diesem Fall die unspezifizierte Adresse.
- ▶ Routen zu entfernten Netzen müssen „gelernt“ werden – entweder durch händisches Eintragen (statisches Routing) oder durch **Routing Protokolle** (dynamisches Routing).

Dynamisches Routing

Mittels **Routing Protokollen** können Router miteinander kommunizieren und Routen untereinander austauschen. Routingprotokolle können nach Ihrer Funktionsweise wie folgt gruppiert werden:

Distanz-Vektor-Protokolle

- ▶ Router kennen nur Richtung (NextHop) und Entfernung (Kosten) zu einem Ziel (vgl. Straßenschild mit Richtungs- und Entfernungsangabe).
- ▶ Router haben keine Information über die Netzwerktopologie.
- ▶ Router tauschen untereinander lediglich kumulierte Kosten aus (z. B. den Inhalt Ihrer Routingtabellen).
- ▶ Funktionsprinzip basiert auf dem **Algorithmus von Bellman-Ford**, der kürzeste Wege ausgehend von einem Startknoten ermittelt und sich leicht verteilt implementieren lässt.

Link-State-Protokolle

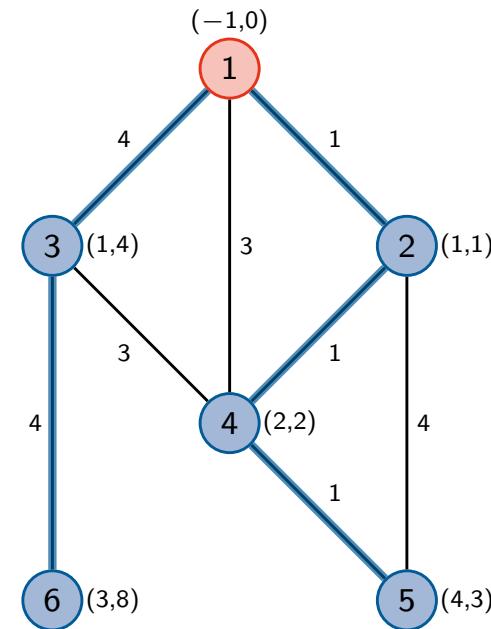
- ▶ Router informieren einander zusätzlich zu den Kosten auch darüber, wie ein Ziel erreichbar ist.
- ▶ Häufig komplexe Nachbarschaftsbeziehungen und Update-Nachrichten.
- ▶ Router erhalten so vollständige Topologieinformationen.
- ▶ Basierend auf den Topologieinformationen bestimmt jeder Router kürzeste Pfade, z. B. mittels **Dijkstras Algorithmus**.

Algorithmus von Bellman-Ford

- ▶ $p[i]$: Vorgänger von Knoten i im Graphen
- ▶ $d[i]$: Distanz von der Wurzel zu Knoten i

```
// Initialisierung
for i ∈ N do
    p[i] = -1
    d[i] = { 0   i = s
             ∞  sonst
end for
T = {s} // Menge der erreichbaren Knoten
```

```
// Berechnung der Pfade
while d[j] changes for some j ∈ N do
    S = {} // Aktualisierte Knoten
    for i ∈ T do
        // Für alle Nachbarn von i:
        for ∀j : (i,j) ∈ E do
            if d[i] + cij < d[j] then
                p[j] = i
                d[j] = d[i] + cij
                S = S ∪ {j}
            end if
        end for
    end for
    T = T ∪ S
end while
```

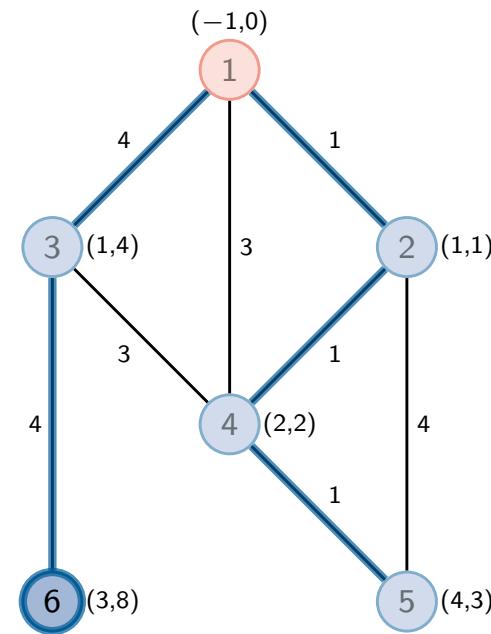


Dijkstras Algorithmus

- ▶ $p[i]$: Vorgänger von Knoten i im Graphen
- ▶ $d[i]$: Distanz von der Wurzel zu Knoten i
- ▶ Priority Queue Q , welche Elemente nach Schlüsseln sortiert ausgibt

```
for i ∈ N do
    p[i] = -1
    d[i] = { 0   i = s
             ∞  sonst
    Q.enqueue(i,d[i])
end for
T = {} // Menge der bearbeiteten Knoten

// Berechnung der Pfade
while T ≠ N do
    i = Q.dequeue()
    T = T ∪ {i}
    // Für alle Nachbarn von i:
    for ∀j : (i,j) ∈ E do
        if d[i] + cij < d[j] then
            Q.decreaseKey(j,d[i] + cij)
            p[j] = i
        end if
    end for
end while
```



Eigenschaften des Algorithmus von Bellman-Ford:

- ▶ Im n -ten Durchlauf der while-Schleife werden alle Pfade der Länge höchstens n berücksichtigt (vergleiche min-plus-Produkt in n -ter Potenz).
- ▶ Keine komplexen Datenstrukturen notwendig.
- ▶ Verteilte (dezentrale) Implementierung ohne Kenntnis der Topologie möglich.
- ▶ Laufzeit in $\mathcal{O}(|N| \cdot |E|)$.

Eigenschaften des Algorithmus von Dijkstra:

- ▶ Es werden immer Pfade über den im jeweiligen Schritt am günstigsten erreichbaren Knoten gesucht ([Greedy-Prinzip](#)).
- ▶ Wurde ein Knoten abgearbeitet, so ist garantiert, dass der kürzeste Pfad zu diesem Knoten gefunden ist.
- ▶ Resourcenintensiver als der Algorithmus von Bellman-Ford, da komplexere Datenstrukturen notwendig sind (Priority Queue).
- ▶ Vollständige Kenntnis der Netzwerktopologie erforderlich.
- ▶ Asymptotisch bessere Laufzeit.
- ▶ Laufzeit in $\mathcal{O}(|E| + |V| \log_2 |V|)$.

[Routing Information Protocol \(RIP\) \[7, 10\]](#)

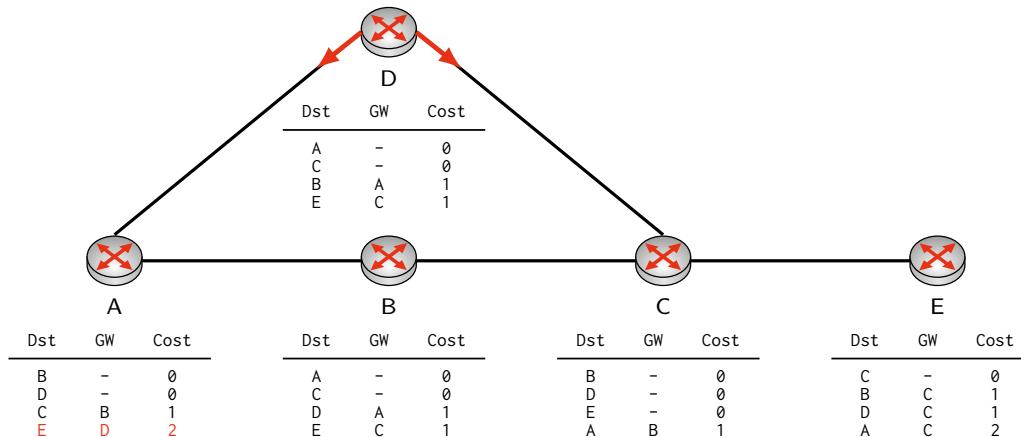
- ▶ Einfach Distanz-Vektor-Protokoll
- ▶ RIPv1 standardisiert in RFC 1058 (1988)
- ▶ Unterstützung für CIDR in RIPv2 hinzugefügt (RFC 2453, 1998)
- ▶ Einzige Metrik: [Hop Count](#) (entspricht Bellman-Ford mit Kantengewicht 1 auf allen Kanten)
- ▶ Hop Count Limit von 15, weiter entfernte Ziele sind nicht erreichbar

Funktionsweise:

- ▶ Router senden in regelmäßigen Abständen (Standardwert 30 s) den Inhalt ihrer Routingtabelle an die Multicast-Adresse 224.0.0.9.
- ▶ Alle Geräte mit dieser Multicast-Adresse akzeptieren das Update.
- ▶ Jeder RIP-Router akzeptiert diese Update-Nachrichten, inkrementiert die Kosten der enthaltenen Routen um 1 und vergleicht die Routen mit bereits vorhandenen Routen aus seiner Routingtabelle:
 - ▶ Enthält das Update eine noch unbekannte Route, wird diese in die eigene Routingtabelle übernommen.
 - ▶ Enthält das Update eine Route zu einem bekannten Ziel aber mit niedrigeren Kosten, so wird die vorhandene Route durch das Update ersetzt.
 - ▶ Andernfalls wird die vorhandene Route beibehalten.
- ▶ Bleiben fünf aufeinanderfolgende Updates von einem Nachbarn aus, so werden alle Routen über diesen Next Hop aus der Routingtabelle entfernt.

Beispiel mit vereinfachter Darstellung:

- ▶ Zwischen jeweils zwei Routern befindet sich sinnvollerweise ein Switch, welcher den Anschluss weiterer Computer an das jeweilige Subnetz ermöglicht.
- ▶ Anstelle von IP- und Netzadressen tragen wir in die Routingtabellen lediglich die Namen der Router ein.
- ▶ Für direkt erreichbare Nachbarn tragen wir kein Gateway ein.
- ▶ Update-Nachrichten werden rundenweise verschickt (erst A, dann B, ...).



- ▶ Nach diesem Schritt kennt jeder Router eine kürzeste Route zu jedem anderen Router.
- ▶ Da mehrere gleich lange Pfade existieren, bleibt es dem Zufall (der Reihenfolge der Updatenachrichten) überlassen, ob beispielsweise Router A als Next Hop zu E Router D oder B lernt.

Das vorangegangene Beispiel lässt sich leicht auf gewichtete Kanten erweitern:

- ▶ Metrik ist nicht mehr die Anzahl der Hops zum Ziel, sondern das Kantengewicht.
- ▶ Router addieren auf erhaltene Updates das Kantengewicht des Links, über den das Update empfangen wurde.

Problem:

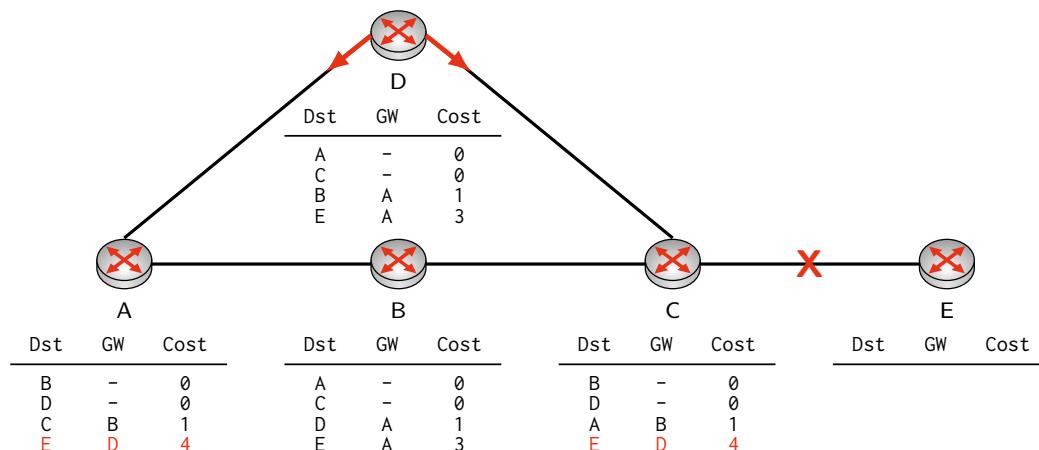
- ▶ Bis jeder Router den besten Next Hop bestimmen kann, dauert es ggf. mehrere „Runden“.
- ▶ Eine obere Schranke für die Anzahl der notwendigen Nachrichten, die jeder Router senden muss, ist die maximale Entfernung zwischen zwei Routern in Hops.
- ▶ Die maximale Entfernung beträgt bei RIP 15 Hops.
- ▶ Da Updates nur alle 30 s verschickt werden, ergibt sich eine maximale Verzögerung von $15 \cdot 30 \text{ s} = 7,5 \text{ min}$.

Lösung: Triggered Updates

- ▶ Sobald ein Router eine Änderung an seiner Routingtabelle vornimmt, sendet er sofort ein Update.
- ▶ Dies führt zu einer Welle von Updates durch das Netzwerk.
- ▶ Konvergenzzeit wird reduziert, aber das Netzwerk während der Updates ggf. stark belastet.

Anderes Problem: Count to infinity

- ▶ Link zwischen C und E fällt aus.
- ▶ Reihenfolge, in der Updates versendet werden, ist dem Zufall überlassen.



Je nach Reihenfolge der Updates

- ▶ wird die fehlerhafte Route zu E weiterverbreitet und
- ▶ die Metrik stets inkrementiert
- ▶ bis schließlich das Hop Count Limit von 15 erreicht ist.

Diesen Vorgang bezeichnet man als **Count to infinity**.

Lösungen:

- ▶ **Split Horizon**
 - ▶ „Sende dem Nachbarn, von dem Du die Route zu X gelernt hast, keine Route zu X.“
 - ▶ Im vorherigen Beispiel würde A die Route zu E nicht an D, wohl aber an B schicken.
 - ▶ Split Horizon verbessert die Situation, kann das Problem aber nicht lösen.
- ▶ **Poison Reverse**
 - ▶ Anstelle dem Nachbarn, von dem eine Route zu X gelernt wurde, keine Route zu X mehr zu schicken, wird eine Route mit unendlicher Metrik gesendet.
 - ▶ Im vorherigen Beispiel würde A die Route zu E an D mit Metrik 15 schicken.
 - ▶ Die fehlerhafte Route würde nach wie vor B erreichen.
 - ▶ Auch Poison Reverse kann das Problem nicht vollständig lösen.
- ▶ **Path Vector**
 - ▶ Sende bei Updates nicht nur Ziel und Kosten, sondern auch den vollständigen Pfad, über den das Ziel erreicht wird.
 - ▶ Jeder Router prüft vor Installation der Route, ob er selbst in diesem Pfad bereits vorhanden ist.
 - ▶ Falls ja, handelt es sich um eine Schleife und das Update wird verworfen.
 - ▶ Path Vector verhindert Routing Loops und damit auch Count to Infinity, vergrößert jedoch die Update-Nachrichten und die Protokollkomplexität.

Übersicht: Ausgewählte Routing-Protokolle

Distanz-Vektor-Protokolle

- ▶ **RIP (Routing Information Protocol)**
Sehr einfaches Protokoll, Hop-Count als einzige Metrik, geeignet für eine geringe Anzahl von Netzen, wird von den meisten Routern unterstützt (sogar einige Heimgeräte)
- ▶ **IGRP (Interior Gateway Routing Protocol)**
Proprietäres Routing Protokoll von Cisco, unterstützt komplexere Metriken als RIP
- ▶ **EIGRP (Enhanced Interior Gateway Routing Protocol)**
Proprietäres Routing Protokoll von Cisco, Nachfolger von IGRP, deutlich verbesserte Konvergenzeigenschaften.
- ▶ **AODV (Ad hoc On-Demand Distance Vector)**
Einsatz in kabellosen vermaschten Netzwerken, Routen werden nicht proaktiv ausgetauscht sondern on-demand gesucht (reaktives Protokoll)

Link-State-Protokolle

- ▶ **OSPF (Open Shortest Path First)**
Industriestandard für mittlere bis große Anzahl von Netzwerken
- ▶ **IS-IS (Intermediate System to Intermediate System)**
Seltener eingesetztes, leistungsfähiges Routingprotokoll, welches unabhängig von IP ist, da es sein eigenes L3-Protokoll mitbringt
- ▶ **HWMP (Hybrid Wireless Mesh Protocol)**
Ermöglicht Routing in IEEE 802.11s (Wireless Mesh Networks), wobei Routing-Entscheidungen hier allerdings auf Basis von MAC-Adressen anstatt von IP-Adressen getroffen werden⁸

⁸ MAC-based Routing ist ein Spezialfall für (kabellose) Meshnetzwerke, in dem sich nicht alle Knoten direkt erreichen aber dennoch eine gemeinsame Broadcast-Domain bilden.

Autonome Systeme

Alle bislang vorgestellten Routingprotokolle

- ▶ bestimmen beste Pfade anhand objektiver Kriterien (Hopcount, Bandbreite, Delay, ...),
- ▶ bieten aber keine bzw. nur eingeschränkte Möglichkeiten, Routen direkt zu beeinflussen.

Manchmal ist es aber wünschenswert, Routen auf Basis anderer Kriterien zu wählen:

- ▶ Tatsächlich anfallende monetäre Kosten
- ▶ Netze / Länder, durch die Datenverkehr zu einem Ziel weitergeleitet wird
- ▶ Infrastrukturentscheidungen (z. B. Belastung einzelner Router)

Routingentscheidungen auf Basis derartiger Kriterien bezeichnet man als **Policy-Based Routing**.

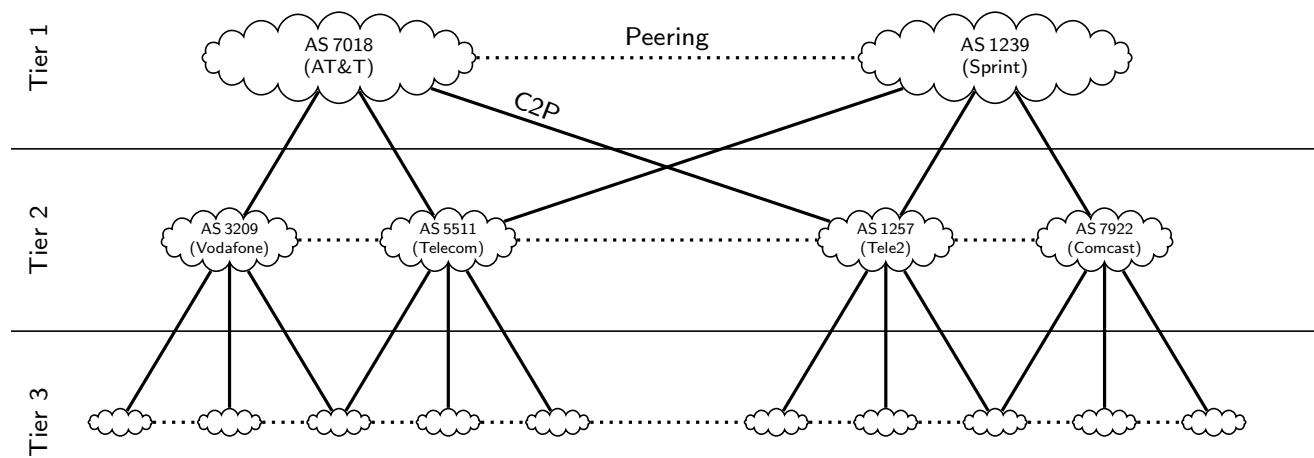
Definition: Autonomes System

Eine Menge von Netzwerken, die unter einheitlicher administrativer Kontrolle stehen, bezeichnet man als **Autonomes System (AS)**. Ein AS wird durch einen 16 bit Identifier, der sog. **AS-Nummer** identifiziert. Beim Einsatz von Routingprotokollen wird unterschieden:

- ▶ Innerhalb eines autonomen Systems werden **Interior Gateway Protocols (IGPs)** wie RIP, OSPF, EIGRP oder IS-IS eingesetzt.
- ▶ Zum Austausch von Routen zwischen Autonomen Systemen wird ein **Exterior Gateway Protocol (EGP)** verwendet.

Das einzige in der Praxis verwendete EGP ist das **Border Gateway Protocol (BGP)**.

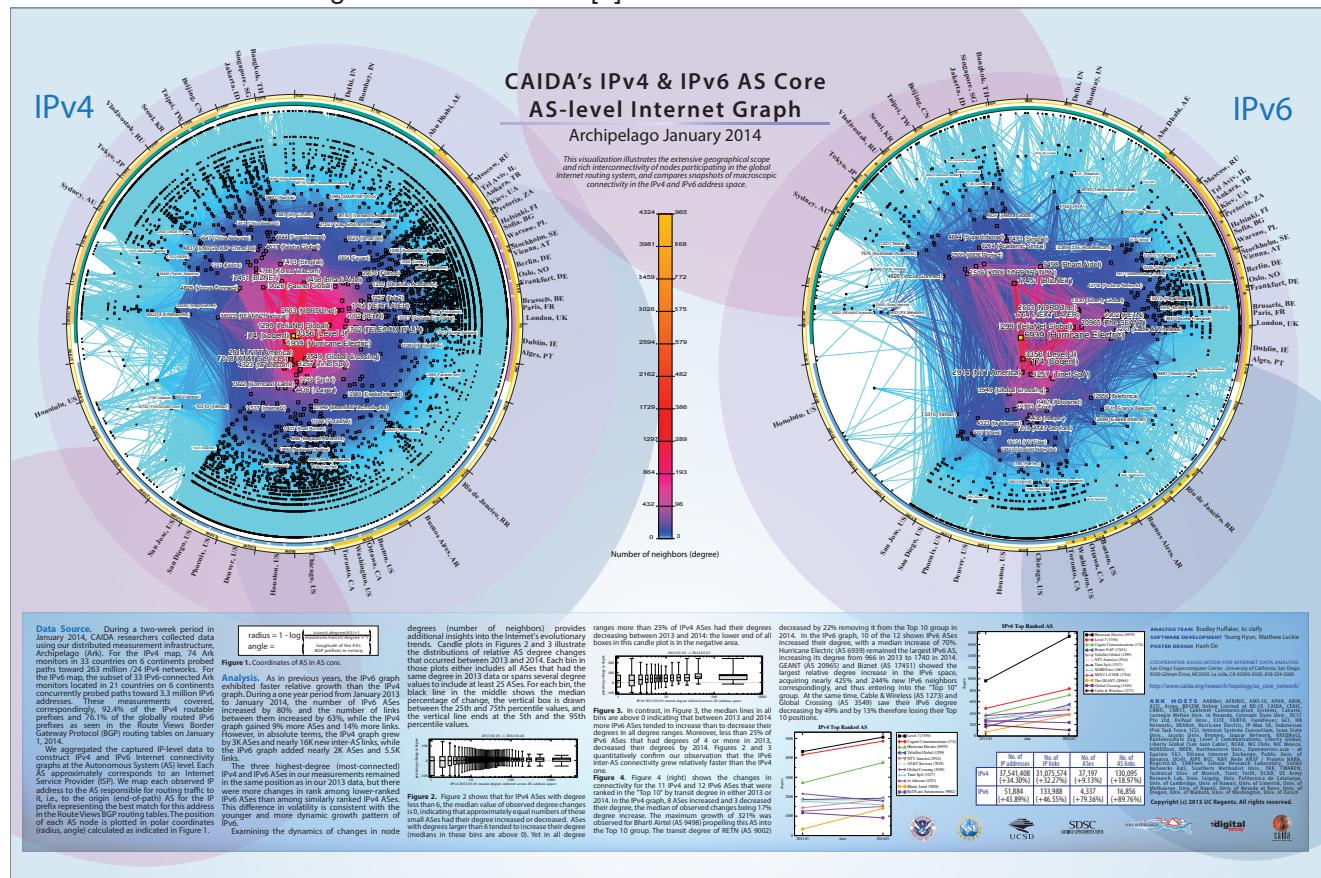
Das Internet: Stark vereinfachte schematische Darstellung



- Autonome Systeme können durch Upstream-Provider oder durch Peering miteinander verbunden sein.
 - An Internet Exchange Points, z. B. DE-CIX, existieren zahlreiche Peering-Verbindungen.
 - Peering-Verbindungen sind aus Kostengründen gegenüber Customer-Provider (C2P) Verbindungen zu bevorzugen.
 - Die Border-Router eines AS „announces“ Präfixe, die über dieses AS erreichbar sind.
 - AS 5511 announced seine eigenen Customer an seine Peerings und Upstream-Provider.
 - AS 5511 würde evtl. die Netze von Vodafone an Tele2 announce, sicher aber nicht an Sprint oder AT&T.

Faustregel: Für vertikale Verbindungen muss der jeweilige Customer („kleinere Provider“) bezahlen, weshalb horizontale Verbindungen (Peerings) bevorzugt werden.

Das Internet: Etwas weniger stark vereinfacht [8]



Zusammenfassung

In diesem Kapitel haben wir

- ▶ die Vorteile von Paketvermittlung gegenüber Leitungs- und Nachrichtenvermittlung diskutiert,
- ▶ die Notwendigkeit logischer Adressen zur End-zu-End Adressierung erkannt,
- ▶ die beiden wichtigsten Protokolle für End-zu-End Adressierung im Internet kennen gelernt,
- ▶ Methoden zur weiteren logischen Unterteilung von Netzen in Subnetze und Präfixe behandelt und
- ▶ ein grundlegendes Verständnis bzgl. des Austauschs von Routinginformationen im Internet entwickelt.

Wir sollten nun wissen,

- ▶ was die Unterschiede zwischen Leitungs-, Nachrichten- und Paketvermittlung sind,
- ▶ Worin der technische und logische Unterschied zwischen den Adressen auf Schicht 2 und 3 besteht,
- ▶ welche Möglichkeiten IPv4 und IPv6 zur logischen Strukturierung bieten und wie dies funktioniert,
- ▶ wie zur einer gegebenen Adresse auf Schicht 3 die zugehörige Link-Layer Adresse bestimmt wird,
- ▶ was eine Routing Tabelle ist,
- ▶ wie Router Weiterleitungssentscheidungen treffen,
- ▶ was der Unterschied zwischen Routing und Forwarding ist,
- ▶ wie Router untereinander Routen austauschen
- ▶ welche Arten von Routingprotokollen es gibt und
- ▶ wie diese funktionieren.

Bibliography I

- [1] CAIDA. IPv4 Census Map. <http://www.caida.org/research/id-consumption/#ipv4-census-map>.
- [2] A. Conta and S. Deering. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification, 2006. <http://tools.ietf.org/html/rfc4443>.
- [3] M. Cotton, L. Vegoda, R. Bonica, and B. Haberman. Special-Purpose IP Address Registries, 2013. <http://tools.ietf.org/html/rfc6890>.
- [4] M. Crawford. Transmission of IPv6 Packets over Ethernet Networks, 2007. <https://tools.ietf.org/html/rfc2464>.
- [5] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification, 1995. <http://tools.ietf.org/html/rfc1883>.
- [6] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification, 1998. <http://tools.ietf.org/html/rfc2460>.
- [7] C. Hedrick. Routing Information Protocol, 1988. <http://tools.ietf.org/html/rfc1058>.
- [8] B. Huffaker, Y. Hyun, and M. Luckie. IPv4 and IPv6 AS Core: Visualizing IPv4 and IPv6 Internet Topology at a Macroscopic Scale in 2014, 2014. http://www.caida.org/research/topology/as_core_network/2014.
- [9] S. Kawamura and M. Kawashima. A Recommendation for IPv6 Address Text Representation, 2010. <http://tools.ietf.org/html/rfc5952>.
- [10] G. Malkin. RIP Version 2, 1998. <http://tools.ietf.org/html/rfc2453>.
- [11] T. Narten, R. Draves, and S. Krishnan. Privacy Extensions for Stateless Address Autoconfiguration in IPv6, 2007. <https://tools.ietf.org/html/rfc4941>.
- [12] E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6), 2007. <http://tools.ietf.org/html/rfc4861>.
- [13] L. L. Peterson and S. D. B. *Computer Networks – A System Approach*, chapter Internetworking, pages 234 – 242. Elsevier, 4. edition, 2007. Auszug s. Moodle/SVN.
- [14] L. L. Peterson and S. D. B. *Computer Networks – A System Approach*, chapter Internetworking, pages 248 – 256. Elsevier, 4. edition, 2007. Auszug s. Moodle/SVN.
- [15] L. L. Peterson and S. D. B. *Computer Networks – A System Approach*, chapter Internetworking, pages 259 – 262. Elsevier, 4. edition, 2007. Auszug s. Moodle/SVN.
- [16] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration, 2007. <http://tools.ietf.org/html/rfc2462>.

Grundlagen Rechnernetze und Verteilte Systeme

Kapitel 4 – Transportschicht

Worum geht es in diesem Kapitel?

Motivation

Multiplexing

Verbindungslose Übertragung

Verbindungsorientierte Übertragung

Sliding-Window-Verfahren

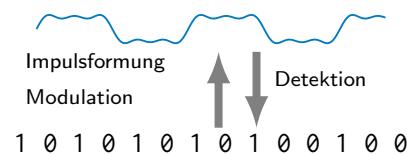
Transmission Control Protocol (TCP)

Fluss- und Staukontrolle bei TCP

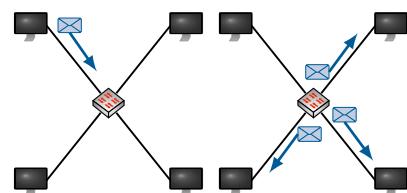
Network Address Translation (NAT)

Wir haben bislang gesehen:

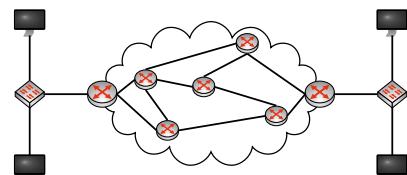
- ▶ Wie digitale Daten durch messbare Größen dargestellt, übertragen und rekonstruiert werden (Schicht 1)



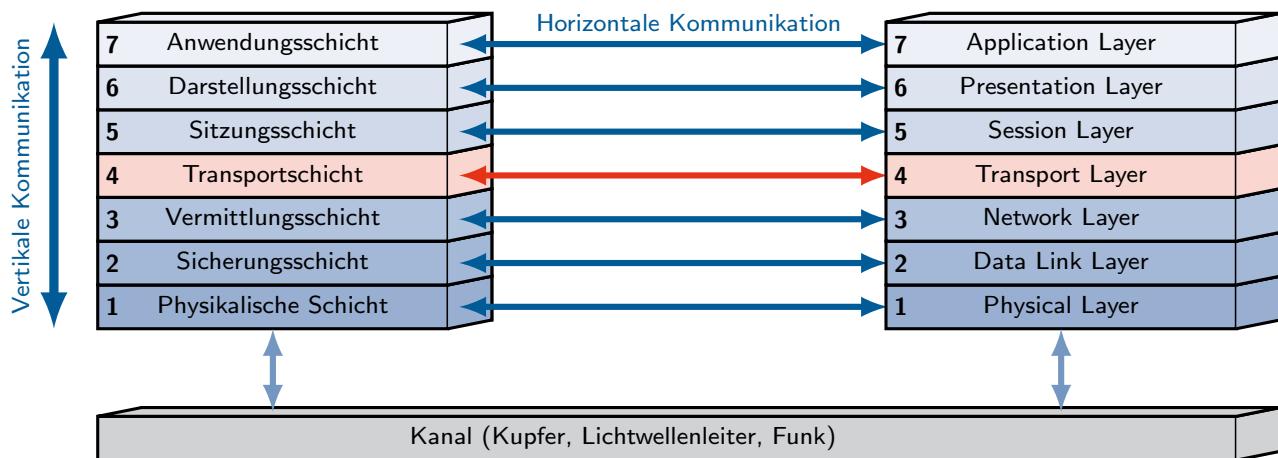
- ▶ Wie der Zugriff auf das Übertragungsmedium gesteuert und der jeweilige Next-Hop adressiert wird (Schicht 2)



- ▶ Wie auf Basis logischer Adressen End-zu-End Verbindungen zwischen Sender und Empfänger hergestellt werden



Einordnung im ISO/OSI-Modell



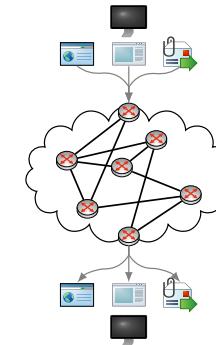
Aufgaben der Transportschicht

Die wesentlichen Aufgaben der Transportschicht sind

- ▶ Multiplexing von Datenströmen unterschiedlicher Anwendungen bzw. Anwendunginstanzen,
- ▶ Bereitstellung verbindungsloser und verbindungsorientierter Transportmechanismen und
- ▶ Mechanismen zur Stau- und Flusskontrolle.

Multiplexing:

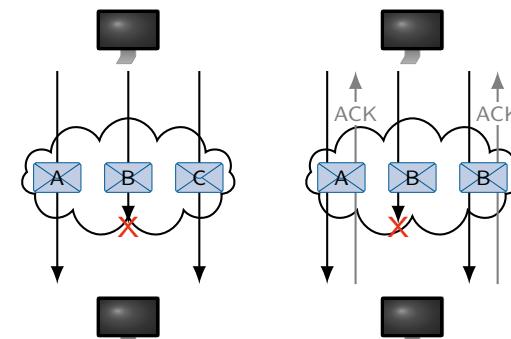
- ▶ Segmentierung der Datenströme unterschiedlicher Anwendungen (Browser, Chat, Email, ...)
- ▶ Segmente werden in jeweils unabhängigen IP-Paketen zum Empfänger geroutet
- ▶ Empfänger muss die Segmente den einzelnen Datenströmen zuordnen und an die jeweilige Anwendung weiterreichen



Aufgaben der Transportschicht

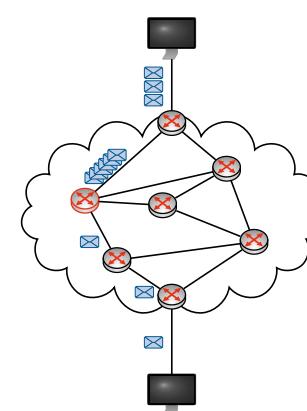
Transportdienste:

- ▶ Verbindungslos (**Best Effort**)
 - ▶ Segmente sind aus Sicht der Transportschicht voneinander unabhängig
 - ▶ Keine Sequenznummern, keine Übertragungswiederholung, keine Garantie der richtigen Reihenfolge
- ▶ Verbindungsorientiert
 - ▶ Übertragungswiederholung bei Fehlern
 - ▶ Garantie der richtigen Reihenfolge einzelner Segmente



Stau- und Flusskontrolle:

- ▶ Staukontrolle (**Congestion Control**)
 - ▶ Reaktion auf drohende Überlast im Netz
- ▶ Flusskontrolle (**Flow Control**)
 - ▶ Laststeuerung durch den Empfänger



Übersicht

Motivation

Multiplexing

Verbindungslose Übertragung

Verbindungsorientierte Übertragung

Sliding-Window-Verfahren

Transmission Control Protocol (TCP)

Fluss- und Staukontrolle bei TCP

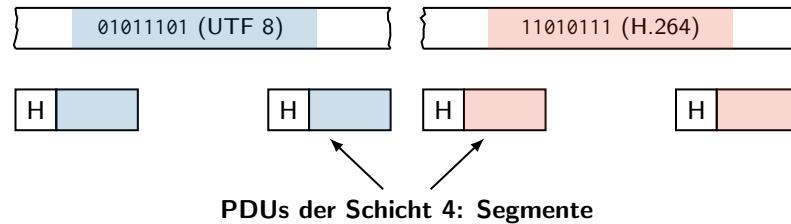
Network Address Translation (NAT)

Multiplexing



Auf der Transportschicht

1. werden die kodierten Datenströme in **Segmente** unterteilt und
2. jedes Segment mit einem Header versehen.



Ein solcher Header enthält jeweils mindestens

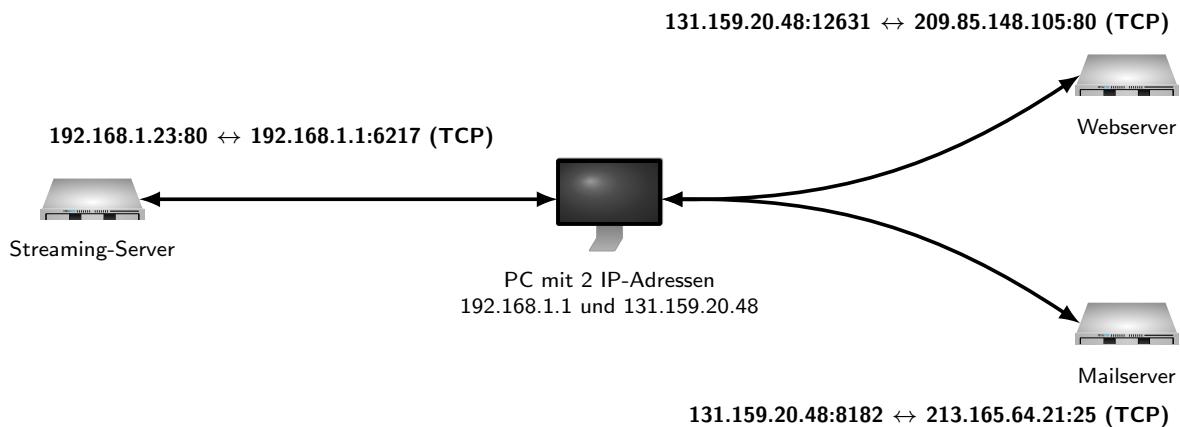
- ▶ einen **Quellport** und
- ▶ einen **Zielport**,

welche zusammen mit den IP-Adressen und dem verwendeten Transportprotokoll die Anwendung auf dem jeweiligen Host eindeutig identifizieren.

⇒ **5-Tupel** bestehend aus:

$$(\text{SrcIPAddr}, \text{SrcPort}, \text{DstIPAddr}, \text{DstPort}, \text{Protocol})$$

Beispiel:



- ▶ Portnummern sind bei den bekannten Transportprotokollen 16 bit lang.
- ▶ Betriebssysteme verwenden das 5-Tupel (IP-Adressen, Portnummern, Protokoll), um Anwendungen **Sockets** bereitzustellen.
- ▶ Eine Anwendung wiederum adressiert einen Socket mittels eines **File-Deskriptors** (ganzzahliger Wert).
- ▶ Verbindungsorientierte Sockets können nach dem Verbindungsaufbau sehr einfach genutzt werden, da der Empfänger bereits feststeht (Lesen und Schreiben mittels Systemaufrufen `read()` und `write()` möglich).
- ▶ Verbindungslose Sockets benötigen Adressangaben, an wen gesendet oder von wem empfangen werden soll (`sendto()` und `recvfrom()`).

Übersicht

Motivation

Multiplexing

Verbindungslose Übertragung

Verbindungsorientierte Übertragung

Sliding-Window-Verfahren

Transmission Control Protocol (TCP)

Fluss- und Staukontrolle bei TCP

Network Address Translation (NAT)

Verbindungslose Übertragung

Funktionsweise: Header eines Transportprotokolls besteht mind. aus

- ▶ Quell- und Zielport sowie
- ▶ einer Längenangabe der Nutzdaten.

Dies ermöglicht es einer Anwendung beim Senden für jedes einzelne Paket

- ▶ den Empfänger (IP-Adresse) und
- ▶ die empfangende Anwendung (Protokoll und Zielport) anzugeben.

Probleme: Da die Segmente unabhängig voneinander und aus Sicht der Transportschicht **zustandslos** versendet werden, kann nicht sichergestellt werden, dass

- ▶ Segmente den Empfänger erreichen (Pakete können verloren gehen) und
- ▶ der Empfänger die Segmente in der richtigen Reihenfolge erhält (Pakete werden unabhängig geroutet).

Folglich spricht man von einer **ungesicherten, verbindungslosen** oder **nachrichtenorientierten** Kommunikation. (Nicht zu verwechseln mit nachrichtenorientierter Übertragung auf Schicht 2)

Hinweise:

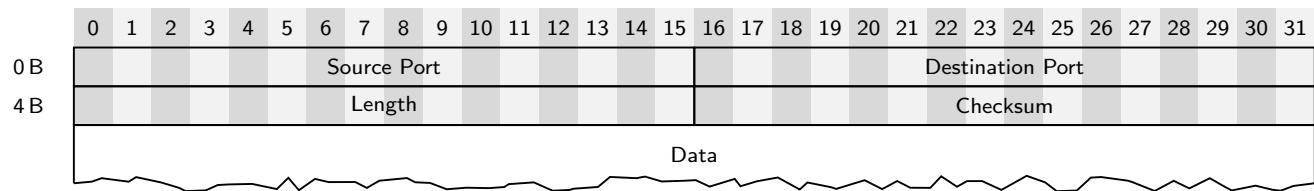
- ▶ Verbindungslose POSIX-Sockets werden mittels des Präprozessormakros SOCK_DGRAM identifiziert.
- ▶ DGRAM steht dabei für **Datagram**, worunter man schlicht eine Nachricht bestimmter Länge versteht, die aus Sicht der Transportschicht als Einheit übertragen werden soll.

Case Study: User Datagram Protocol (UDP)

Das **User Datagram Protocol (UDP)** ist eines der beiden am häufigsten verwendeten Transportprotokolle im Internet. Es bietet

- ▶ ungesicherte und nachrichtenorientierte Übertragung
- ▶ bei geringem Overhead.

UDP-Header:



- ▶ „Length“ gibt die Länge von Header und Daten in Vielfachen von Byte an.
- ▶ Die Checksumme erstreckt sich über Header und Daten.
 - ▶ Die UDP-Checksumme ist bei IPv4 optional, wird für IPv6 jedoch vorausgesetzt.
 - ▶ Wird sie nicht verwendet, wird das Feld auf 0 gesetzt.
 - ▶ Wird sie verwendet, wird zur Berechnung ein **Pseudo-Header** genutzt (eine Art „Default-IP-Header“ der nur zur Berechnung der Prüfsumme dient). Er beinhaltet folgende Felder des IP-Headers: Quell- und Ziel-IP-Adresse, ein 8-Bit-Feld mit Nullen, Protocol-ID und Länge des UDP-Datagramms.

Vorteile von UDP:

- ▶ Geringer Overhead
- ▶ Keine Verzögerung durch Verbindungsaufbau oder Retransmits und Reordering von Segmenten
- ▶ Gut geeignet für Echtzeitanwendungen (Voice over IP, Online-Spiele) sofern gelegentlicher Paketverlust in Kauf genommen werden kann
- ▶ Keine Beeinflussung der Datenrate durch Fluss- und Staukontrollmechanismen (kann Vorteile haben, siehe Übung)

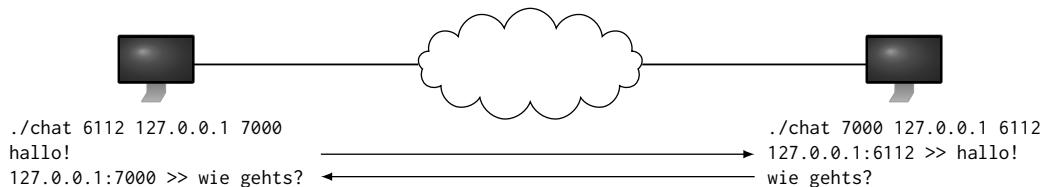
Nachteile von UDP:

- ▶ Keine Zusicherung irgendeiner Form von Dienstqualität (beliebig hohe Fehlerrate)
- ▶ Datagramme können out-of-order ausgeliefert werden (beispielsweise bei Verwendung mehrerer Pfade zu einem Ziel)
- ▶ Keine Flusskontrolle (schneller Sender kann langsamen Empfänger überfordern)
- ▶ Keine Staukontrollmechanismen (Überlast im Netz führt zu hohen Verlustraten)

Case Study: UDP-Chat

Was wir wollen:

- ▶ Eine Anwendung, die gleichzeitig als Client und Server arbeiten soll (P2P-Modell)
- ▶ Nur 1:1-Verbindungen, also keine Gruppen-Chats



Was wir brauchen: Einen Socket,

- ▶ auf dem ausgehende Nachrichten gesendet werden (an Ziel-IP und Ziel-Port) und
- ▶ der an die lokale(n) IP(s) und Portnummer gebunden wird, um Nachrichten empfangen zu können

Welche Sprache?

- ▶ C natürlich (;

Download

- ▶ `wget http://grnvs.net/udpchat.tar`
- ▶ `tar xvf udpchat.tar`
- ▶ `cd udpchat; make`

Wichtige structs

- ▶ `struct sockaddr_in`: Enthält Adressfamilie (AF_INET oder AF_INET6), Portnummer und IP-Adresse.

```
struct sockaddr_in {  
    __kernel_sa_family_t sin_family; /* Address family */  
    __be16                sin_port;   /* Port number */  
    struct in_addr         sin_addr;   /* Internet address */  
  
    /* Pad to size of struct sockaddr. */  
    unsigned char __pad[__SOCK_SIZE__ - sizeof(short int) -  
                      sizeof(unsigned short int) - sizeof(struct in_addr)];  
};
```

- ▶ `struct sockaddr_in`: Repräsentiert eine IPv4-Adresse in Network Byte Order.

```
struct in_addr {  
    __be32    s_addr;  
};
```

- ▶ `struct sockaddr`: Adress-Struktur, die den Typ des Sockets offen lässt (generalisiert sockaddr_in und sockaddr_un).

```
struct sockaddr {  
    sa_family_t      sa_family;     /* address family, AF_xxx */  
    char            sa_data[14];   /* 14 bytes of protocol address */  
};
```

Code für unser Programm:

```
struct sockaddr_in local;  
local.sin_family      = AF_INET;  
local.sin_port        = htons(LOCALPORT); // anwendungsspezifischer Port  
local.sin_addr.s_addr= INADDR_ANY;    // empfange von allen Adressen
```

Sockets

- ▶ Aus Sicht des Betriebssystems ist ein Socket nichts weiter als ein **Filedescriptor**, d. h. ein Integer.
- ▶ Sockets stellen die Schnittstelle zwischen einem Programm (unserer Chatanwendung) und dem Betriebssystem dar.

Ein Socket für unser Programm:

```
int sd;
if (0 > (sd=socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP))) {
    perror("socket() failed");
    exit(1);
}
```

- ▶ **socket()** erzeugt einen neuen Socket vom angegebenen Typ:
 - ▶ **AF_INET** spezifiziert einen IPv4 Socket.
 - ▶ **SOCK_DGRAM** gibt an, dass es ein datagram-oriented Socket sein soll.
 - ▶ **IPPROTO_UDP** gibt das Transportprotokoll an.
- ▶ Rückgabewert ist der Socketdescriptor oder -1 bei einem Fehler.

Der Socket muss noch eine Adresse bekommen:

```
if (0 > bind(sd,(struct sockaddr *)&local,sizeof(local))) {
    perror("bind() failed");
    exit(1);
}
```

- ▶ **bind()** assoziert einen Filedescriptor mit den zugehörigen Adressinformationen.
- ▶ Der Cast in **struct sockaddr** ist deswegen notwendig, da **bind()** nicht nur mit **struct sockaddr_in** zurechtkommt.
- ▶ Rückgabewert ist 0 bei Erfolg und -1 bei Fehlern.

Wie merkt unser Programm, wenn neue Daten ankommen?

Hier gibt es 3 Möglichkeiten:

- ▶ Einfach ein **read()** auf dem Socket:
 - ▶ **read()** blockiert, solange bis etwas kommt.
 - ▶ Mit einem einzelnen Prozess bzw. Thread können wir so nur einen einzigen Socket überwachen.
 - ▶ Unser Programm würde nicht einmal auf Tastatureingaben reagieren.
- ▶ Der scheinbar komplizierte Weg über **select()** oder **pselect()**:
 - ▶ Wir packen alle Filedescriptors, die überwacht werden sollen, in ein **fd_set**.
 - ▶ Wir übergeben **select()** dieses Set.
 - ▶ Sobald etwas passiert, modifiziert **select()** das übergebene **fd_set**, so dass es genau die Filedescriptors enthält, die bereit geworden sind.
 - ▶ Rückgabewert von **select()** ist die Anzahl bereitgewordener Filedescriptors oder -1 bei einem Fehler.
- ▶ Bei einer großen Anzahl von Filedescriptors wird **select()** ggf. ineffizient. Hier bietet sich dann **epoll()** an (hier nicht weiter behandelt).

Ein **select()** für unser Programm:

```
fd_set rdfs, rfd;
FD_ZERO(&rdfs);
FD_SET(STDIN_FILENO,&rdfs);
FD_SET(sd,&rdfs);
maxfd = MAX(sd,STDIN_FILENO);

for (;;) {
    rfd = rdfs;
    if (0 > select(maxfd+1,&rfds,NULL,NULL,NULL)) {
        perror("select() failed");
        exit(1);
    }
    (...)
```

Empfangen von Daten

- ▶ Sobald etwas Interessantes passiert, wird `select()` uns das sagen.
- ▶ Wir müssen feststellen, welcher der File-Deskriptoren bereit ist.
- ▶ Im Fall der Standardeingabe (STDIN) können wir mit `fgets()` einfach die Eingabe lesen.
- ▶ Wenn der File-Descriptor des Sockets bereit ist, könnten wir `read()` verwenden. Dann werden wir aber bei verbindungslosen Transportprotokollen wie UDP nie erfahren, wer uns etwas geschickt hat.
- ▶ Besser wir nutzen `recvfrom()`: Hier können wir ein `struct sockaddr_in` übergeben, in das uns `recvfrom()` reinschreibt, von wem wir etwas empfangen haben.

Ein `recvfrom()` für unser Programm:

```
for (;;) {
    (...)

    if (FD_ISSET(sd,&rfd)) {
        len = recvfrom(sd,buffer,BUFFLEN-1,0,
                       (struct sockaddr *)&from,&slen));
        len = recvfrom(sd,buffer,BUFFLEN-1,0,(struct sockaddr*)&from,
                       &slen));
        if (0 > len)
            perror("recvfrom() failed");
            exit(1);
    }
    fprintf(stdout,"%s:%d >> %s\n",inet_ntoa(from.sin_addr),
            ntohs(from.sin_port),buffer);
}
(...)
```

Senden von Daten

- ▶ Um Daten zu Senden, müssen wir mit verbindungslosen Protokollen `sendto()` nutzen.
- ▶ Diesem muss man ein `struct sockaddr_in` übergeben, in dem steht, wer der Empfänger sein soll.
- ▶ Ein einfaches `write()` funktioniert nicht, da das Betriebssystem dann nicht weiß, an wen es die Daten senden soll.

Ein `sendto()` für unser Programm:

```
for (;;) {
    (...)

    if (FD_ISSET(STDIN_FILENO,&rfd)) {
        if (NULL == (s=fgets(buffer,BUFFLEN,stdin)))
            continue;

        len = sendto(sd,buffer,strlen(buffer),0,
                     (struct sockaddr *)&remote,sizeof(remote));

        if (0 > len)
            perror("sendto() failed");
            exit(1);
    }
}
(...)
```

Für alle, die mit der Indentation der Codebeispiele unglücklich sind, sei der [Linux kernel coding style Pflichtlektüre!](#)

Übersicht

Motivation

Multiplexing

Verbindungslose Übertragung

Verbindungsorientierte Übertragung

Sliding-Window-Verfahren

Transmission Control Protocol (TCP)

Fluss- und Staukontrolle bei TCP

Network Address Translation (NAT)

Verbindungsorientierte Übertragung

Grundlegende Idee: Linear durchnummeriere Segmente mittels **Sequenznummern** im Protokollheader
Sequenznummern ermöglichen insbesondere

- ▶ **Bestätigung** erfolgreich übertragener Segmente,
- ▶ **Identifikation** fehlender Segmente,
- ▶ **erneutes Anfordern** fehlender Segmente und
- ▶ **Zusammensetzen** der Segmente in der **richtigen Reihenfolge**.

Probleme: Sender und Empfänger müssen

- ▶ sich zunächst synchronisieren (Austausch der initialen Sequenznummern) und
- ▶ Zustand halten (aktuelle Sequenznummer, bereits bestätigte Segmente, ...).

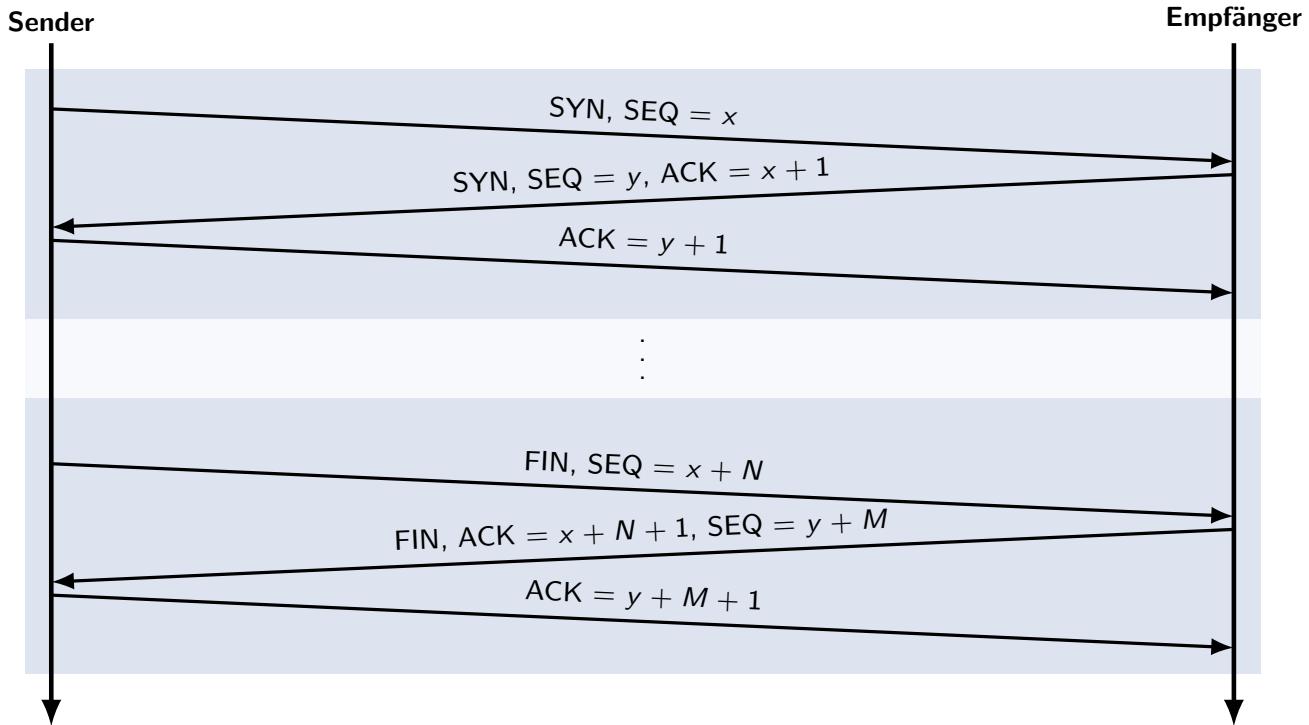
Verbindungsphasen:

1. **Verbindungsaufbau (Handshake)**
2. **Datenübertragung**
3. **Verbindungsabbau (Teardown)**

Vereinbarungen: Wir gehen zunächst davon aus,

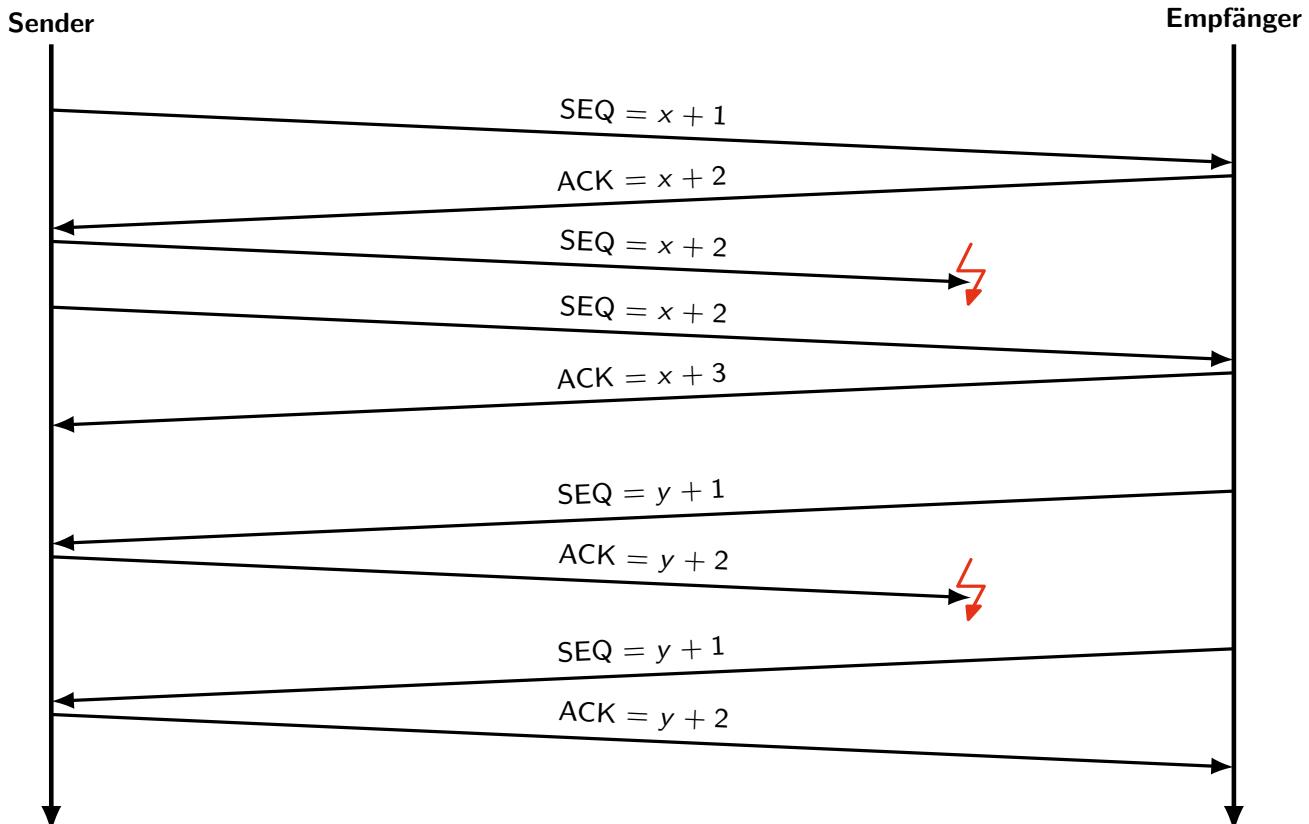
- ▶ dass stets ganze Segmente bestätigt werden und
- ▶ dass in einer Quittung das nächste erwartete Segment angegeben wird.

Beispiel: Aufbau und Abbau einer Verbindung



Diese Art des Verbindungsauftbaus bezeichnet man als [3-Way-Handshake](#).

Beispiel: Übertragungsphase



Sliding-Window Verfahren

Bislang:

- ▶ Im vorherigen Beispiel hat der Sender stets nur ein Segment gesendet und dann auf eine Bestätigung gewartet
- ▶ Dieses Verfahren ist ineffizient, da abhängig von der Umlaufverzögerung (Round Trip Time, RTT) zwischen Sender und Empfänger viel Bandbreite ungenutzt bleibt („Stop and Wait“-Verfahren)

Idee: Teile dem Sender mit, wie viele Segmente **nach** dem letzten bestätigten Segment auf einmal übertragen werden dürfen, ohne dass der Sender auf eine Bestätigung warten muss.

Vorteile:

- ▶ Zeit zwischen dem Absenden eines Segments und dem Eintreffen einer Bestätigung kann effizienter genutzt werden
- ▶ Durch die Aushandlung dieser **Fenstergrößen** kann der Empfänger die Datenrate steuern → **Flusskontrolle**
- ▶ Durch algorithmische Anpassung der Fenstergröße kann die Datenrate an die verfügbare Datenrate auf dem Übertragungspfad zwischen Sender und Empfänger angepasst werden → **Staukontrolle**

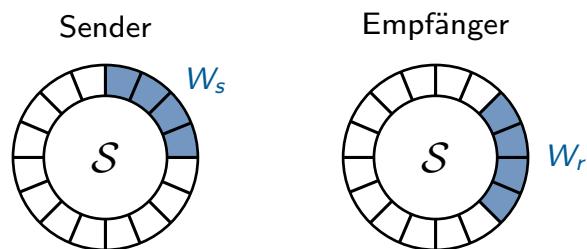
Probleme:

- ▶ Sender und Empfänger müssen mehr Zustand halten
(Was wurde bereits empfangen? Was wird als nächstes erwartet?)
- ▶ Der Sequenznummernraum ist endlich → Wie werden Missverständnisse verhindert?

Zur Notation:

- ▶ Sender und Empfänger haben denselben Sequenznummernraum $\mathcal{S} = \{0, 1, 2, \dots, N - 1\}$.

Beispiel: $N = 16$:



- ▶ Sendefenster (**Send Window**) $W_s \subset \mathcal{S}, |W_s| = w_s$:
Es dürfen w_s Segmente nach dem letzten bestätigten Segment auf einmal gesendet werden.
- ▶ Empfangsfenster (**Receive Window**) $W_r \subset \mathcal{S}, |W_r| = w_r$:
Sequenznummern der Segmente, die als nächstes akzeptiert werden.
- ▶ Sende- und Empfangsfenster „verschieben“ und überlappen sich während des Datenaustauschs.

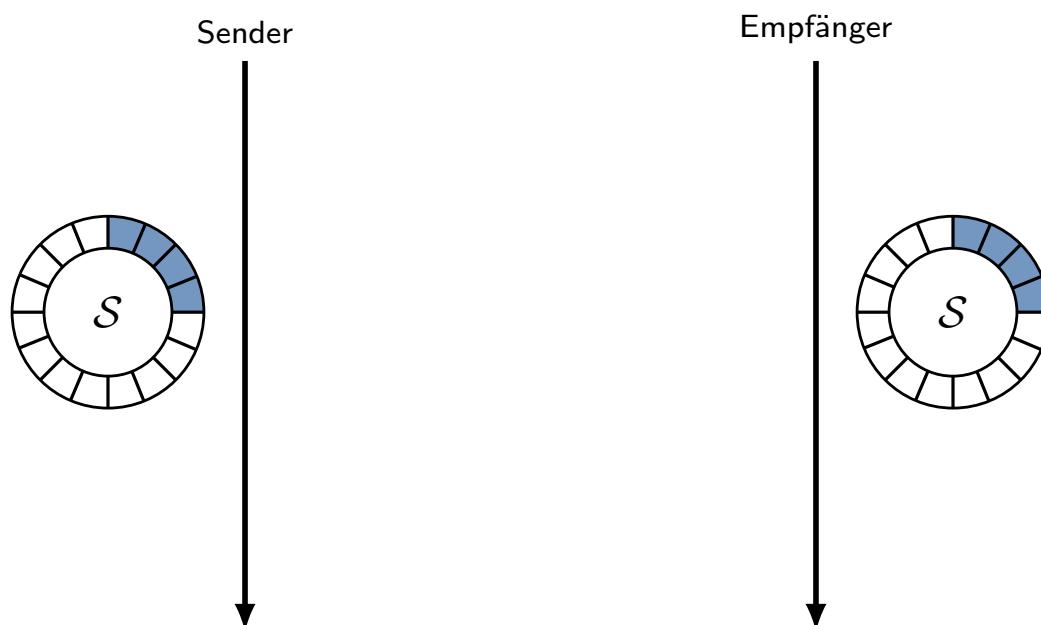
Vereinbarungen:

- ▶ Eine Bestätigung $\text{ACK} = m + 1$ bestätigt alle Segmente mit $\text{SEQ} \leq m$. Dies wird als **kumulative Bestätigung** bezeichnet.
- ▶ Gewöhnlich löst **jedes erfolgreich empfangene Segment** das Senden einer Bestätigung aus, wobei stets das **nächste erwartete Segment** bestätigt wird. Dies wird als **Forward Acknowledgement** bezeichnet.

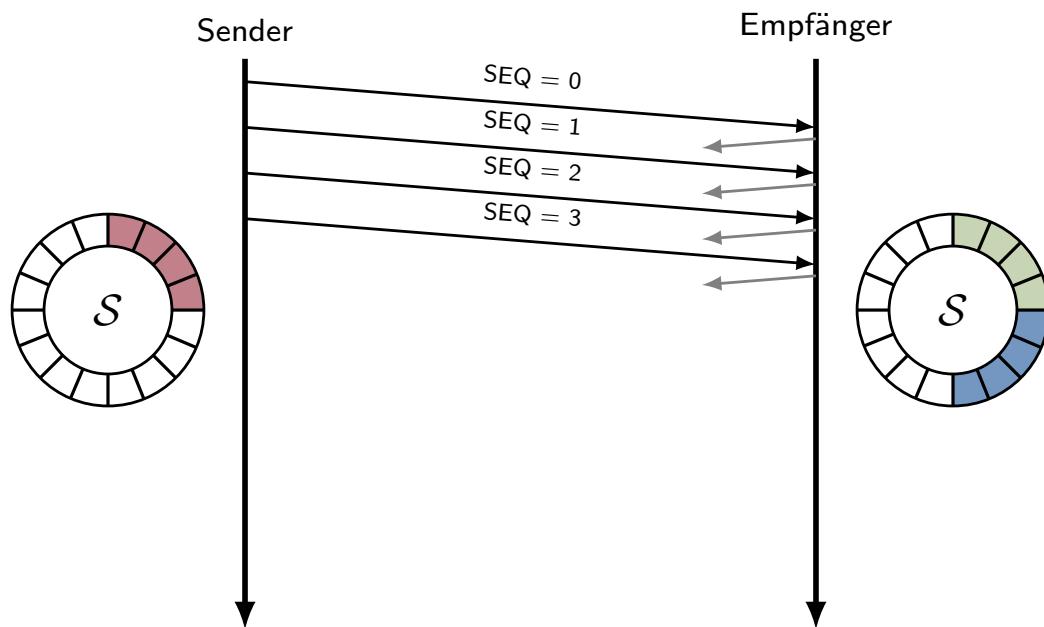
Wichtig:

- ▶ In den folgenden Grafiken sind die meisten Bestätigungen zwecks Übersichtlichkeit nur angedeutet (graue Pfeile).
- ▶ Die Auswirkungen auf Sende- und Empfangsfenster beziehen sich nur auf den Erhalt der schwarz eingezeichneten Bestätigungen.
- ▶ Dies ist äquivalent zur Annahme, dass die angedeuteten Bestätigungen verloren gehen.

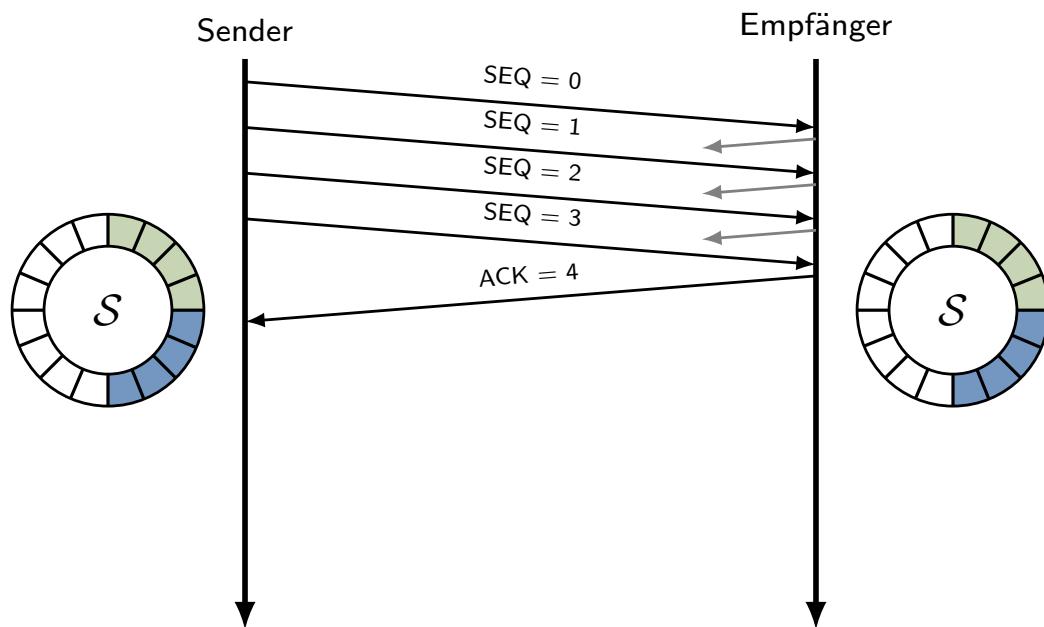
- Sendefenster W_s bzw. Empfangsfenster W_r
- gesendet aber noch nicht bestätigt
- gesendet und bestätigt/empfangen



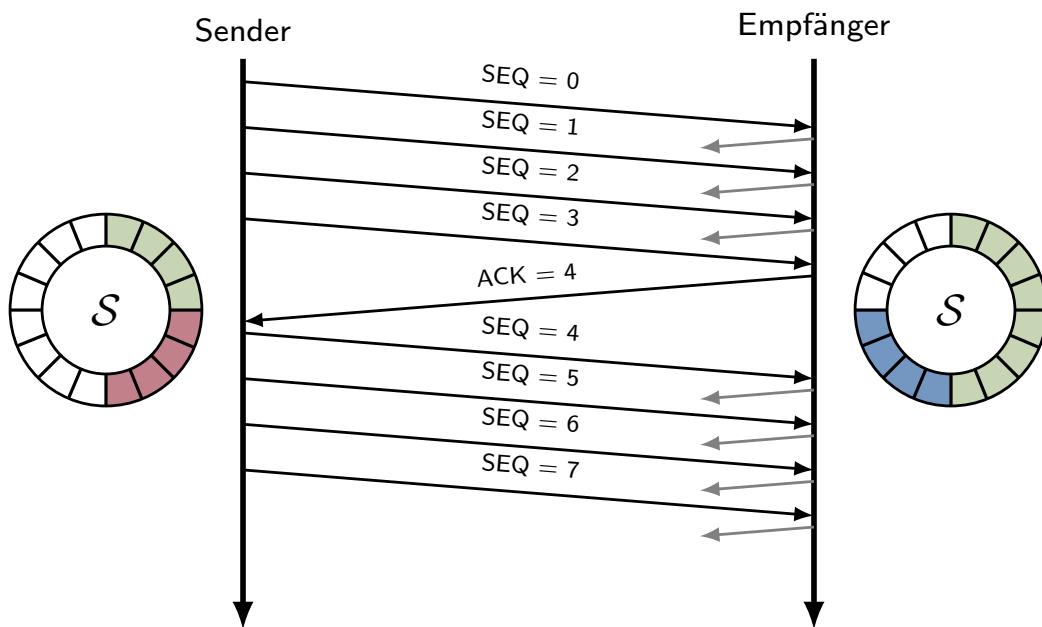
- Sendefenster W_s bzw. Empfangsfenster W_r
- gesendet aber noch nicht bestätigt
- gesendet und bestätigt/empfangen



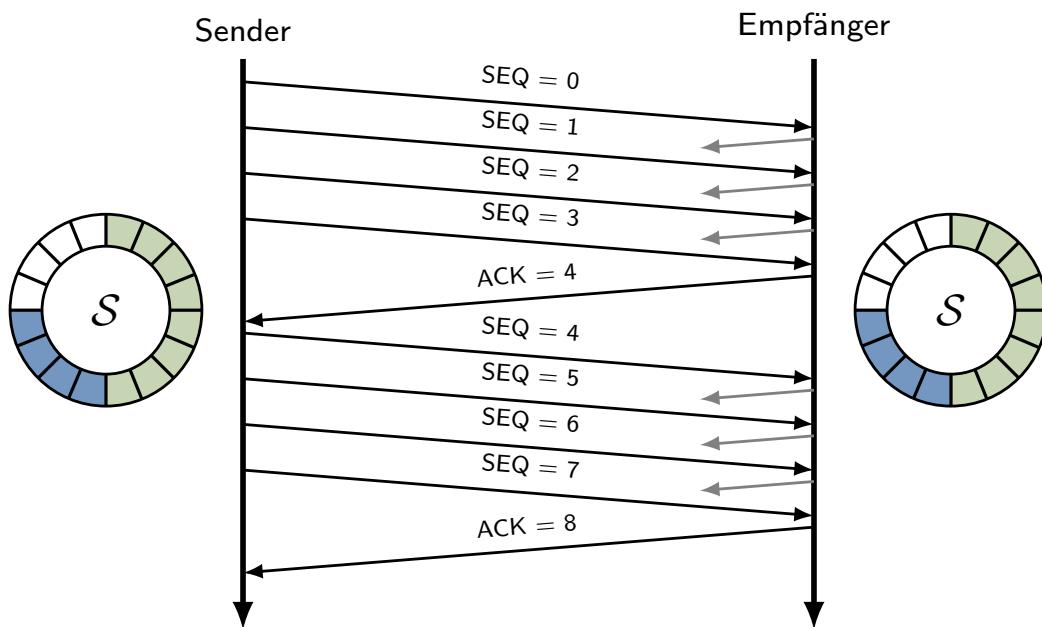
- Sendefenster W_s bzw. Empfangsfenster W_r
- gesendet aber noch nicht bestätigt
- gesendet und bestätigt/empfangen



- Sendefenster W_s bzw. Empfangsfenster W_r
- gesendet aber noch nicht bestätigt
- gesendet und bestätigt/empfangen



- Sendefenster W_s bzw. Empfangsfenster W_r
- gesendet aber noch nicht bestätigt
- gesendet und bestätigt/empfangen



Neues Problem: Wie wird jetzt mit Segmentverlusten umgegangen?

Zwei Möglichkeiten:

1. Go-Back-N

- ▶ Akzeptiere stets nur die nächste erwartete Sequenznummer
- ▶ Alle anderen Segmente werden verworfen

2. Selective-Repeat

- ▶ Akzeptiere alle Sequenznummern, die in das aktuelle Empfangsfenster fallen
- ▶ Diese müssen gepuffert werden, bis fehlende Segmente erneut übertragen wurden

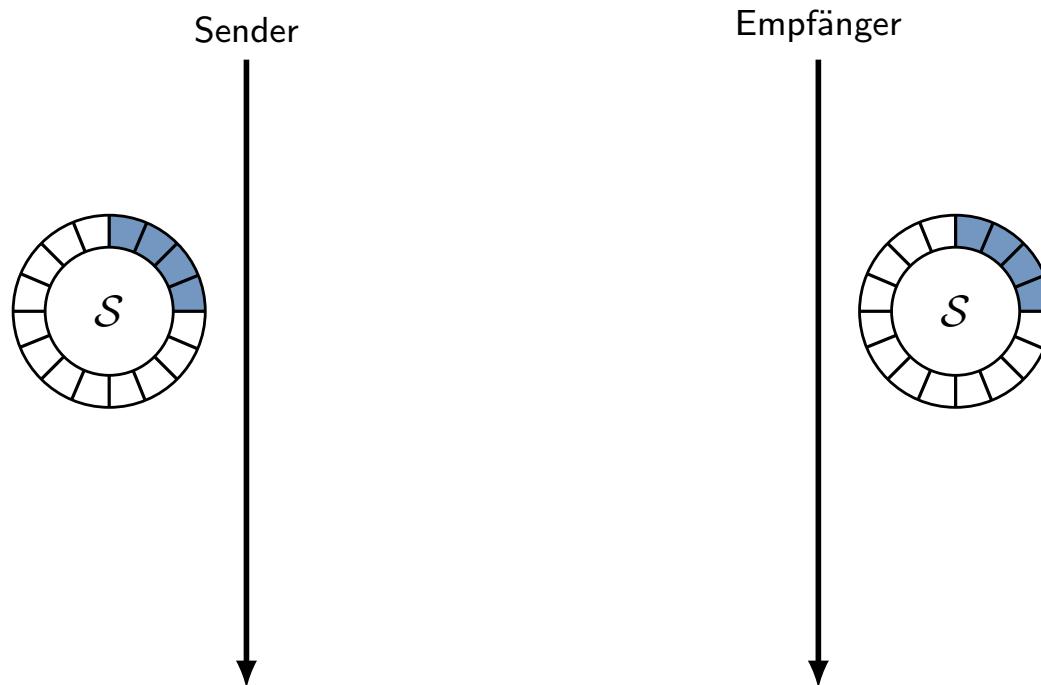
Wichtig:

- ▶ In beiden Fällen muss der Sequenznummernraum so gewählt werden, dass wiederholte Segmente eindeutig von neuen Segmenten unterschieden werden können.
- ▶ Andernfalls würde es zu Verwechslungen kommen
→ Auslieferung von Duplikaten an höhere Schichten, keine korrekte Reihenfolge.

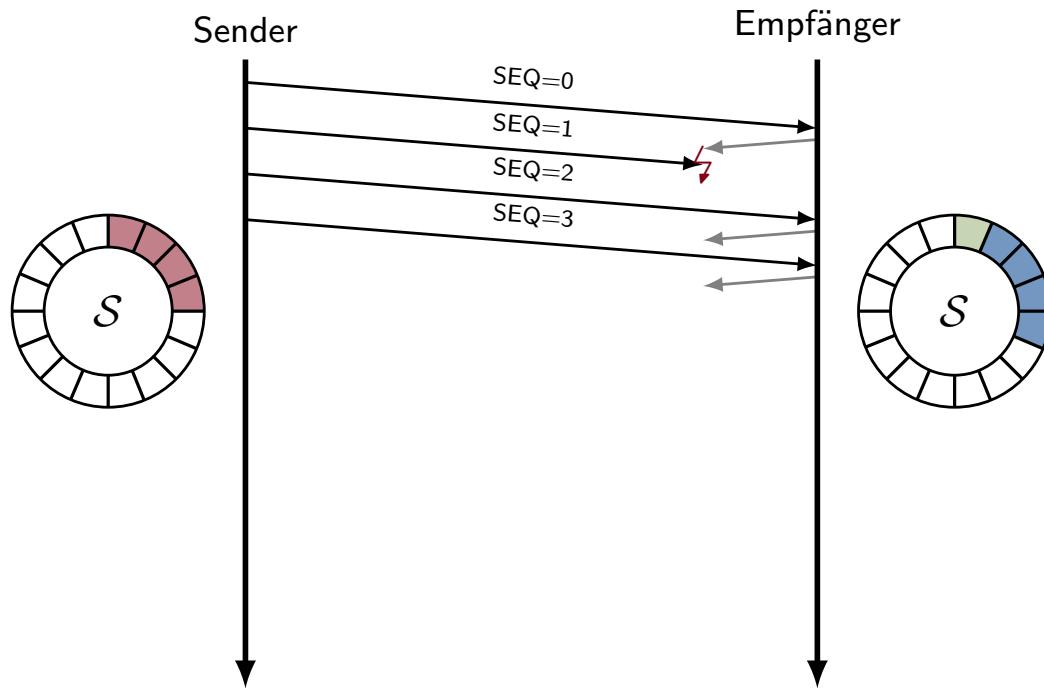
Frage: (siehe Übung)

Wie groß darf das Sendefenster W_s in Abhängigkeit des Sequenznummernraums \mathcal{S} höchstens gewählt werden, so dass die Verfahren funktionieren?

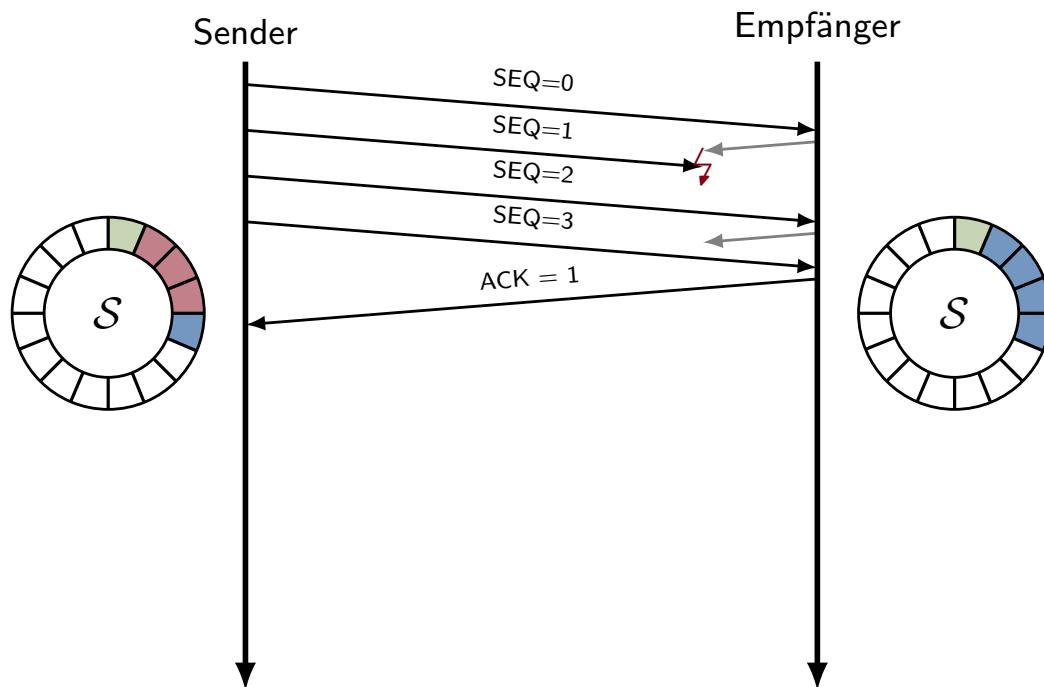
Go-Back-N: $N = 16$, $w_s = 4$, $w_r = 4$



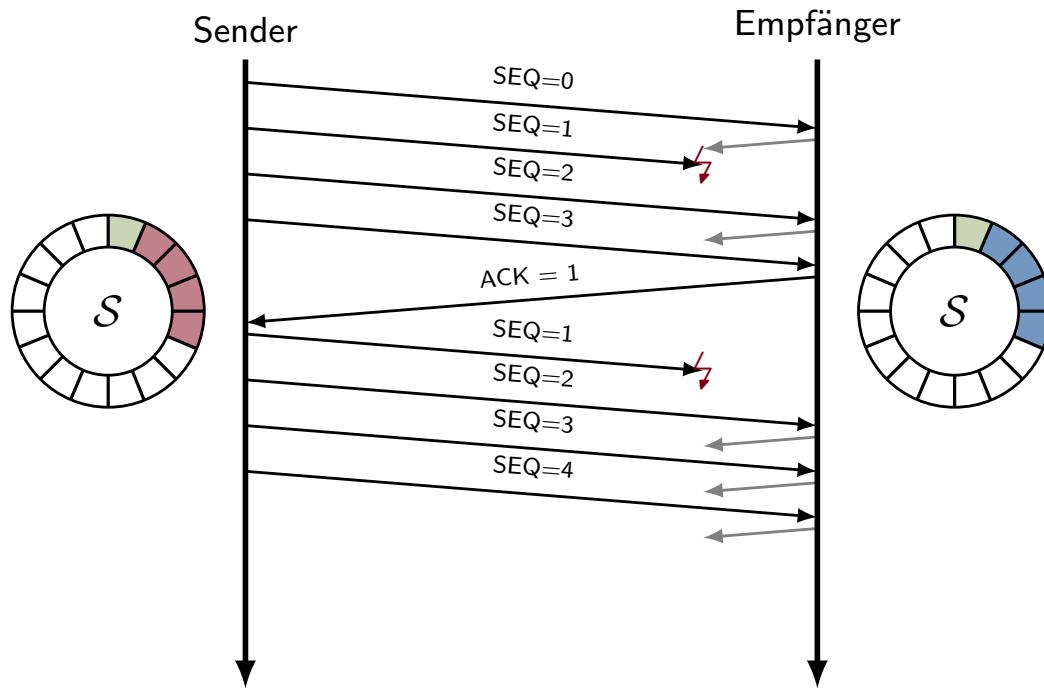
Go-Back-N: $N = 16$, $w_s = 4$, $w_r = 4$



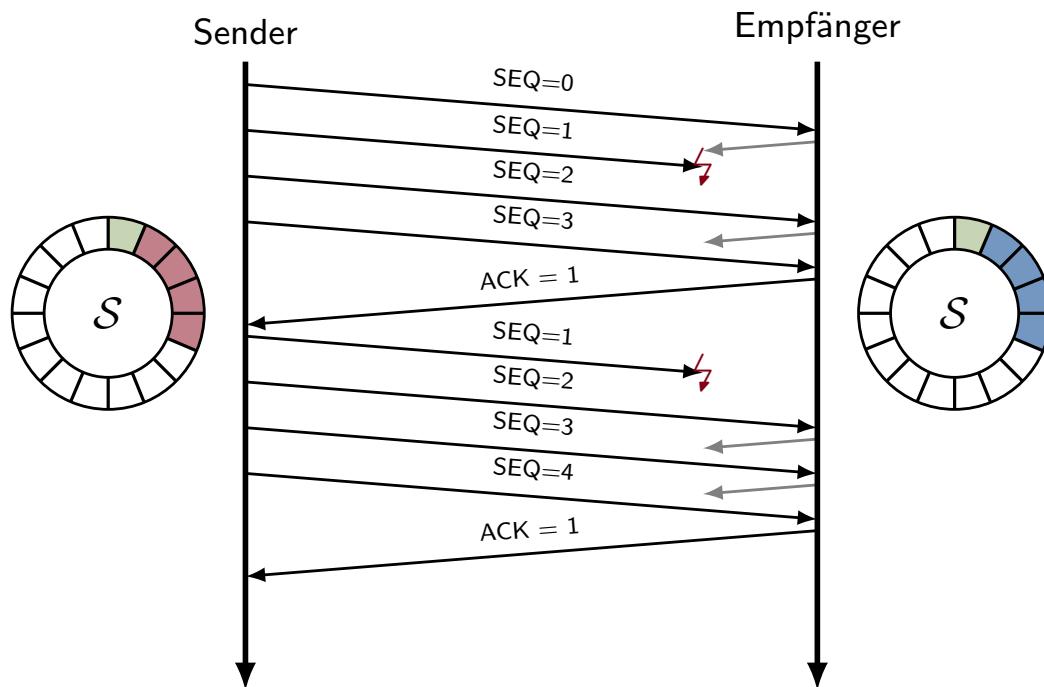
Go-Back-N: $N = 16$, $w_s = 4$, $w_r = 4$



Go-Back-N: $N = 16$, $w_s = 4$, $w_r = 4$



Go-Back-N: $N = 16$, $w_s = 4$, $w_r = 4$



Anmerkungen zu Go-Back-N

- ▶ Da der Empfänger stets nur das nächste erwartete Segment akzeptiert, reicht ein Empfangsfenster der Größe $w_r = 1$ prinzipiell aus. Unabhängig davon muss für praktische Implementierungen ein ausreichend großer Empfangspuffer verfügbar sein.

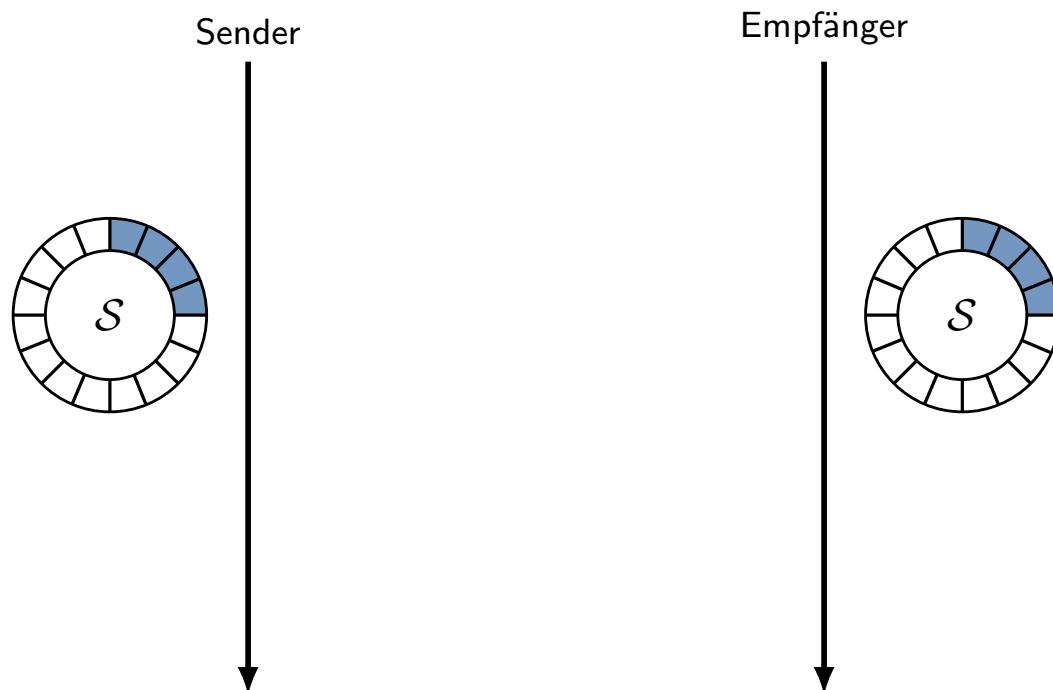
- ▶ Bei einem Sequenznummernraum der Kardinalität N muss für das Sendefenster stets gelten:

$$w_s \leq N - 1.$$

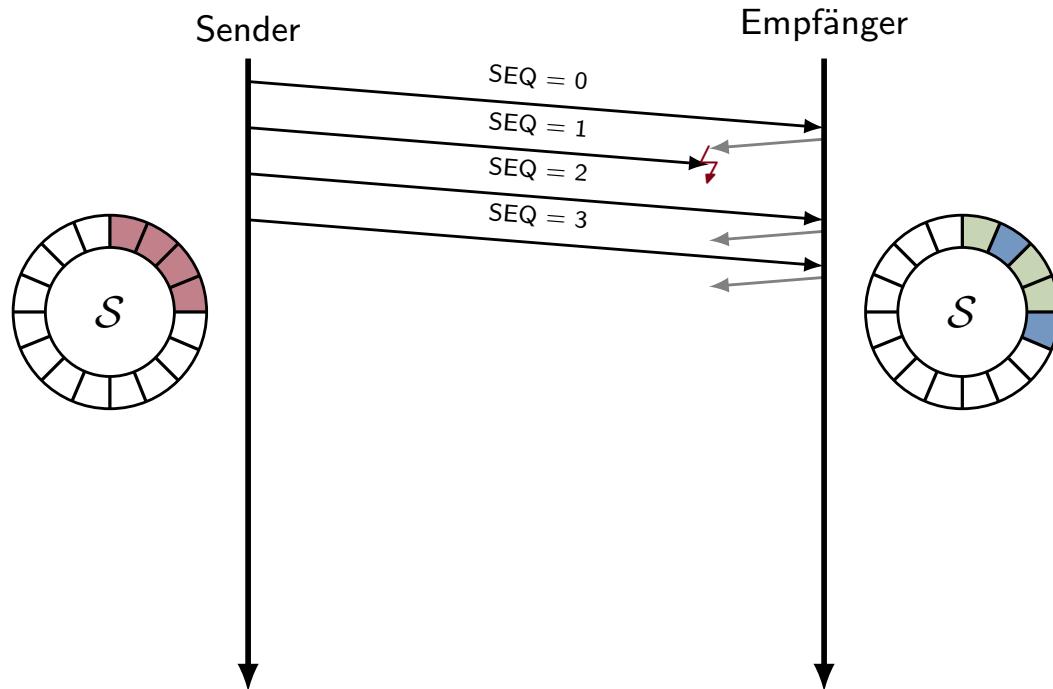
Andernfalls kann es zu Verwechslungen kommen (s. Übung).

- ▶ Das Verwerfen erfolgreich übertragener aber nicht in der erwarteten Reihenfolge eintreffender Segmente macht das Verfahren einfach zu implementieren aber weniger effizient.

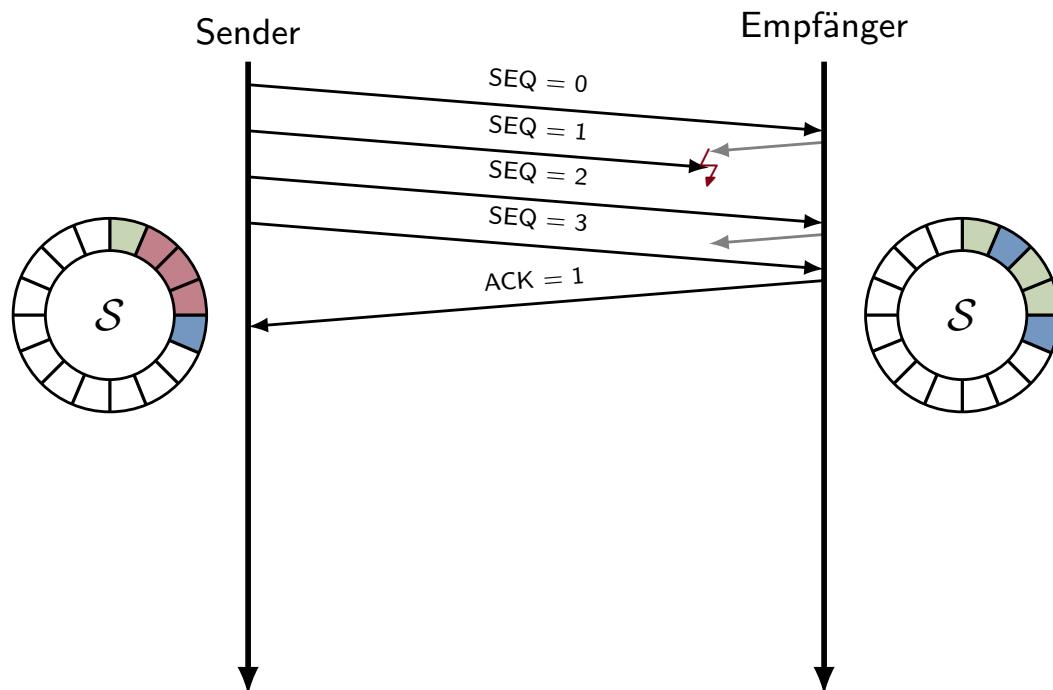
Selective Repeat: $N = 16$, $w_s = 4$, $w_r = 4$



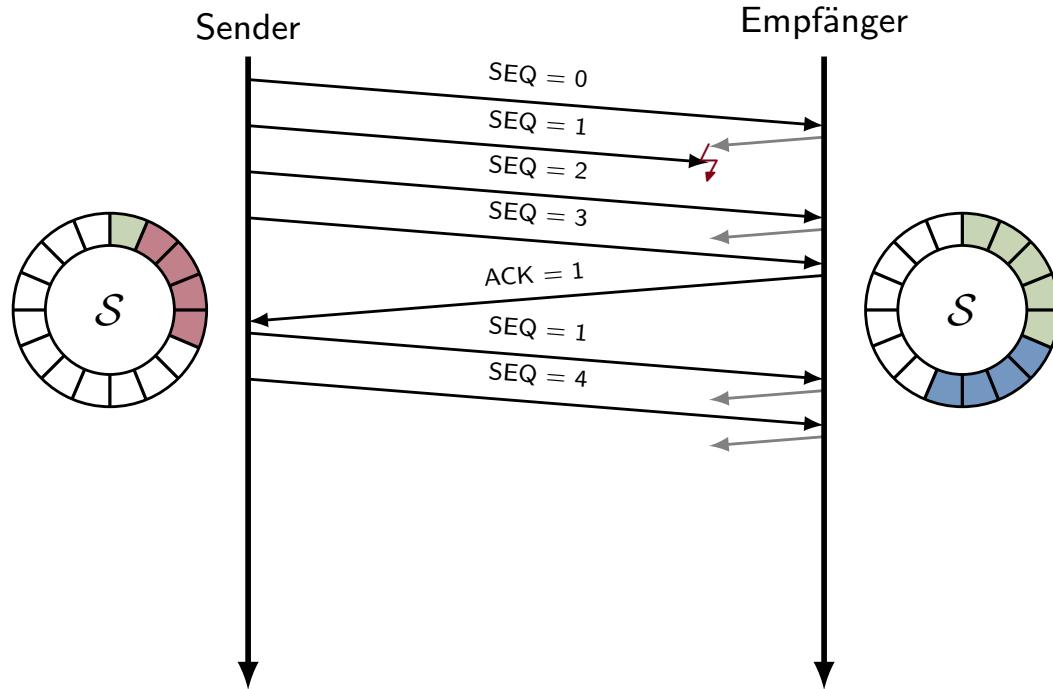
Selective Repeat: $N = 16$, $w_s = 4$, $w_r = 4$



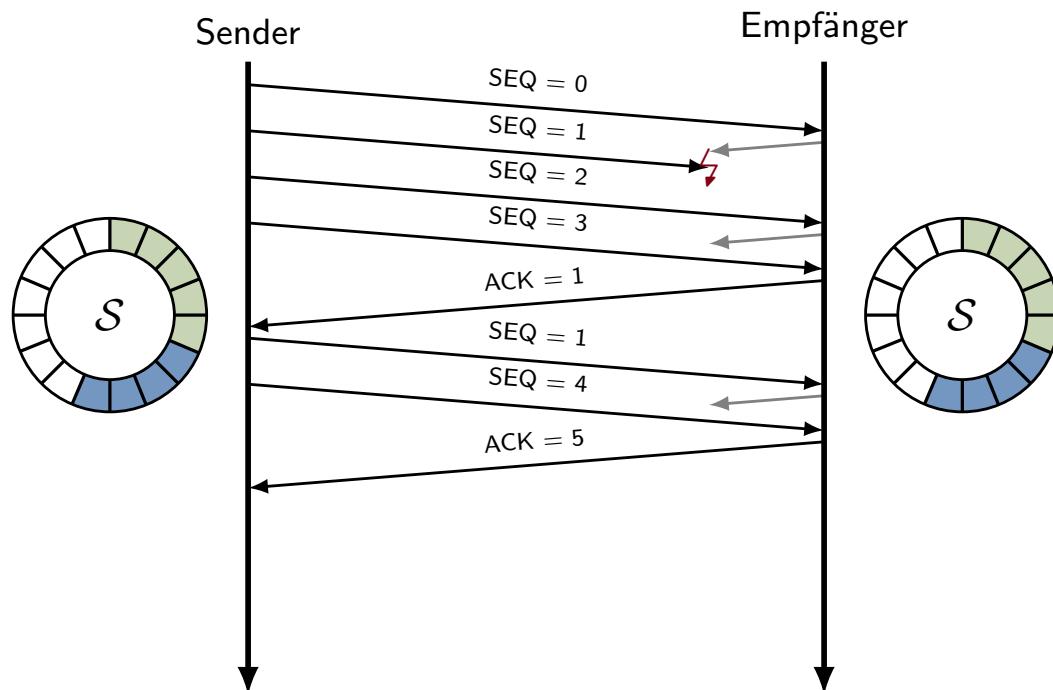
Selective Repeat: $N = 16$, $w_s = 4$, $w_r = 4$



Selective Repeat: $N = 16$, $w_s = 4$, $w_r = 4$



Selective Repeat: $N = 16$, $w_s = 4$, $w_r = 4$



Anmerkungen zu Selective Repeat

- ▶ Wählt man $w_r = 1$ und w_s unabhängig von w_r , so degeneriert Selective Repeat zu Go-Back-N.
- ▶ Bei einem Sequenznummernraum der Kardinalität N muss für das Sendefenster stets gelten:

$$w_s \leq \left\lfloor \frac{N}{2} \right\rfloor.$$

Andernfalls kann es zu Verwechslungen kommen (s. Übung).

Allgemeine Anmerkungen

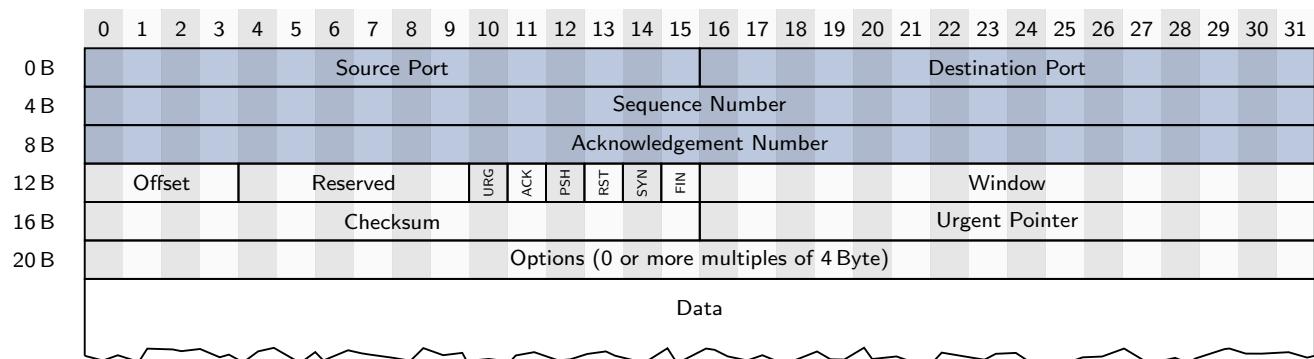
- ▶ Bei einer Umsetzung dieser Konzepte benötigt insbesondere der Empfänger einen **Empfangspuffer**, dessen Größe an die Sende- und Empfangsfenster angepasst ist.
- ▶ Für praktische Anwendungen werden die Größen von W_s und W_r dynamisch angepasst (siehe Case Study zu TCP), wodurch Algorithmen zur **Staukontrolle** und **Flusskontrolle** auf Schicht 4 ermöglicht werden.

Transmission Control Protocol (TCP)

Das **Transmission Control Protocol (TCP)** ist das dominierende Transportprotokoll im Internet (rund 90 % des Datenverkehrs im Internet [1]). Es bietet

- ▶ gesicherte/stromorientierte Übertragung mittels Sliding-Window und Selective Repeat sowie
- ▶ Mechanismen zur Fluss- und Staukontrolle.

TCP-Header:



- ▶ Quell- und Zielport werden analog zu UDP verwendet.
- ▶ Sequenz- und Bestätigungsnummer dienen der gesicherten Übertragung. Es werden bei TCP **nicht** ganze Segmente sondern einzelne Bytes bestätigt (stromorientierte Übertragung).

Transmission Control Protocol (TCP)

(Data) Offset

- ▶ Gibt die Länge des TCP-Headers in Vielfachen von 4 B an.
- ▶ Der TCP-Header hat variable Länge (Optionen, vgl. IPv4-Header).

Reserved

- ▶ Hat in bisherigen TCP-Versionen keine Verwendung. Muss auf 0 gesetzt werden, so dass zukünftige TCP-Versionen bei Bedarf das Feld nutzen können.

Flag URG („urgent“) (selten verwendet)

- ▶ Ist das Flag gesetzt, werden die Daten im aktuellen TCP-Segment beginnend mit dem ersten Byte bis zu der Stelle, an die das Feld **Urgent Pointer** zeigt, sofort an höhere Schichten weitergeleitet.

Flag ACK („acknowledgement“)

- ▶ Ist das Flag gesetzt, handelt es sich um eine Empfangsbestätigung.
- ▶ Bestätigungen können bei TCP auch „huckepack“ (engl. **piggy backing**) übertragen werden, d. h. es werden gleichzeitig Nutzdaten von *A* nach *B* übertragen und ein zuvor von *B* nach *A* gesendetes Segment bestätigt.
- ▶ Die Acknowledgement-Number gibt bei TCP stets **das nächste erwartete Byte** an.

Flag PSH („push“)

- ▶ Ist das Flag gesetzt, werden sende- und empfangsseitige Puffer des TCP-Stacks umgangen.
- ▶ Sinnvoll für interaktive Anwendungen (z. B. **Telnet**-Verbindungen).

Flag RST („reset“)

- ▶ Dient dem Abbruch einer TCP-Verbindung ohne ordnungsgemäßen Verbindungsabbau.

Transmission Control Protocol (TCP)

Flag SYN („synchronization“)

- ▶ Ist das Flag gesetzt, handelt es sich um ein Segment, welches zum Verbindungsaufbau gehört (initialer Austausch von Sequenznummern).
- ▶ Ein gesetztes SYN-Flag inkrementiert Sequenz- und Bestätigungsnummern um 1 obwohl keine Nutzdaten transportiert werden.

Flag FIN („finish“)

- ▶ Ist das Flag gesetzt, handelt es sich um ein Segment, welches zum Verbindungsabbau gehört.
- ▶ Ein gesetztes FIN-Flag inkrementiert Sequenz- und Bestätigungsnummern um 1 obwohl keine Nutzdaten transportiert werden.

Receive Window

- ▶ Größe des aktuellen Empfangsfensters W_r in Byte.
- ▶ Ermöglicht es dem Empfänger, die Datenrate des Senders zu drosseln.

Checksum

- ▶ Checksumme über Header und Daten.
- ▶ Wie bei UDP wird zur Berechnung ein **Pseudo-Header** verwendet.

Urgent Pointer (selten verwendet)

- ▶ Gibt das Ende der „Urgent-Daten“ an, welche unmittelbar nach dem Header beginnen und bei gesetztem URG-Flag sofort an höhere Schichten weitergereicht werden sollen.

Options

- ▶ Zusätzliche Optionen, z. B. **Window Scaling** (s. Übung), selektive Bestätigungen oder Angabe der **Maximum Segment Size (MSS)**.

Anmerkungen zur MSS

- ▶ Die MSS gibt die maximale Größe eines TCP-Segments (Nutzdaten ohne TCP-Header) an.
- ▶ Zum Vergleich gibt die MTU (Maximum Transfer Unit) die maximale Größe der Nutzdaten aus Sicht von Schicht 2 an (alles einschließlich des IP-Headers).
- ▶ In der Praxis sollte die MSS so gewählt werden, dass keine IP-Fragmentierung beim Senden notwendig ist¹

Beispiele:

- ▶ MSS bei FastEthernet
 - ▶ MTU beträgt 1500 B.
 - ▶ Davon entfallen 20 B auf den IPv4-Header und weitere 20 B auf den TCP-Header (sofern keine Optionen verwendet werden).
 - ▶ Die sinnvolle MSS beträgt demnach 1460 B.
- ▶ DSL-Verbindungen
 - ▶ Zwischen Ethernet- und IP-Header wird ein 8 B langer PPPoE-Header eingefügt.
 - ▶ Demzufolge sollte die MSS auf 1452 B reduziert werden.
- ▶ VPN-Verbindungen
 - ▶ Abhängig vom eingesetzten Verschlüsselungsverfahren sind weitere Header notwendig.
 - ▶ Die sinnvolle MSS ist hier nicht immer offensichtlich.

¹ Das ist in der Praxis natürlich nicht immer möglich, da auf Schicht 4 im Allgemeinen unbekannt ist, welches Protokoll auf Schicht 3 verwendet wird, ob Optionen/Extension Header verwendet werden oder es auf noch eine zusätzliche Encapsulation zwischen Schicht 3 und Schicht 2 gibt (z.B. PPPoE bei DSL-Verbindungen).

Fluss- und Staukontrolle bei TCP

TCP-Flusskontrolle

Ziel der **Flusskontrolle** ist es, Überlastsituationen beim Empfänger zu vermeiden. Dies wird erreicht, indem der Empfänger eine Maximalgröße für das Sendefenster des Senders vorgibt.

- ▶ Empfänger teilt dem Sender über das Feld **Receive Window** im TCP-Header die aktuelle Größe des Empfangsfensters W_r mit.
- ▶ Der Sender interpretiert diesen Wert als die maximale Anzahl an Byte, die ohne Abwarten einer Bestätigung übertragen werden dürfen.
- ▶ Durch Herabsetzen des Wertes kann die Übertragungsrate des Senders gedrosselt werden, z. B. wenn sich der Empfangspuffer des Empfängers füllt.

TCP-Staukontrolle

Ziel der **Staukontrolle** ist es, Überlastsituationen im Netz zu vermeiden. Dazu muss der Sender Engpässe im Netz erkennen und die Größe des Sendefensters entsprechend anpassen.

Zu diesem Zweck wird beim Sender zusätzlich ein **Staukontrollfenster** (engl. **Congestion Window**) W_c eingeführt, dessen Größe wir mit w_c bezeichnen:

- ▶ W_c wird vergrößert, solange Daten verlustfrei übertragen werden.
- ▶ W_c wird verkleinert, wenn Verluste auftreten.
- ▶ Für das tatsächliche Sendefenster gilt stets $w_s = \min\{w_c, w_r\}$.

TCP-Staukontrolle

Man unterscheidet bei TCP grundsätzlich zwischen zwei Phasen der Staukontrolle:

1. Slow-Start:

- ▶ Für jedes bestätigte Segment wird W_c um eine MSS vergrößert.
- ▶ Dies führt zu **exponentiellem Wachstum** des Staukontrollfensters bis ein Schwellwert (engl. **Congestion Threshold**) erreicht ist.
- ▶ Danach wird mit der Congestion-Avoidance-Phase fortgefahrene.

2. Congestion Avoidance:

- ▶ Für jedes bestätigte Segment wird W_c lediglich um $(1/w_c)$ MSS vergrößert, d. h. nach Bestätigung eines vollständigen Staukontrollfensters um genau eine MSS.
- ▶ Ein vollständiges Fenster kann frühestens nach 1 RTT sein.
- ▶ Dies führt zu **linearem Wachstum** des Staukontrollfensters in der RTT.

TCP-Varianten:

- ▶ Wir betrachten hier eine auf das Wesentliche reduzierte Implementierung von TCP, die auf **TCP Reno** basiert.
- ▶ Die einzelnen TCP-Version (**Tahoe**, **Reno**, **New Reno**, **Cubic**, ...) unterscheiden sich in Details, sind aber alle zueinander kompatibel.
- ▶ Linux verwendet derzeit **TCP Cubic**, welches das Congestion Window schneller anwachsen lässt als andere TCP-Varianten.

Die folgende Beschreibung bezieht sich auf eine vereinfachte Implementierung von **TCP Reno**):

1. 3 duplizierte Bestätigungen (Duplicate ACKs)

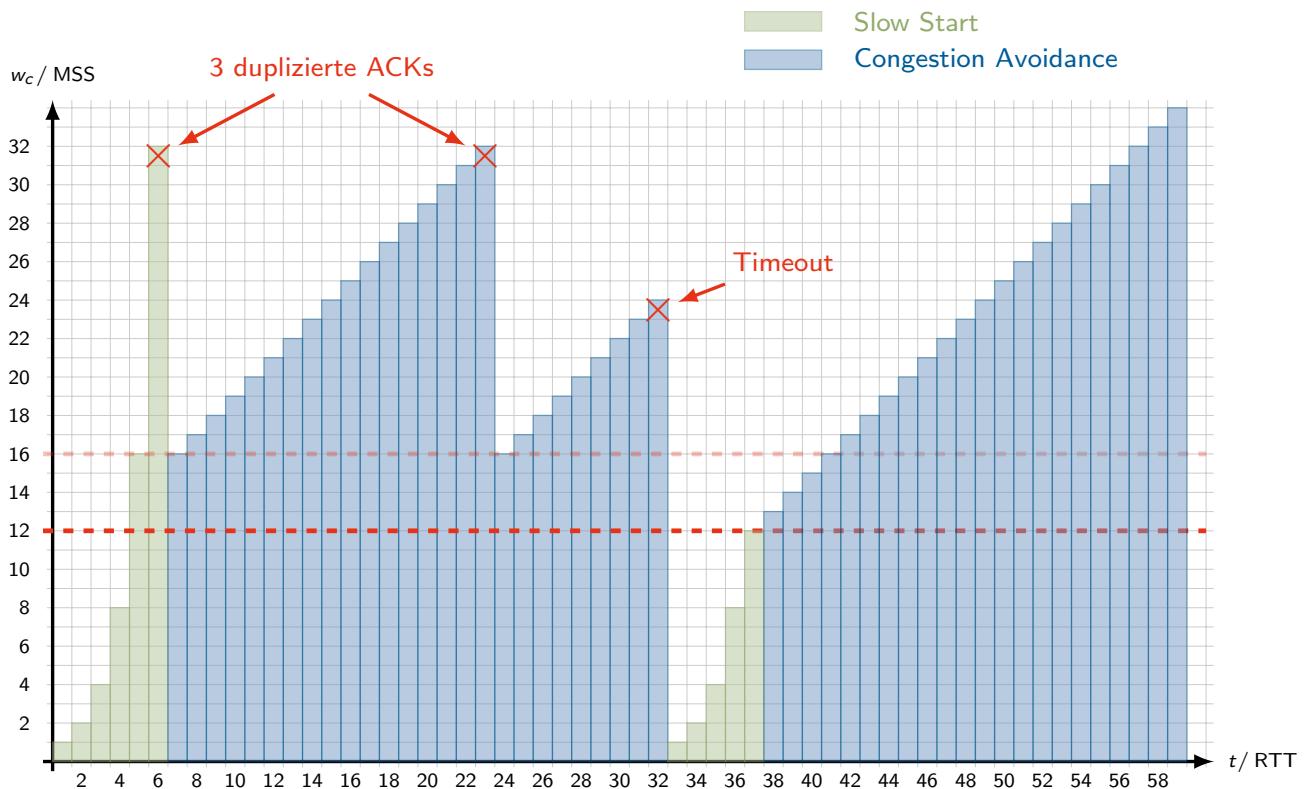
- ▶ Setze den Schwellwert für die Stauvermeidung auf $w_c/2$.
- ▶ Reduziere W_c auf die Größe dieses Schwellwerts.
- ▶ Beginne mit der Stauvermeidungsphase.

2. Timeout

- ▶ Setze den Schwellwert für die Stauvermeidung auf $w_c/2$.
- ▶ Setze $w_c = 1$ MSS.
- ▶ Beginne mit einem neuen Slow-Start.

- ▶ Der Vorgänger **TCP-Tahoe** unterscheidet z. B. nicht zwischen diesen beiden Fällen und führt immer Fall 2 aus.
- ▶ Grundsätzlich sind alle TCP-Versionen kompatibel zueinander, allerdings können sich die unterschiedlichen Staukontrollverfahren gegenseitig nachteilig beeinflussen.

Beispiel: TCP-Reno (mit einigen Vereinfachungen)



Anmerkungen

Obwohl TCP gesicherte Verbindungen ermöglicht, dient es [nicht der Kompensation eines unzuverlässigen Physical oder Data Link Layers](#):

- ▶ TCP interpretiert von Verlust von Paketen (Daten und Bestätigungen) stets als eine Folge einer **Überlastsituation**.
- ▶ In der Folge reduziert TCP die Datenrate.
- ▶ Handelt es sich bei den Paketverlusten jedoch um die Folge von Bitfehlern, so wird die Datenrate unnötiger Weise gedrosselt.
- ▶ Durch die ständige Halbierung der Datenrate oder neue Slow-Starts kann das Sendefenster nicht mehr auf sinnvolle Größen anwachsen.
- ▶ In der Praxis ist TCP bereits mit 1 % Paketverlust, der nicht auf Überlast zurückzuführen ist, überfordert.

⇒ Die Schichten 1 – 3 müssen eine für TCP „ausreichend geringe“ Paketfehlerrate bereitstellen.

- ▶ In der Praxis bedeutet dies, dass Verlustwahrscheinlichkeiten in der Größenordnung von 10^{-3} und niedriger notwendig bzw. anzustreben sind.
- ▶ Bei Bedarf müssen zusätzliche Bestätigungsverfahren auf Schicht 2 zum Einsatz kommen, um dies zu gewährleisten (z. B. IEEE 802.11).

Übersicht

Motivation

Multiplexing

Verbindungslose Übertragung

Verbindungsorientierte Übertragung

Sliding-Window-Verfahren

Transmission Control Protocol (TCP)

Fluss- und Staukontrolle bei TCP

Network Address Translation (NAT)

Network Address Translation (NAT)

In Kapitel 3 haben wir gelernt, dass

- ▶ IP-Adressen zur End-zu-End-Adressierung verwendet werden,
- ▶ aus diesem Grund global eindeutig sind und
- ▶ speziell die heute hauptsächlich verwendeten IPv4-Adressen sehr knapp sind.

Frage: Müssen IP-Adressen immer **eindeutig** sein?

Antwort: Nein, IP-Adressen müssen nicht eindeutig sein, wenn

- ▶ keine Kommunikation mit im Internet befindlichen Hosts möglich sein muss **oder**
- ▶ die nicht eindeutigen **privaten IP-Adressen** auf geeignete Weise in **öffentliche Adressen** übersetzt werden.

Definition: NAT

Als **Network Address Translation (NAT)** bezeichnet man allgemein Techniken, welche es ermöglichen, N private (nicht global eindeutige) IP-Adressen auf M globale (weltweit eindeutige) IP-Adressen abzubilden.

- ▶ $N \leq M$: Die Übersetzung geschieht statisch oder dynamisch indem jeder privaten IP-Adresse mind. eine öffentliche IP-Adresse zugeordnet wird.
- ▶ $N > M$: In diesem Fall wird eine öffentliche IP-Adresse von mehreren Computer gleichzeitig genutzt. Eine eindeutige Unterscheidung kann mittels **Port-Multiplexing** erreicht werden. Der häufigste Fall ist $M = 1$, z. B. ein privater DSL-Anschluss.

Was sind private IP-Adressen?

Private IP-Adressen sind spezielle Adressbereiche, welche

- ▶ zur privaten Nutzung ohne vorherige Registrierung freigaben sind,
- ▶ deswegen in unterschiedlichen Netzen vorkommen können,
- ▶ aus diesem Grund nicht eindeutig und zur End-Zu-End-Addressierung zwischen öffentlich erreichbaren Netzwerken geeignet sind und
- ▶ daher IP-Pakete mit privaten Empfänger-Adressen von Routern im Internet nicht weitergeleitet werden (oder werden sollten).

Die privaten Adressbereiche sind:

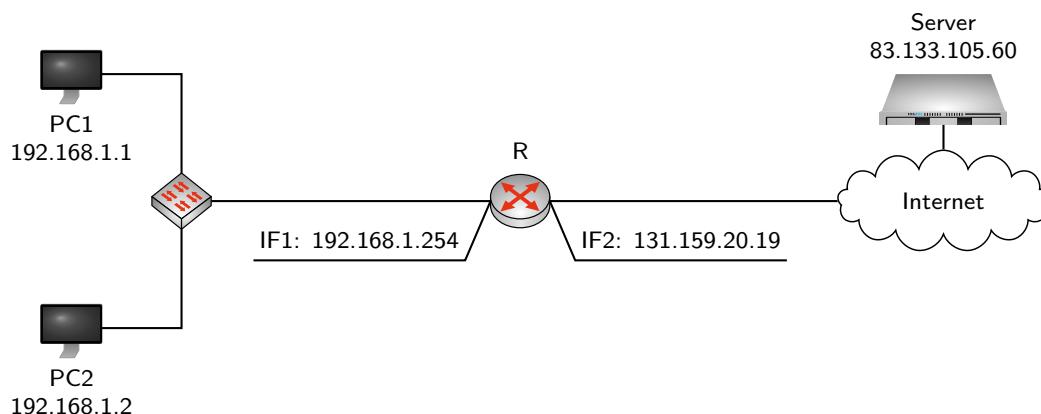
- ▶ 10.0.0.0/8
- ▶ 172.16.0.0/18
- ▶ 169.254.0.0/16
- ▶ 192.168.0.0/16

Der Bereich 169.254.0.0/16 wird zur automatischen Adressvergabe (Automatic Private IP Addressing) genutzt:

- ▶ Startet ein Computer ohne statisch vergebene Adresse, versucht dieser, einen DHCP-Server zu erreichen.
- ▶ Kann kein DHCP-Server gefunden werden, vergibt das Betriebssystem eine zufällig gewählte Adresse aus diesem Adressblock.
- ▶ Schlägt anschließend die ARP-Auflösung zu dieser Adresse fehl, wird angenommen, dass diese Adresse im lokalen Subnetz noch nicht verwendet wird. Andernfalls wird eine andere Adresse gewählt und der Vorgang wiederholt.

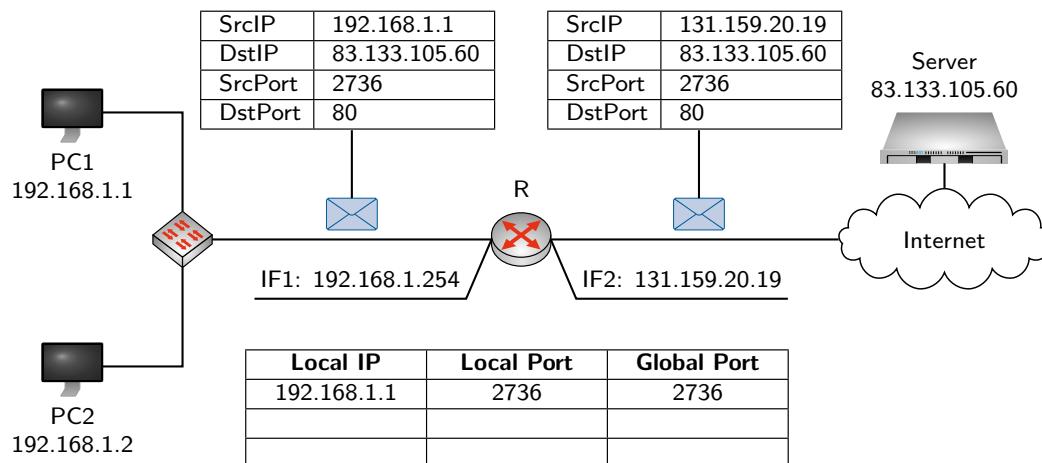
Wie funktioniert NAT im Detail?

Im Allgemeinen übernehmen Router die Netzwerkadressübersetzung:



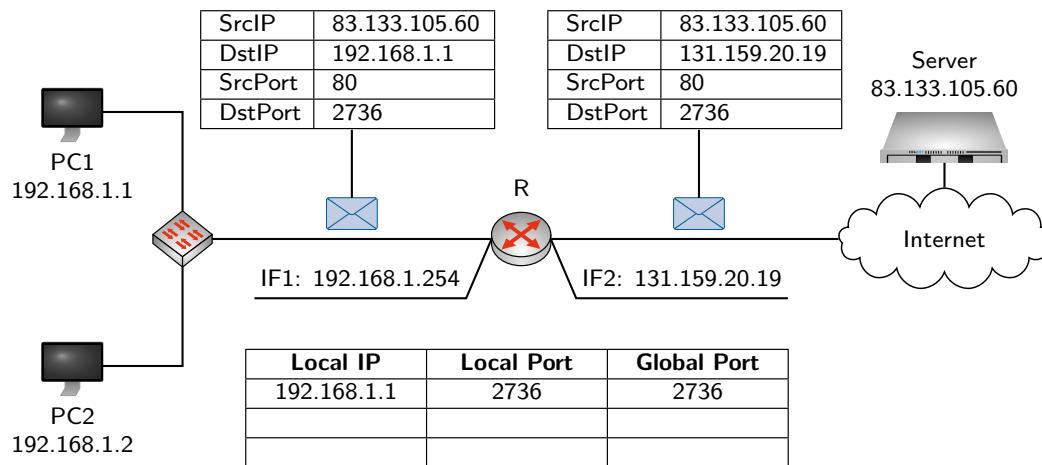
- ▶ PC1, PC2 und R können mittels privater IP-Adressen im Subnetz 192.168.1.0/24 miteinander kommunizieren.
- ▶ R ist über seine öffentliche Adresse 131.159.20.19 global erreichbar.
- ▶ PC1 und PC2 können wegen ihrer privaten Adressen nicht direkt mit anderen Hosts im Internet kommunizieren.
- ▶ Hosts im Internet können ebensowenig PC1 oder PC2 erreichen – selbst dann, wenn sie wissen, dass sich PC1 und PC2 hinter R befinden und die globale Adresse von R bekannt ist.

PC1 greift auf eine Webseite zu, welche auf dem Server mit der Adresse 83.133.105.60 liegt:



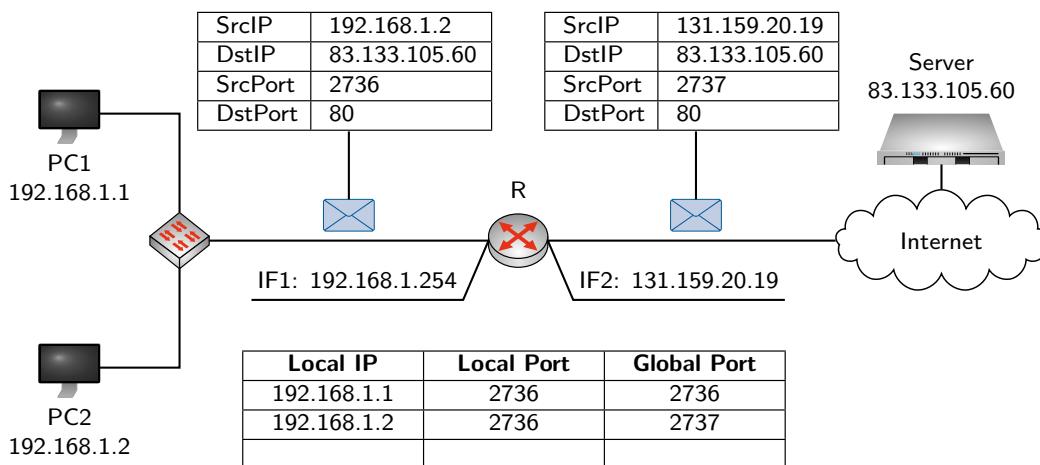
- ▶ Die NAT-Tabelle von R sei zu Beginn leer.
- ▶ PC1 sendet ein Paket (TCP SYN) an den Server:
 - ▶ PC1 verwendet seine private IP-Adresse als Absenderadresse
 - ▶ Der Quellport wird von PC1 zufällig im Bereich [1024,65535] gewählt (sog. **Ephemeral Ports**)
 - ▶ Der Zielport ist durch das Application Layer Protocol vorgegeben (80 = HTTP)
- ▶ Adressübersetzung an R:
 - ▶ R tauscht die Absenderadresse durch seine eigene globale Adresse aus
 - ▶ Sofern der Quellport nicht zu einer Kollision in der NAT-Tabelle führen würde, wird dieser beibehalten (andernfalls wird dieser ebenfalls ausgetauscht)
 - ▶ R erzeugt einen neuen Eintrag in seiner NAT-Tabelle, welche die Änderungen an dem Paket dokumentieren

Antwort vom Server an PC1



- ▶ Der Server generiert eine Antwort:
 - ▶ Der Server weiß nichts von der Adressübersetzung und hält R für PC1
 - ▶ Die Empfängeradresse ist daher die öffentliche IP von R, der Zielport der von R übersetzte Quellport aus der vorherigen Nachricht
- ▶ R macht die Adressübersetzung rückgängig
 - ▶ In der NAT-Tabelle wird nach der Zielportnummer in der Spalte Global Port gesucht, dieser in Local Port zurückübersetzt und die Ziel-IP des Pakets gegen die private IP-Adresse von PC1 ausgetauscht
 - ▶ Das so modifizierte Paket wird an PC1 weitergeleitet
 - ▶ Wie der Server weiß auch PC1 nichts von der Adressübersetzung

PC2 greift nun ebenfalls auf den Server zu:



- ▶ PC2 sendet ebenfalls ein Paket (TCP SYN) an den Server:
 - ▶ Rein zufällig wählt PC2 denselben Quell-Port wie PC1 (Portnummer 2736)
- ▶ Adressübersetzung an R:
 - ▶ R bemerkt, dass es bereits einen zu PC1 gehörenden Eintrag für den lokalen Port 2736 gibt
 - ▶ R erzeugt einen neuen Eintrag in der NAT-Tabelle, wobei für den globalen Port ein zufälliger Wert gewählt wird (z. B. der ursprüngliche Port von PC2 + 1)
 - ▶ Das Paket von PC2 wird entsprechend modifiziert und an den Server weitergeleitet
- ▶ Aus Sicht des Servers hat der „Computer“ R einfach zwei TCP-Verbindungen aufgebaut.

Ein Router könnte in die NAT-Tabelle zusätzliche Informationen aufnehmen:

- ▶ Ziel-IP und Ziel-Port
- ▶ Das verwendete Protokoll (TCP, UDP)
- ▶ Die eigene globale IP (sinnvoll, wenn ein Router mehr als eine globale IP besitzt)

In Abhängigkeit der gespeicherten Informationen unterscheidet man unterschiedliche Typen von NAT. Die eben diskutierte Variante (zzgl. eines Vermerks des Protokolls in der NAT-Tabelle) bezeichnet man als **Full Cone NAT**.

Eigenschaften von Full Cone NAT:

- ▶ Bei eingehenden Verbindungen findet keine Prüfung der Absender-IP oder des Absender-Ports statt, da die NAT-Tabelle nur den Ziel-Port und die zugehörige IP-Adresse bzw. Portnummer im lokalen Netz enthält.
- ▶ Existiert also einmal ein Eintrag in der NAT-Tabelle, so ist ein interner Host aus dem Internet über diesen Eintrag auch für jeden erreichbar, der ein TCP- bzw. UDP-Paket an die richtige Portnummer sendet.

Andere NAT-Varianten:

- ▶ Port Restricted NAT
- ▶ Address Restricted NAT
- ▶ Port and Address Restricted NAT
- ▶ Symmetric NAT

Allgemeine Anmerkungen

- ▶ Ist NAT eine Firewall?
 - ▶ Nein.
 - ▶ Restriktive NAT-Varianten bieten zwar insofern einen grundlegenden Schutz, da sie eingehende Verbindungen ohne vorherigen Verbindungsaufbau aus dem lokalen Netz heraus erlauben, dies sollte aber nicht mit den Funktionen einer Firewall verwechselt werden.
 - ▶ Eine darüber hinausgehende Filterung von Verbindungen (wie es bei einer Firewall der Fall wäre) findet nicht statt.
- ▶ Wie viele Einträge kann eine NAT-Tabelle fassen?
 - ▶ Im einfachsten Fall (Full Cone NAT) beträgt die theoretische Maximalgrenze ca. 2^{16} pro Transportprotokoll (TCP und UDP) und pro globaler IP-Adresse.
 - ▶ Bei komplexeren NAT-Typen sind durch die Aufnahme der Ziel-Ports mehr Kombinationen möglich.
 - ▶ In der Praxis ist die Größe durch die Fähigkeiten des Routers beschränkt (einige 1000 Mappings).
- ▶ Werden Mappings aus der NAT-Tabelle wieder gelöscht?
 - ▶ Dynamisch erzeugte Mappings werden nach einer gewissen Inaktivitätszeit gelöscht.
 - ▶ U.U. entfernt ein NAT-fähiger Router auch Mappings sofort, wenn er einen TCP-Verbindungsabbau erkennt (implementierungsabhängig).
- ▶ Können Einträge in der NAT-Tabelle auch von Hand erzeugt werden
 - ▶ Ja, diesen Vorgang nennt man [Port Forwarding](#).
 - ▶ Auf diese Weise wird es möglich, hinter einem NAT einen auf einem bestimmten Port öffentlich erreichbaren Server zu betreiben.

NAT und ICMP

- ▶ NAT verwendet Portnummern des Transportprotokolls
- ▶ Was ist, wenn das Transportprotokoll keine Portnummern hat oder IP-Pakete ohne TCP-/UDP-Header verschickt werden, z. B. ICMP?

Antwort: Die ICMP-ID kann anstelle der Portnummern genutzt werden.

Problem: Traceroute funktioniert mit manchen Virtualisierungslösungen nicht, z. B. wenn ältere Versionen der Virtualbox-NAT-Implementierung verwendet werden.

- ▶ Traceroute basiert auf ICMP-TTL-Exceeded-Nachrichten
- ▶ Diese Nachrichten haben (anders als ein ECMP Echo Reply) keine ICMP-ID.
- ▶ Das liegt daran, dass jedes beliebige IP-Paket (nicht zwangsläufig ein ICMP-Echo-Request) ein ICMP-TTL-Exceeded auslösen kann und dieses (wie im Fall eines TCP-Pakets) natürlich keine ICMP-ID besitzt.
- ▶ Stattdessen trägt der Time-Exceeded den vollständigen IP-Header und die ersten 8 Byte der Payload des Pakets, welches den Time-Exceeded ausgelöst hat.
- ▶ Eine NAT-Implementierung müsste nun im Fall eines TTL-Exceeded in diesen ersten 8 Byte nach der ICMP-ID eines Echo-Requests oder aber nach den Portnummern eines Transportprotokolls suchen, um die Übersetzungen rückgängig machen zu können.
- ▶ Genau diese Rückübersetzung führen ältere Versionen der NAT-Implementierung von Virtualbox nicht durch.

NAT und IPv6

- ▶ NAT kann auch für IPv6 verwendet werden.

Präfix-Übersetzung

- ▶ Aufgrund der Probleme und Herausforderungen durch NAT bei IPv4 ist NAT für IPv6 sogar in RFC 6296 spezifiziert.
- ▶ Dabei wird eine One-to-One Mapping von Adressen erzeugt.
 - ▶ Dies wäre auch bei IPv4 möglich.
- ▶ Damit können **Unique-Local Unicast-Adressen** ($fc00::/7$, also **private** IPv6-Adressen) in global gültige Adressen übersetzt werden.
- ▶ Die Übersetzung erfolgt auf Präfixen:
 - ▶ Ein interner Präfix $fd01:0203:0405::/48$ wird z. B. auf das globale Präfix $2001:db8:0001::/48$ abgebildet.
- ▶ Dabei werden keine Layer 4 Merkmale (Ports, Identifier) verwendet.
- ▶ Die Übersetzung erfolgt, abgesehen von der Konfiguration der Adresspräfixe, zustandslos. Es wird keine NAT-Tabelle benötigt.
- ▶ Um zu verhindern, dass die Internet-Checksums in höheren Schichten modifiziert werden müssen, kann die Adressübersetzung so gewählt werden, dass die ursprüngliche Checksum weiterhin stimmt.

Einsatz von NAT bei IPv6

- ▶ Ein häufiger Grund für NAT (die Adressknappheit bei IPv4) ist bei IPv6 aber (noch) nicht gegeben.

Bibliography I

- [1] Analyzing UDP Usage in Internet Traffic, 2009. <http://www.caida.org/research/traffic-analysis/tcpudpratio/>.

Grundlagen Rechnernetze und Verteilte Systeme

Kapitel 5 – Sitzungs-, Darstellungs- und Anwendungsschicht

Worum geht es in diesem Kapitel?

Einordnung im ISO/OSI-Modell

Sitzungsschicht

Dienste der Sitzungsschicht
Realisierung der Funktionalität der Sitzungsschicht

Darstellungsschicht

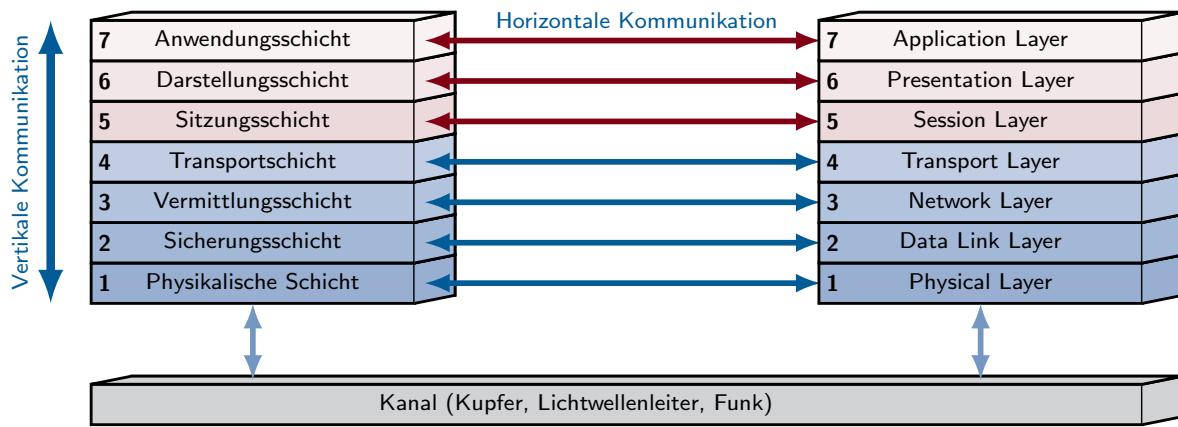
Aufgaben der Darstellungsschicht
Zeichensätze und Kodierung
Kodierung
Strukturierte Darstellung
Datenkompression

Anwendungsschicht

Domain Name System (DNS)
Uniform Resource Locator (URL)
HyperText Transfer Protocol (HTTP)
Simple Mail Transfer Protocol (SMTP)
File Transfer Protocol (FTP)

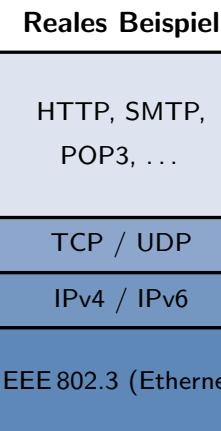
Zusammenfassung

Einordnung im ISO/OSI-Modell



Modell und Realität

- ▶ Dienste der Sitzungs- und der Darstellungsschicht sind in einzelnen Fällen in Form standardisierter Protokolle implementiert.
- ▶ In anderen Fällen sind Funktionen, die der Sitzungs- bzw. der Darstellungsschicht zuzuordnen sind, in die Anwendung integriert.



Modell und Realität

- ▶ Die Standards der ITU-Serie X.200 beschreiben Dienste der sieben OSI-Schichten sowie Protokolle zur Erbringung dieser Dienste.
- ▶ Die in diesen Standards vorgenommene Strukturierung ist nützlich.
- ▶ Etliche OSI-Protokolle haben in der Praxis kaum Bedeutung.
- ▶ Oft ist keine strikte Trennung zwischen Sitzung-, Darstellungs- und Anwendungsschicht möglich.
- ▶ Im [Internet Model](#) (RFC 1122) werden alle diese Funktionen der Anwendungsschicht zugeordnet.

Im Folgenden werden wir

- ▶ die Aufgaben der Sitzungs- und Darstellungsschicht erläutern,
- ▶ beispielhaft einige Protokolle kennenlernen, deren Funktionen den Schichten 5 und 6 zugeordnet werden können,
- ▶ sowie wichtige Protokolle der Anwendungsschicht erläutern.

Übersicht

Einordnung im ISO/OSI-Modell

Sitzungsschicht

Dienste der Sitzungsschicht
Realisierung der Funktionalität der Sitzungsschicht

Darstellungsschicht

Aufgaben der Darstellungsschicht
Zeichensätze und Kodierung
Kodierung
Strukturierte Darstellung
Datenkompression

Anwendungsschicht

Domain Name System (DNS)
Uniform Resource Locator (URL)
HyperText Transfer Protocol (HTTP)
Simple Mail Transfer Protocol (SMTP)
File Transfer Protocol (FTP)

Zusammenfassung

Sitzungsschicht

Die Sitzungsschicht bietet nach X.200 in zwei grundlegend verschiedene Betriebsarten:

1. verbindungsorientiert (engl. „connection-oriented“)

- ▶ Es wird eine Verbindung zwischen den Kommunikationspartnern aufgebaut.
- ▶ Die Verbindung bleibt dabei über die Dauer einzelner Transfers (oder Verbindungen) der Transportschicht hinweg bestehen.
- ▶ Wie bei TCP-Verbindungen kann auch hier zwischen den Phasen **Verbindungsauftbau**, **Datentransfer** und **Verbindungsabbau** unterschieden werden.

2. verbindungslos (engl. „connection-less“)

- ▶ Daten werden im Wesentlichen nur an die Transportschicht durchgereicht.
- ▶ Es wird keine Verbindung aufgebaut und kein Zustand zwischen den Kommunikationspartnern gehalten.

Hinweis

Eine Verbindung der Sitzungsschicht ist nicht gleichbedeutend mit einer Verbindung der Transportschicht. Eine **Session** kann beispielsweise nacheinander mehrere TCP-Verbindungen beinhalten.

Dienste der Sitzungsschicht

Definition (Session)

Eine **Session** beschreibt die Kommunikation zwischen mindestens zwei Teilnehmern mit definiertem Anfang und Ende sowie sich daraus ergebender Dauer.

Um für die dienstnehmende Schicht (Darstellungsschicht) eine Dialogführung zu ermöglichen, müssen gegebenenfalls mehrere Transportschicht-Verbindungen verwendet und kontrolliert werden. Dies kann auch die Behandlung abgebrochener und wiederaufgenommener TCP-Verbindungen beinhalten.

Im verbindungsorientierten Modus werden verschiedene Dienste angeboten:

- ▶ **Aufbau** und **Abbau** von Sessions,
- ▶ normaler und beschleunigter **Datentransfer**¹,
- ▶ Token-Management zur **Koordination** der Teilnehmer,
- ▶ **Synchronisation** und Resynchronisation,
- ▶ **Fehlermeldungen** und Aktivitätsmanagement, sowie
- ▶ **Erhaltung** und **Wiederaufnahme** von Sessions nach Verbindungsabbrüchen.

¹ Expedited Data Transfer: Dringliche Daten, z. B. Alarme oder Interrupts

Realisierung der Funktionalität der Sitzungsschicht

Beispiel 1: HTTP (Hyper Text Transfer Protocol) → Details später

- ▶ HTTP ist zunächst zustandslos.
- ▶ Zwischen mehreren Anfragen und Antworten besteht zunächst kein Zusammenhang.
- ▶ **Cookies** ermöglichen es, dass eine Sitzung über mehrere Anfragen und Antworten, Interaktionen und TCP-Verbindungen hinweg bestehen bleibt.
- ▶ Cookies sind kleine Datenfragmente, welche von einer Website (bzw. Anwendung) auf den Client übertragen und dort gespeichert werden können.
- ▶ Dadurch wird es möglich, zeitlich getrennte oder von unterschiedlichen Adressen stammende Anfragen einem bestimmten Client und sogar einem bestimmten Nutzer zuordnen zu können.
- ▶ HTTP wird überlicherweise der Anwendungsschicht (Schicht 7) zugeordnet, beinhaltet aber auch Funktionen der Darstellungs- und Sitzungsschicht (Schichten 6/5).
- ▶ Eine strikte Zuordnung ist nur schwer möglich.

Beispiel 2: TLS (Transport Layer Security)¹

- ▶ TLS ist ein Protokoll zur verschlüsselten Übertragung von Daten.
- ▶ Es ist die Grundlage beispielsweise für HTTPS.
- ▶ Es bietet unter anderem
 - ▶ Authentifizierung,
 - ▶ Integritätsschutz und
 - ▶ Vertraulichkeit (Verschlüsselung).
- ▶ Beim Verbindungsaufbau werden zunächst die Kommunikationsparameter der Sitzung (z. B. Algorithmen, Schlüssel, Authentifizierung) ausgehandelt.
- ▶ Sitzungen können mit Hilfe von Session-IDs oder Session-Tickets über mehrere TCP-Verbindungen hinweg erhalten bleiben.
- ▶ Während die Funktionen zur Verwaltung und Wiederaufnahme von Sessions der Sitzungsschicht zuzuordnen sind, fallen insbesondere die Verschlüsselungsfunktionen in den Bereich der Darstellungsschicht.

¹ Dient hier lediglich als Beispiel, wird im Rahmen der Vorlesung aber nicht eingehender behandelt. Man sollte allerdings wissen, wozu es dient.

Übersicht

Einordnung im ISO/OSI-Modell

Sitzungsschicht

Dienste der Sitzungsschicht
Realisierung der Funktionalität der Sitzungsschicht

Darstellungsschicht

Aufgaben der Darstellungsschicht
Zeichensätze und Kodierung
Kodierung
Strukturierte Darstellung
Datenkompression

Anwendungsschicht

Domain Name System (DNS)
Uniform Resource Locator (URL)
HyperText Transfer Protocol (HTTP)
Simple Mail Transfer Protocol (SMTP)
File Transfer Protocol (FTP)

Zusammenfassung

Darstellungsschicht

Die Aufgabe der **Darstellungsschicht** (engl. **Presentation Layer**) ist es, den Kommunikationspartnern eine einheitliche Interpretation der Daten zu ermöglichen, d. h. Daten in einem einheitlichen Format zu übertragen.

Der Darstellungsschicht sind grundsätzlich folgende Aufgaben zugeordnet:

- ▶ die Darstellung der Daten (Syntax),
- ▶ die Datenstrukturen zur Übertragung der Daten
- ▶ die Darstellung der Aktionen an diesen Datenstrukturen, sowie
- ▶ Datentransformationen.

Hinweis

Die Darstellung auf Schicht 6 muss nicht der Darstellung auf Schicht 7 (Anwendungsschicht) entsprechen. Die Darstellungsschicht ist für die **Syntax** der Nutzdaten verantwortlich, die **Semantik** verbleibt bei den Anwendungen.²

- ▶ Anwendungen sollen syntaxunabhängig miteinander kommunizieren können.
- ▶ Anwendungsspezifische Syntax kann von der Darstellungsschicht in eine einheitliche Form umgewandelt und dann übertragen werden.

² Unter **Syntax** versteht man die Darstellung von Daten nach bestimmten Regeln (**Grammatik**). Werden Daten durch Bedeutung ergänzt, spricht man von Information (Aufgabe der **Semantik**).

Aufgaben der Darstellungsschicht

Den grundlegenden Aufgaben der Darstellungsschicht lassen sich konkrete Funktionen zuordnen:

- ▶ **Kodierung** der Daten
 - ▶ Übersetzung zwischen Zeichensätzen und Codewörtern gemäß standardisierter Kodierungsvorschriften
 - ▶ Kompression von Daten vor dem Senden (Entfernung von unerwünschter Redundanz)
 - ▶ Verschlüsselung
- ▶ **Strukturierte Darstellung** von Daten
 - ▶ Plattformunabhängige, einheitliche Darstellung
 - ▶ Übersetzung zwischen verschiedenen Datenformaten
 - ▶ Serialisierung von Binärdaten (Übersetzung von binären Datenformaten in Textdarstellungen)

Wie auch bei der Sitzungsschicht gilt hier: Protokolle lassen sich meist nicht eindeutig der Darstellungsschicht zuordnen, da sie häufig auch Funktionen anderer Schichten erfüllen.

Beispiel: TLS

- ▶ Die Verschlüsselungsfunktionen von TLS können der Darstellungsschicht zugeordnet werden.
- ▶ Funktionen wie Verbindungsaufbau und -abbau sowie Authentifizierung und Autorisierung fallen hingegen in den Bereich der Sitzungsschicht.

Zeichensätze und Kodierung

Daten liegen in einer von zwei Formen vor:

1. **Textzeichen bzw. Symbole** in lesbbarer Form („human readable“), z. B. Buchstaben, Textrepräsentation von Zahlen, Sonderzeichen...
 - ▶ Ein **Zeichensatz** ist eine Menge textuell darstellbarer Zeichen sowie deren Zuordnung zu einem **Codepoint**¹.
 - ▶ Wie die Codepoints eines bestimmten Zeichensatzes in binärer Form (also mittels einer Sequenz von Bits) dargestellt werden wird durch **Kodierungsvorschriften** festgelegt.
 - ▶ Für einen Zeichensatz können ggf. mehrere mögliche Kodierungen existieren.
2. **Binäre Daten** (also eine Sequenz von Bits), z. B. binäre Darstellung von Buchstaben, Zahlen und Symbolen aber auch Bilder, Musik, Filme, etc. in digitaler Form...
 - ▶ Ein **Datum** ist eine für Computer verarbeitbare „Einheit“, d. h. eine kurze Sequenz von Bits, deren Länge meist ein Vielfaches von 8 bit ist.²
 - ▶ Was ein Datum repräsentiert – eine Zahl, ein Zeichen, einen Teil davon oder doch ein Bild – ist kontextabhängig (vgl. Kapitel 1 „Information und deren Bedeutung“).
 - ▶ Ist bekannt, dass es sich bei den vorliegenden Daten um Text handelt, welche Kodierung verwendet wurde und um welchen Zeichensatz es sich handelt, lassen sich die binären Daten leicht wieder in Textzeichen übersetzen.

Hinweis: Binäre Daten (z. B. ein Bild) können nicht ohne Weiteres mit einem Zeichensatz dargestellt werden. Hierzu gibt es eigene Kodierungsvorschriften, die es ermöglichen, Binärdaten zu rekodieren, so dass sie in einem Zeichensatz darstellbar werden.

¹ Ein Codepoint ist eine eindeutige „Kennzahl“ für höchstens ein Textzeichen (nicht alle Codepoints müssen vergeben sein).

² Die kleinste im Speicher adressierbare Einheit ist für heutige Computer ein Oktett, also ein Block von 8 bit, welches im Sprachgebrauch als Byte bezeichnet wird. Üblich sind daneben noch die Größen 16 bit, 32 bit und 64 bit, welche häufig als Word, Double Word bzw. Quad Word bezeichnet werden (aber genauso wenig standardisiert sind wie das Byte). Diese sind letztendlich durch die Registergröße der Prozessoren beschränkt. Da der Speicher aber byteweise adressiert wird, ist es prozessorabhängig, in welcher Reihenfolge die einzelnen Oktette geladen werden. Folglich ist die **Byte Order** von essentieller Bedeutung.

- ▶ Ein **Zeichensatz** verknüpft ein **Zeichen** mit einem **Codepoint**.
- ▶ Beispiele für Zeichensätze sind **ASCII**, **ISO-8859-15 (ISO-8859-1 mit €)** und **Unicode**.
- ▶ Ein Zeichensatz kann **druckbare Zeichen** sowie **Steuerzeichen** enthalten.
 - ▶ ASCII definiert 128 Zeichen.
 - ▶ Davon sind die Zeichen 0 – 31 und 127 sind **Steuerzeichen**.
 - ▶ Steuerzeichen sind z. B. Zeilenumbruch, Tabulator, und Protokollzeichen.
 - ▶ Ursprünglich wurden darüber Terminals und Drucker angesteuert.
- ▶ ISO-8859-15 definiert 256 Codepoints.
 - ▶ Zeichen 0 – 127 entsprechen dem ASCII Zeichensatz.
 - ▶ ISO-8859-15 war vor Unicode der im westeuropäischen Sprachraum ein verbreiteter Zeichensatz.

Unicode

- ▶ Unicode in Version 8.0 definiert 120 737 Zeichen.
- ▶ Die Größe des Coderaums (Anzahl möglicher Zeichen) beträgt bei Unicode 1 114 112 Zeichen.
- ▶ Unicode hat das Ziel, alle Schriftkulturen und Zeichensysteme abzubilden.
- ▶ Die Codepoints 0 – 255 entsprechen ISO-8859-1.
- ▶ Unicode definiert, wie Zeichen normalisiert, sortiert und verglichen werden sollen.
- ▶ Im Gegensatz zu eingeschränkten Zeichensätzen wie z. B. ASCII und ISO-8859-15 wird der Zeichensatz bei Unicode regelmäßig aktualisiert und erweitert.

Kodierung

Zur Übertragung müssen Zeichen (bzw. die entsprechenden Codepoints) **kodiert** werden. Man unterscheidet zwischen:

- ▶ **Fixed-Length Codes**, bei welchen alle Zeichen mit Codewörtern derselben Länge kodiert werden, z. B. ASCII (7 bit) oder UCS-2 (16 bit).
- ▶ **Variable-Length Codes**, bei denen Zeichen mit Codewörtern unterschiedlicher Länge kodiert werden, z. B. UTF-8 (1 – 4 B).¹

Die möglichen Kodierungsverfahren hängen dabei vom jeweiligen Zeichensatz ab:

- ▶ ASCII und ISO-8859-15 definieren die Kodierung mit dem Zeichensatz.
 - ▶ Codewörter sind bei ASCII 7 bit lang, wobei das highest-order Bit eines Oktetts stets 0 ist.
 - ▶ ISO-8859-15 verwendet 8 bit lange Codewörter.
- ▶ Unicode definiert keine Kodierung sondern nur einen Zeichensatz. Am häufigsten wird hier hierfür **UTF-8** verwendet, welches kompatibel zu ASCII² ist.

¹ Morse-Zeichen sind ein weiteres Beispiel: Hier werden häufiger auftretenden Zeichen kürzere Codewörter zugewiesen.

² Die Codewörter 0 – 127 entsprechen ASCII.

Unicode Transformation Format (UTF-8)

UTF-8 kodiert den Unicode Zeichensatz abhängig vom Codepoint mit 1 – 4 B langen Codewörtern:

Unicode-Bereich	Länge	binäre UTF-8 Kodierung	kodierbare Bits
U+0000 – U+007F	1 B	0xxxxxxx	7
U+0080 – U+07FF	2 B	110xxxxx 10xxxxxx	11
U+0800 – U+FFFF	3 B	1110xxxx 10xxxxxx 10xxxxxx	16
U+10000 – U+1FFFFFF	4 B	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	21

- ▶ Die Darstellung U+xxxx ist lediglich eine Notation der Codepoints für Unicode. Die hexadezimalen Ziffern geben dabei den Wert der **kodierten Bits** eines Codeworts an.
- ▶ Bei Codewörtern, die länger als 1 B sind, gibt die Anzahl der führenden 1-en vor der ersten 0 im ersten Oktett die Länge des Codeworts an.
- ▶ Die beiden highest-order Bits aller nachfolgenden Oktette eines Codeworts sind 10.
- ▶ Bei Codewörtern, die nur aus einem Oktett bestehen, ist das highest-order Bit stets 0 (vgl. ASCII).

Eigenschaften:

- ▶ UTF-8 ist rückwärts-kompatibel zu ASCII: ASCII kodierter Text ist valides UTF-8 und kann dementsprechend ohne Konvertierung als Unicode interpretiert werden.
- ▶ UTF-8 ist präfixfrei¹ und selbstsynch�nisiert.
- ▶ Nicht alle Kombinationen sind gültige Codewörter (Kompatibilitätsgründe mit UTF-16).

¹ Kein gültiges Codewort ist ein echtes Präfix eines anderen Codeworts.

Beispiel 1: Kodierung des Umlauts ä

- ▶ **Unicode:** Codepoint 228, Codewort in UTF-8: U+00E4 = 11000011 10100100 („LATIN SMALL LETTER A WITH DIAERESIS“)
- ▶ **ISO-8859-1 und ISO8859-15:** Codepoint 228, Codewort: 11100100 Obwohl beide Zeichensätze eine Teilmenge von Unicode sind, ergeben sich andere Codewörter
- ▶ **ASCII:** Keine Kodierung möglich, da Umlaute nicht Teil des Zeichensatzes ist.

Beispiel 2: Kodierung des Eurosymbols €

- ▶ **Unicode:** Codepoint 8384, Codewort in UTF-8: U+00E4 = 11100010 10000010 10101100
- ▶ **ISO-8859-15:** Codepoint 164, Codewort 10100100
- ▶ **ISO-8859-1 und ASCII:** Keine Kodierung möglich

Vom verwendeten Zeichensatz nicht unterstützte Zeichen können häufig mittels **Zeichen-Entität-Referenzen** kodiert werden:

- ▶ XML erlaubt beispielsweise die Kodierung beliebiger Unicode-Zeichen durch Angabe des Codepoints, z. B. Ġ
- ▶ HTML erlaubt die Kodierung häufiger verwendeter Zeichen mittels **named entities**, z. B. ä.
- ▶ LATEX erlaubt die Kodierung verschiedener Zeichen auf ähnliche Art, z. B. \ "a¹

Die Syntax ist aber abhängig vom jeweils verwendeten Protokoll (HTTP, SMTP → später) bzw. der Anwendung.

¹ Vertippt man sich und schreibt Verschlüsselung anstelle von Verschlüsselung, ergibt das dann in den Folien Verschlüsselung.

Strukturierte Darstellung

Damit Anwendungen Daten austauschen können, müssen diese eine einheitliche Syntax für die ausgetauschten Daten verwenden. Möglichkeiten hierfür sind:

- ▶ **(gepackte)¹ structs / serialisierte Speicherbereiche**
Daten werden so wie sie im Speicher vorliegen übertragen. Integration zwischen verschiedenen Systemen schwierig, weil diese die selben Datenstrukturen (und Compiler) verwenden müssen. Erweiterungen/Änderungen sind nur dann möglich, wenn alle beteiligten Systeme gleichzeitig aktualisiert werden.
- ▶ **Ad-hoc Datenformate**
Datenformat wird bei Bedarf „entworfen“. Problematisch sind hierbei die Dokumentation, Eindeutigkeit, Fehlerfreiheit (wie gut und sicher kann das Format geparsed werden) und Erweiterbarkeit.
- ▶ **Strukturierte Serialisierungsformate** wie JSON oder XML.

Beispiel: JavaScript Object Notation (JSON)

- ▶ Definiert in ECMA-404 [1] und RFC 7159 [3].
- ▶ Ursprünglich von JavaScript abgeleitet, mittlerweile aber sprachenunabhängiges Datenformat.
- ▶ Daten werden strukturiert und in lesbarer („human-readable“) Form als Text übertragen.

¹ In C bleibt es dem Compiler überlassen, wie viel Speicher ein struct tatsächlich belegt. Compiler-spezifische Keywords (bei gcc `__attribute__((packed))`) erzwingen, dass ein struct den minimal notwendigen Speicher belegt.

JSON

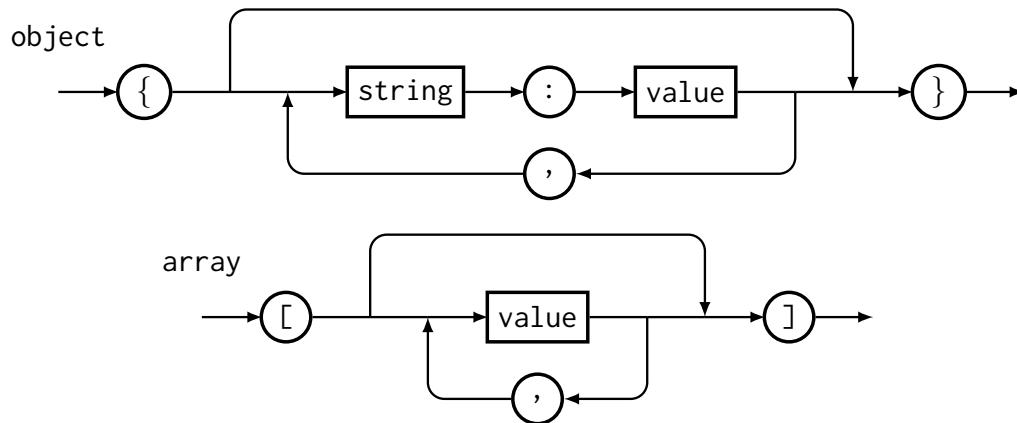
JSON definiert die folgenden Datentypen:

- ▶ number
 - ▶ string
 - ▶ boolean
 - ▶ array
 - ▶ object
 - ▶ null
-
- ▶ Zwischen den Elementen kann (beliebiger) Whitespace eingefügt werden.
 - ▶ JSON wird im allgemeinen als UTF-8 kodiert.
 - ▶ JSON Dokumente besitzen, anders als XML, kein explizites Schema.

Beispiel:

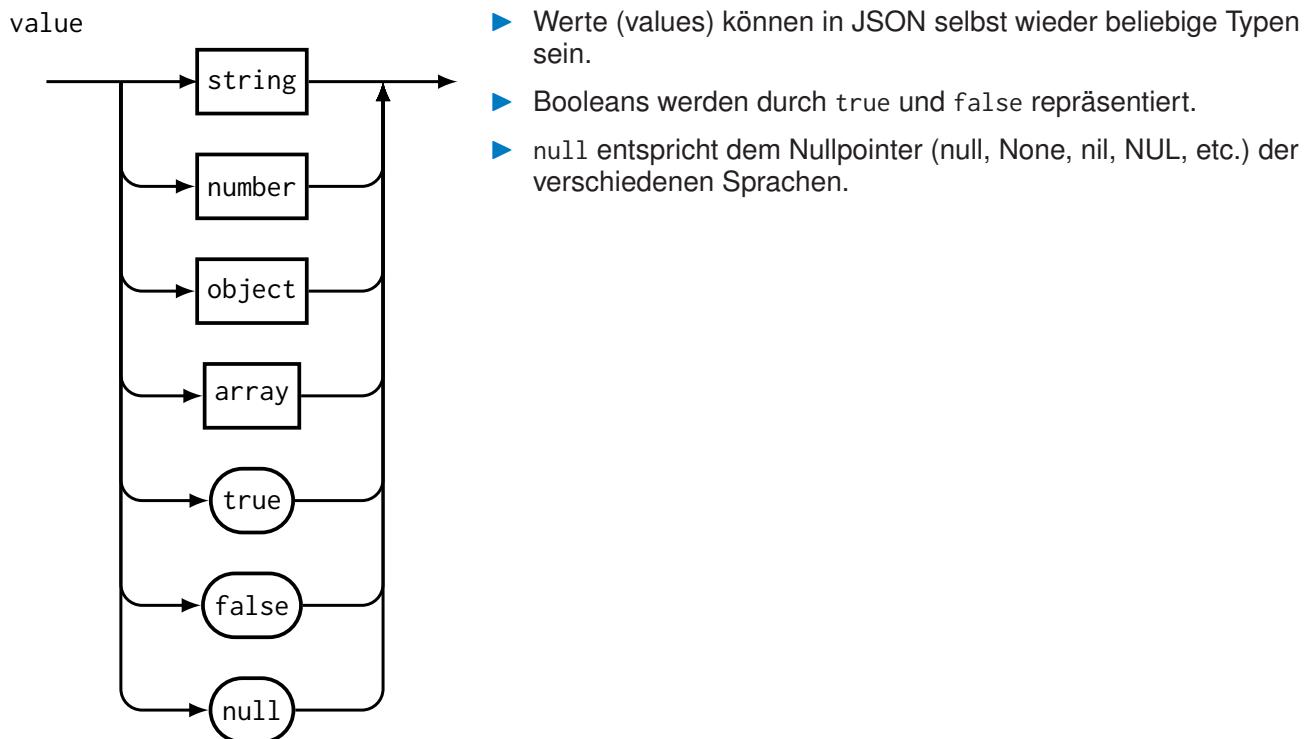
```
{  
    "Image": {  
        "Width": 800,  
        "Height": 600,  
        "Title": "View from 15th Floor",  
        "Thumbnail": {  
            "Url": "http://www.example.com/image/481989943",  
            "Height": 125,  
            "Width": 100  
        },  
        "Animated": false,  
        "IDs": [116, 943, 234, 38793]  
    }  
}
```

JSON Objects und Arrays

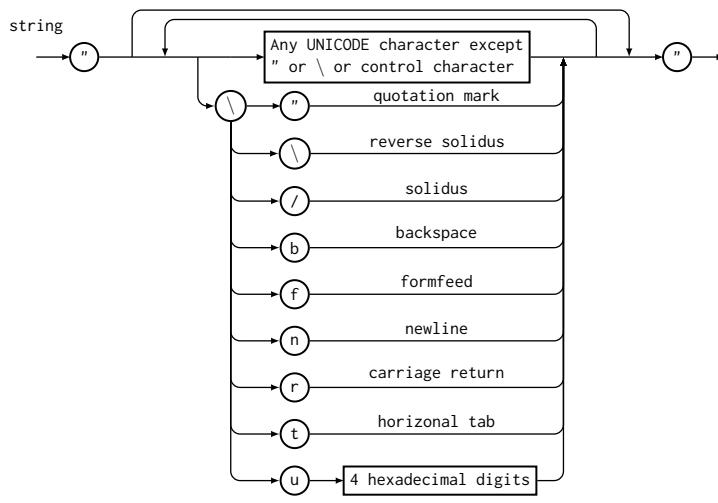


- ▶ Ein **JSON-Objekt** ist eine ungeordnete Sammlung von **Key/Value-Paaren**.
- ▶ Der Key (Schlüssel) ist ein Unicode-String.
- ▶ Die Values (Werte) können unterschiedliche Typen aufweisen.
- ▶ Eine Sammlung von JSON-Objekten ist konzeptuell vergleichbar mit Hashmaps bzw. Dictionaries verschiedener Programmiersprachen.
- ▶ Arrays sind (geordnete) Listen (leere Listen sind erlaubt).
- ▶ Bei der Übertragung bleibt die Reihenfolge der Elemente innerhalb einer Liste erhalten.
- ▶ Listen können wiederum Werte von unterschiedlichen Typen aufweisen.

JSON Values



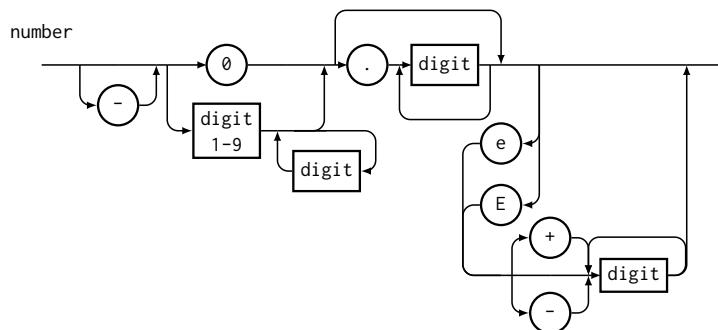
JSON Strings



- ▶ Strings können beliebige Unicode-Zeichenketten sein.
- ▶ Strings werden gequoted (also in der Form <text> notiert) und können leer sein.
- ▶ Steuerzeichen müssen escaped werden.
- ▶ Da Strings aus Unicode-Zeichen bestehen, können beliebige Binärdaten nicht direkt kodiert werden (es muss unterscheidbar sein, ob es sich um einen String oder um Binärdaten handelt).
- ▶ Das Problem kann durch Serialisierung der Binärdaten z. B. mittels Base64 [5] umgangen werden.¹

¹ Anything is human readable when base64 encoded and wrapped in XML ;)

JSON Numbers



- ▶ Numbers sind dezimal notierte Zahlen
- ▶ Es wird nicht zwischen Ganzzahlen und Gleitkommazahlen unterschieden.
- ▶ Hintergrund hierfür: JavaScript – worauf JSON ja ursprünglich basierte – kennt nur Gleitkommazahlen.
- ▶ Es ist implementierungsabhängig, ob Ganzzahlen und Gleitkommazahlen unterschieden werden und mit welcher Präzision diese abgebildet werden.
- ▶ Implementierungsabhängig ist auch, ob die Zahlenwerte 42, 4.2e1 und 42.0 als gleich angesehen werden.¹

¹ Ein Grund mehr, niemals Gleitkommazahlen auf Gleichheit zu testen.

Datenkompression

Bei Kompressionsverfahren muss unterscheiden werden:

1. Verlustfreie Komprimierung (engl. lossless compression)

- ▶ Komprimierte Daten können verlustfrei, d. h. exakt und ohne Informationsverlust, wiederhergestellt werden.
- ▶ Verlustfrei komprimierte Dateiformate sind beispielsweise ZIP, PNG² (Bilder), FLAC³ (Musik), ...

2. Verlustbehaftete Komprimierung (engl. lossy compression):

- ▶ Komprimierte Daten können im Allgemeinen nicht wieder exakt rekonstruiert werden.
- ▶ Es tritt also ein Verlust von Information bei der Komprimierung auf.
- ▶ Dafür ermöglichen diese Verfahren meist höhere und in Abhängigkeit des Verlustfaktors variable Kompressionsraten.
- ▶ Verlustbehaftet komprimierte Dateiformate sind beispielsweise MP3, MPEG, JPEG, ...

² Portable Network Graphics

³ Free Lossless Audio Codec

Beispiel 1: Huffman-Code

- ▶ Viele Protokolle komprimieren Daten vor dem Senden (Quellenkodierung).
- ▶ TLS beispielsweise bietet optional Kompressionsmethoden. Diese werden **vor** der Verschlüsselung angewandt. (Warum davor?)
- ▶ Ein häufig (u. a. von TLS) verwendetes Kompressionsverfahren für Texte ist der **Huffman-Code**.

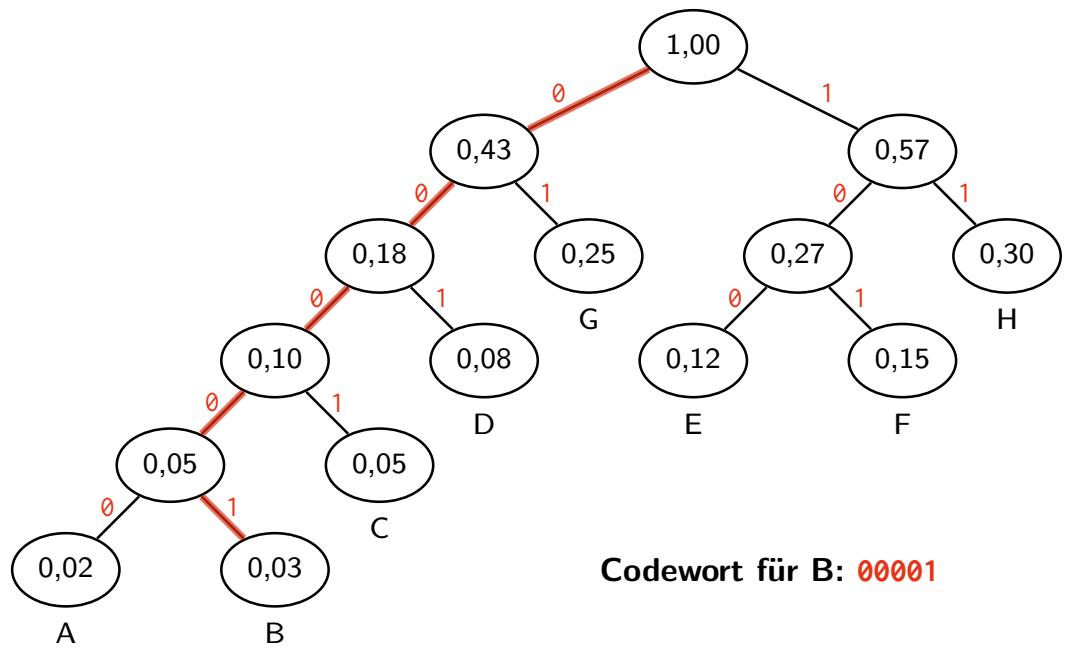
Grundlegende Idee der Huffman-Kodierung:

- ▶ Nicht alle Textzeichen treten mit derselben Häufigkeit auf, z. B. tritt der Buchstabe „E“ in der deutschen Sprache mit einer Häufigkeit von 17,4 % gefolgt von „N“ mit 9,8 % auf.
- ▶ Anstelle Zeichen mit uniformer Codewortlänge zu kodieren (z. B. ASCII-Code), werden **häufigen Zeichen kürzere Codewörter** zugewiesen.
- ▶ Die Abbildung zwischen Zeichen und Codewörtern bleibt dabei eindeutig und umkehrbar, weswegen es sich um ein verlustloses Kompressionsverfahren handelt.

Konstruktion eines Huffman-Codes

- Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftrittswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$.
- Es sei außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.¹

z	$\Pr[X = z]$
A	0,02
B	0,03
C	0,05
D	0,08
E	0,12
F	0,15
G	0,25
H	0,30



¹ Andernfalls würden die Aussagen zur Optimalität des Huffman-Codes im Allgemeinen nicht mehr zutreffen.

Durchschnittliche Codewortlänge

z	$\Pr[X = z]$	Huffman-Code	Länge $I_H(z)$	Uniformer Code
A	0,02	00000	5	000
B	0,03	00001	5	001
C	0,05	0001	4	010
D	0,08	001	3	011
E	0,12	010	3	100
F	0,15	011	3	101
G	0,25	10	2	110
H	0,30	11	2	111

- Uniformer Code:

$$E[I(z)] = 3,0 \text{ , da alle Codewörter gleich lang sind}$$

- Huffman-Code:

$$E[I_H(z)] = \sum_{z \in \mathcal{A}} \Pr[X = z] I_H(z) = 2,6$$

$$\Rightarrow \text{Die Einsparung beträgt } 1 - \frac{E[I_H(z)]}{E[I(z)]} \approx 13 \%$$

Anmerkungen zum Huffman-Code:

- ▶ Statische Huffman-Codes sind darauf angewiesen, dass die Auftrittswahrscheinlichkeit der Zeichen den Erwartungen entspricht.
- ▶ Zeichenhäufigkeiten können dynamisch bestimmt werden, allerdings muss dem Empfänger dann das verwendete **Codebuch** mitgeteilt werden.
- ▶ Längere Codewörter (z. B. ganze Wörter statt einzelner Zeichen) werden infolge der Komplexität zum Bestimmen des Codebuchs ein Problem.
- ▶ Der Huffman-Code ist ein **optimaler** und **präfixfreier** Code.

Definition (Optimaler Präfixcode)

Bei einem **präfixfreien Code** sind gültige Codewörter niemals Präfix eines anderen Codeworts desselben Codes. Ein **optimaler** präfixfreier Code minimiert darüber hinaus die mittlere Codewortlänge

$$\sum_{i \in \mathcal{A}} p(i) \cdot |c(i)|,$$

wobei $p(i)$ die Auftrittswahrscheinlichkeit von $i \in \mathcal{A}$ und $c(i)$ die Abbildung auf ein entsprechendes Codewort bezeichnen.

Beispiel 2: Familie der Run-length Codes

Grundlegende Idee

- ▶ Daten weisen häufig Wiederholungen einzelner Zeichen oder Gruppen von Zeichen auf.
- ▶ Anstelle bei Wiederholungen jedes Zeichen bzw. jede Zeichengruppe erneut zu kodieren, geschieht dies nur einmal.
- ▶ Zur Rekonstruktion wird an den betroffenen Stelle die **Anzahl der Wiederholungen** kodiert.
- ▶ Ob einzelne Bits, Zeichen oder ganze Sequenzen kodiert werden, hängt vom jeweiligen Code ab.

Eigenschaften

- ▶ Verlustfreie Kompression
- ▶ Einfach (sogar in Hardware) implementierbar
- ▶ Im Allgemeinen nicht optimal

Verwendung

- ▶ In den verschiedensten Bildkompressionsverfahren
- ▶ Fax
- ▶ Analog- und ISDN-Modems

Übersicht

Einordnung im ISO/OSI-Modell

Sitzungsschicht

Dienste der Sitzungsschicht
Realisierung der Funktionalität der Sitzungsschicht

Darstellungsschicht

Aufgaben der Darstellungsschicht
Zeichensätze und Kodierung
Kodierung
Strukturierte Darstellung
Datenkompression

Anwendungsschicht

Domain Name System (DNS)
Uniform Resource Locator (URL)
HyperText Transfer Protocol (HTTP)
Simple Mail Transfer Protocol (SMTP)
File Transfer Protocol (FTP)

Zusammenfassung

Anwendungsschicht

Die **Anwendungsschicht (Application Layer)** ist die höchste Schnittstelle¹ zwischen Anwendungen und dem Netzwerk. Protokolle der Anwendungsschicht stellen spezifische Dienste bereit.

Beispiele:

- ▶ **Domain Name System (DNS)**
Auflösung sog. vollqualifizierter Domänennamen² in IP-Adressen und umgekehrt.
- ▶ **Hyper Text Transfer Protocol (HTTP)**
Protokoll zum Transfer von Webseiten und Daten.
- ▶ **File Transfer Protocol (FTP)**
Protokoll zum Transfer von Binärdaten von und zu Servern.
- ▶ **Simple Mail Transfer Protocol (SMTP)**
Dient dem Versand von Emails sowie der Kommunikation zwischen Mailservern.
- ▶ **Post Office Protocol (POP) und Internet Message Access Protocol (IMAP)**
Abrufen von bzw. Zugriff auf Emails.
- ▶ **Telnet**
Einfaches Protokoll zur interaktiven Kommunikation mit anderen Hosts (vgl. TCP-Chat Client).
- ▶ **Secure Shell (SSH)**
Verschlüsselte entfernte Anmeldung an einem Host.
- ▶ **Simple Network Management Protocol (SNMP)**
Dient der Überwachung und dem Management von Netzkomponenten.

¹ In vielen Fällen sind Protokolle der Anwendungsschicht Bestandteil der Anwendungen selbst, also innerhalb einer Anwendung implementiert und nicht notwendigerweise Bestandteil des Betriebssystems. Innerhalb gewisser Grenzen trifft dies bereits auf die Schichten 5 und 6 zu.

² Umgangssprachlich als „Webadressen“ bezeichnet.

Domain Name System (DNS) [2, 4, 6, 7]

Motivation:

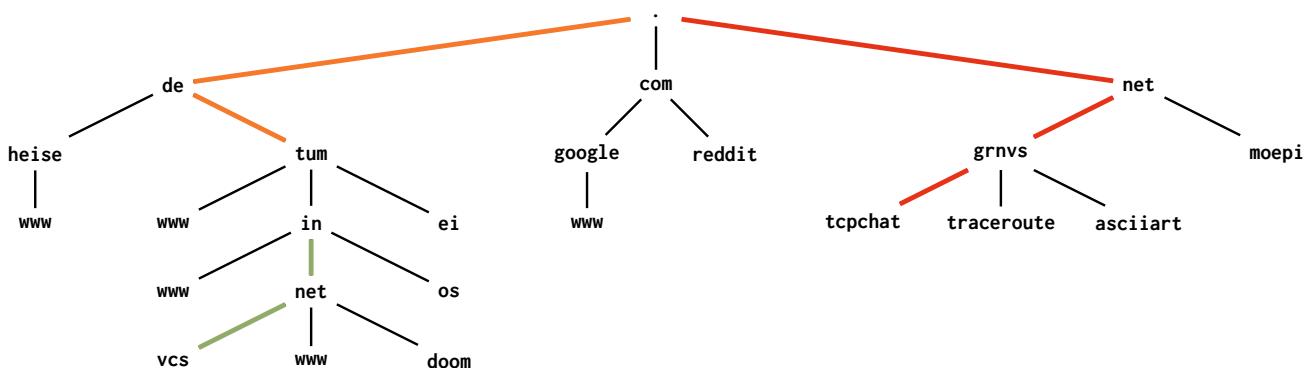
- ▶ Möchte ein Nutzer (Mensch) einen Computer adressieren, z. B. beim Aufruf einer Webseite, will er sich gewöhnlich nicht dessen IP-Adresse merken müssen.
- ▶ Stattdessen adressiert man das Ziel überlicherweise mittels eines hierarchisch aufgebauten Namens, z. B. www.google.com.

Das **Domain Name System (DNS)** besteht aus drei wesentlichen Komponenten:

1. Der **Domain Namespace** ist
 - ▶ ein hierarchisch aufgebauter Namensraum mit
 - ▶ baumartiger Struktur.
2. **Nameservers** speichern
 - ▶ Informationen über den Namensraum,
 - ▶ wobei jeder Server nur kleine Ausschnitte des Namensraums kennt.
3. **Resolver** sind Programme,
 - ▶ die durch Anfragen an Nameserver Informationen aus dem Namespace extrahieren und
 - ▶ anfragenden Clients bzw. Anwendungen zur Verfügung stellen.

Domain Namespace

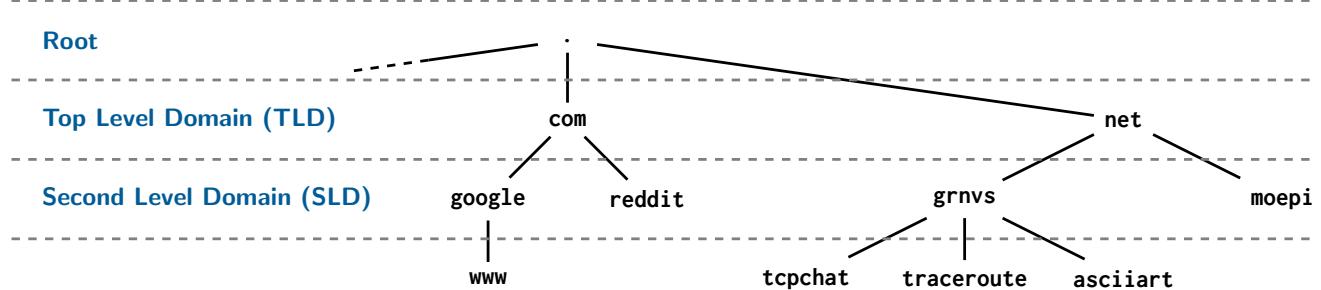
Ein kleiner Auszug aus dem Namespace:



- ▶ Ein **Label** ist ein beliebiger Knoten im Namespace.
- ▶ Ein **Domain Name** ist eine Sequenz von Labels:
 - ▶ Ein **Fully Qualified Domain Name (FQDN)** besteht aus der vollständigen Sequenz von Labels ausgehend von einem Knoten bis zur Wurzel und endet mit einem Punkt, z. B. **tum.de.** oder **tcpchat.grnvs.net..**
 - ▶ Endet er nicht mit einem Punkt, handelt es sich zwar ebenfalls um einen Domain Name, allerdings ist dessen Angabe relativ ausgehend von einem anderen Knoten als der Wurzel, z.B. **vcs.net.in.**
 - ▶ Ein FQDN kann als **Suffix** für einen nicht-qualifizierten Namen verwendet werden, z. B. ergibt **vcs.net.in.** zusammen mit dem FQDN **tum.de.** einen neuen FQDN **vcs.net.in.tum.de..**
 - ▶ Ob ein FQDN existiert (z. B. in eine Adresse aufgelöst werden kann), bleibt zunächst offen.

Übliche Bezeichnungen

Für die ersten drei Hierarchieebenen im Name Space eigene Bezeichnungen üblich:



Darunter liegende Ebenen werden gelegentlich als **Subdomain** bezeichnet.

Vergabe von TLDs und SLDs

- ▶ Top Level Domains werden von der [Internet Corporation for Assigned Names and Numbers \(ICANN\)](#) vergeben.
- ▶ Second Level Domains werden von verschiedenen Registraren vergeben.

Zeichensatz

- ▶ Erlaubt sind nur Buchstaben (A–Z) und Zahlen sowie -, wobei letzterer nicht das erste oder letzte Zeichen sein darf.¹
- ▶ Zwischen Groß- und Kleinschreibung wird nicht unterschieden.

¹ Technisch gesehen könnte ein Eintrag im DNS alle möglichen Oktette enthalten.

Nameserver

Der Namespace wird

- ▶ in Form einer verteilten Datenbank
- ▶ von einer großen Anzahl von Servern gespeichert,
- ▶ wobei jeder Server nur einen kleinen Teil des gesamten Namespaces kennt.

Zu diesem Zweck ist der Namespace in **Zonen** unterteilt:

- ▶ Zonen sind **zusammenhängende Teilbäume** des Namespaces.
- ▶ Eine Zone kann daher mehrere Ebenen des Namespaces umfassen, aber keine Teilbäume ohne gemeinsame Wurzel.
- ▶ Nameserver bezeichnet man als **autoritativ** für die jeweiligen Zonen, die sie speichern.
- ▶ Dieselbe Zone kann auf mehreren Nameservern gespeichert sein.
- ▶ DNS sieht Mechanismen zum Transfer von Zonen zwischen autoritativen Nameservern vor.
- ▶ Dabei gibt es einen primären Nameserver, auf dem Änderungen an einer Zone vorgenommen werden können, sowie beliebig viele sekundäre Nameserver, welche lediglich über Kopien der Zone verfügen.

Nameserver erwarten eingehende Verbindungen auf UDP/TCP 53:

- ▶ Anfragen werden meist an UDP 53 gestellt.
- ▶ Anfragen, die größer als 512 B sind, werden an TCP 53 gestellt.
- ▶ Zone Transfers finden immer über TCP 53 statt.

Resource Records

Die Informationen, die in einer Zone gespeichert sind, bezeichnet man als **Resource Records**:

- ▶ **SOA Record (Start of Authority)** ist ein spezieller Record, der die Wurzel der Zone angibt, für die ein Nameserver autoritativ ist.
- ▶ **NS Records** geben den FQDN eines Nameservers an. Dieser kann auch auf FQDNs in anderen Zonen verweisen.
- ▶ **A Records** assoziieren einen FQDN mit einer IPv4-Adresse.
- ▶ **AAAA Records** assoziieren einen FQDN mit einer IPv6-Adresse.
- ▶ **CNAME Records** sind Aliase, d. h. ein FQDN verweist auf einen "Canonical Name", der selbst wiederum ein FQDN ist.
- ▶ **MX Records** geben den FQDN eines Mailservers für eine bestimmte Domain an, welcher sich nicht notwendigerweise in derselben Zone befinden muss.
- ▶ **TXT Records** assoziieren einen FQDN mit einem String (Text). Wird für unterschiedliche Zwecke verwendet und missbraucht.
- ▶ **PTR Records** assoziieren eine IPv4- oder IPv6-Adresse mit einem FQDN (Gegenstück zu A bzw. AAAA Records).

Hinweise:

- ▶ Mehrere A oder AAAA Records (auch unterschiedlicher Zonen) können mit derselben IP-Adresse assoziiert sein.
- ▶ Für einen FQDN kann es maximal einen CNAME geben. Wenn ein CNAME existiert, handelt es sich um einen Alias, weswegen es keine weiteren Resource Records für den betreffenden FQDN mehr geben darf.
- ▶ Für eine Zone bzw. Domain gibt es üblicherweise mehrere NS bzw. MX Records.

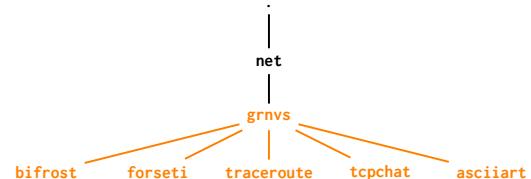
Die Resource Records einer Zone werden auf Nameservern in Form von **Zone Files** gespeichert:

```
$TTL 86400 ; 1 day
grnvs.net. IN SOA bifrost.grnvs.net. hostmaster.grnvs.net. (
    164160 ; serial
    1800   ; refresh (30 minutes)
    300    ; retry (5 minutes)
    604800 ; expire (1 week)
    1800   ; nxdomain (30 minutes)
)
NS    bifrost.grnvs.net.
NS    forseti.grnvs.net.
A     129.187.145.241

$ORIGIN grnvs.net.
bifrost      A      129.187.145.241
forseti      A      78.47.25.36
            AAAA  2a01:4f8:190:60a3::2

$TTL 3600 ; 1 hour
traceroute   A      89.163.225.145
            AAAA  2001:4ba0:ffec:0193::0
tcpchat      A      89.163.225.145
asciart      CNAME svm502.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zur abgebildeten Zone File korrespondierende Teil ist hervorgehoben):



Resolver

Resolver sind Server, die Informationen aus dem DNS extrahieren und das Ergebnis an den anfragenden Client zurückliefern.

- ▶ Da das DNS einer verteilten Datenbank entspricht und kein einzelner Nameserver alle Zonen kennt, sind i. A. mehrere Anfragen notwendig.
- ▶ Resolver fragen dabei schrittweise bei den autoritativen Nameservern der jeweiligen Zonen an.
- ▶ Das Ergebnis wird an den anfragenden Client zurückgegeben und kann (hoffentlich unter Beachtung der TTL im SOA Record der jeweiligen Zone) gecached werden.
- ▶ Stellt ein Client innerhalb dieser Zeit nochmal dieselbe Anfrage, kann diese aus dem Cache beantwortet werden.

(Öffentliche) Resolver sind i. d. R. selbst für keine Zonen selbst autoritativ.

Problem: Woher weiß ein Resolver, wo er anfangen soll?

- ▶ Resolver verfügen über eine statische Liste der 13 Root-Server¹, die für die Root-Zone autoritativ sind.
- ▶ Die Root-Zone wird von der [Internet Corporation for Assigned Names and Numbers](#) verwaltet. Änderungen bedürfen jedoch der Zustimmung durch das [US Department of Commerce](#).
- ▶ Betrieben werden die Root-Server von verschiedenen Organisationen, u. a. ICANN, Versign, U. S. Army, RIPE, NASA, etc.

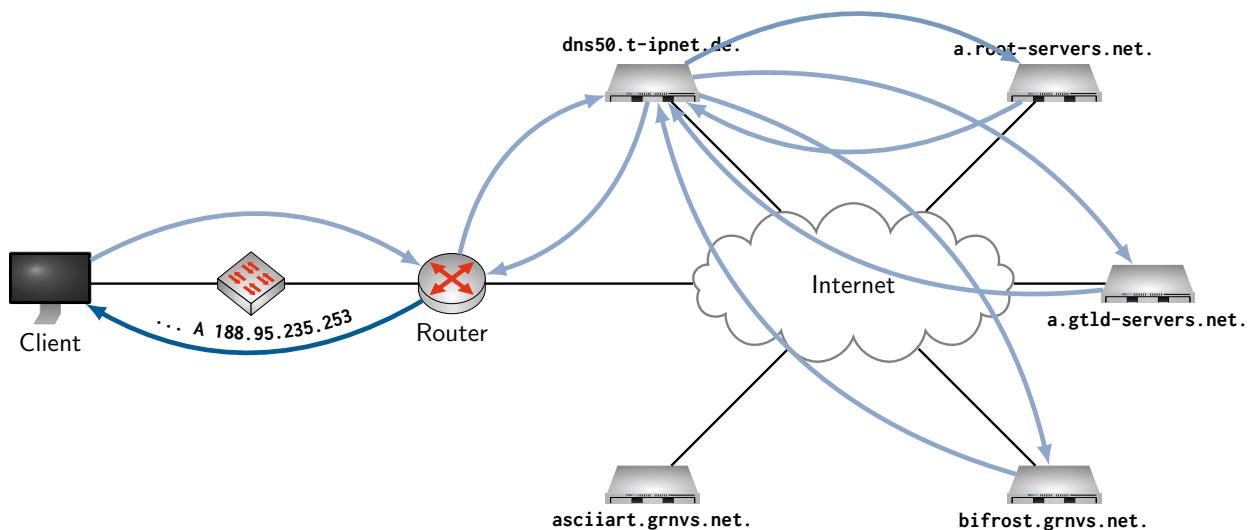
¹ In Wirklichkeit handelt es sich dabei um hunderte Server, welche über die 13 IP-Adressen via Anycast erreichbar sind.

Ausschnitt der Root-Hints

```
; This file holds the information on root name servers needed to
; initialize cache of Internet domain name servers
; (e.g. reference this file in the "cache . <file>" configuration
;       file of BIND domain name servers).
;
; This file is made available by InterNIC
; under anonymous FTP as
;      file          /domain/named.cache
;      on server     FTP.INTERNIC.NET
;      -OR-
;      RS.INTERNIC.NET
;
; last update:   May 23, 2015
; related version of root zone:   2015052300
;
; formerly NS.INTERNIC.NET
;
;          3600000    NS    A.ROOT-SERVERS.NET .
A.ROOT-SERVERS.NET. 3600000    A    198.41.0.4
A.ROOT-SERVERS.NET. 3600000    AAAA   2001:503:ba3e::2:30
;
; FORMERLY NS1.ISI.EDU
;
;          3600000    NS    B.ROOT-SERVERS.NET .
B.ROOT-SERVERS.NET. 3600000    A    192.228.79.201
B.ROOT-SERVERS.NET. 3600000    AAAA   2001:500:84::b
;
; FORMERLY C.PSI.NET
;
;
```

Beispiel: Gewöhnlicher privater Internetanschluss eines motivierten Studenten, der zur Bearbeitung der 4. Programmieraufgabe mittels Webbrower auf den Server mit FQDN `asciart.grnvs.net.` zugreift.

- ▶ Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.¹
- ▶ Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- ▶ DNS-Anfragen des Clients an den Router sind **rekursiv (recursive queries)**.
- ▶ Die eigentliche Namensauflösung wird vom Resolver `dns50.t-ipnet.de.` mittels einer Reihe von **iterativen Anfragen (iterative queries)** erledigt.



¹ Diesen Vorgang bezeichnet man als **Forwarding**. Weswegen könnte das sowohl aus Providersicht als auch aus Nutzersicht vorteilhaft sein?

Das vorherige Beispiel war in einigen Punkten vereinfacht:

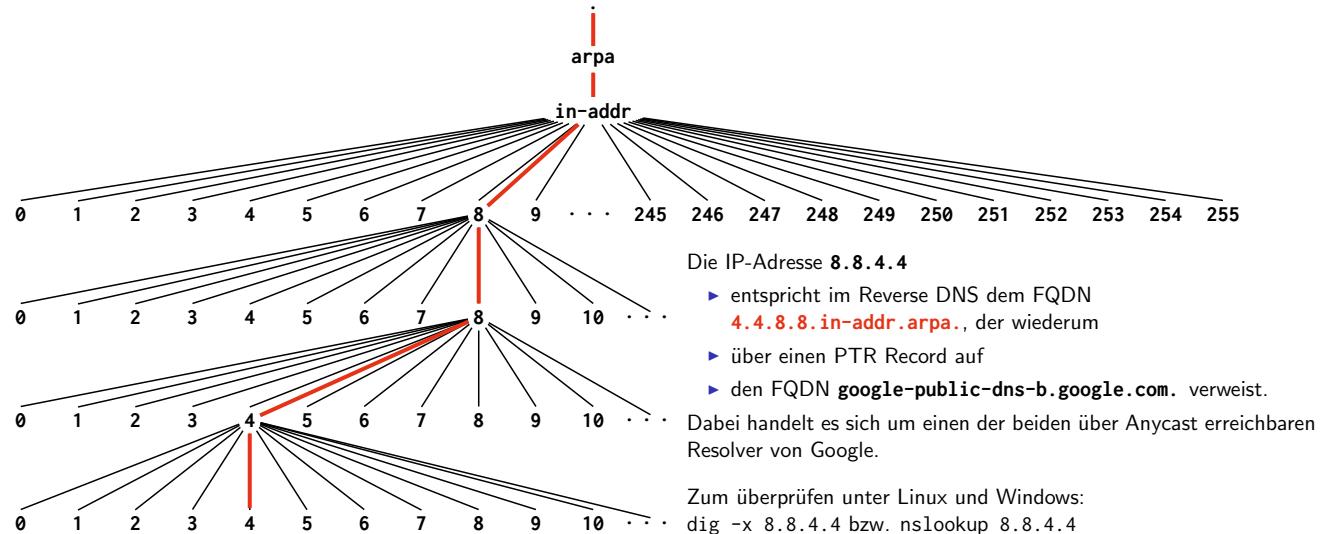
1. Der Resolver hätte von `bifrost.grnvs.net.` in Wirklichkeit kein A Record mit der gesuchten IP-Adresse, sondern lediglich einen CNAME Record erhalten.
 - ▶ Das liegt daran, dass laut der Zone File auf Folie 38 kein A Record sondern nur ein CNAME Record für `asciart.grnvs.net.` existiert (das ist nur dem Beispiel geschuldet).
 - ▶ In Wirklichkeit hätte der Resolver im Anschluss mittels weiterer Anfragen einen autoritativen Nameserver für `svm502.net.in.tum.de.` ermitteln müssen (der FQDN, den der CNAME Record liefert).
 - ▶ Aus Sicht des Routers und des Clients hätte sich aber nichts geändert.
2. Die Antworten der Nameserver auf die iterativen Anfragen liefern lediglich die FQDNs der autoritativen Nameserver für eine Zone.
 - ▶ Der Resolver erhält also z. B. im ersten Schritt von `a.root-servers.net.` lediglich den FQDN der für `.net.` autoritativen Nameserver.
 - ▶ Bevor der Resolver mit seiner eigentlichen Aufgabe fortfahren kann, muss er zunächst den entsprechenden A oder AAAA Record für `a.gtld-servers.net.` bestimmen.
 - ▶ Wenn der FQDN des Nameservers in der gesuchten Zone liegt (wie es hier der Fall ist), müssen in der darüberliegenden Zone so genannte **Glue Records** eingetragen werden. Glue Records enthalten die IP Adresse der gesuchten Nameserver.
 - ▶ Aus Sicht des Routers und Clients ändert sich auch in diesem Fall nichts.

Reverse DNS

Im DNS können mittels **PTR (Pointer) Records** auch FQDNs zu IP-Adressen hinterlegt werden. Dies bezeichnet man als **Reverse DNS**. Hierzu existiert für IPv4 und IPv6 jeweils eine eigene Zone:

- ▶ **in-addr.arpa.** für IPv4
- ▶ **ip6.arpa.** für IPv6

Für IPv4 wird der Namespace unterhalb von **in-addr.arpa.** durch die vier Oktette in umgekehrter Reihenfolge erzeugt:



Für den Namespace unterhalb von **in-addr.arpa.** ergeben sich einige Einschränkungen:

- ▶ Da jede Ebene einem ganzen Oktett entspricht, gibt es maximal vier Ebenen.
- ▶ Subnetze, deren Präfixlänge nicht 8, 16, 24 oder 32 ist, können nicht in getrennten Zonen gespeichert werden (letztere entsprechen einzelnen IP-Adressen).
- ▶ Die Abbildung von Subnetzen anderer Größen ist nur mit Tricks möglich.

Der Namespace für IPv6 (unterhalb von **ip6.arpa.**) ist sehr ähnlich aufgebaut:

- ▶ Die Aufteilung findet an 4 bit-Grenzen anstelle ganzer Oktette statt.
- ▶ Dies erweitert die Möglichkeiten zur Aufteilung in getrennte Zonen.
- ▶ Der Namespace ist infolge der 128 bit langen IPv6-Adressen entsprechend größer.

Uniform Resource Locator (URL)

Mittels FQDN können wir

- ▶ mit Hilfe von DNS das Ziel einer Verbindung auf Schicht 3 identifizieren,
- ▶ aber weder das zu verwendende Anwendungsprotokoll angeben noch die bestimmten Resource adressieren, z. B. eine bestimmte Datei.

Uniform Resource Locator (URL) sind Adressangaben der Form

```
<protocol>://[<username>[:<password>]@]<fqdn>[:<port>][/<path>][?<query>][#<fragment>]
```

- ▶ **<protocol>** gibt das Anwendungsprotokoll an, z. B. HTTP(S), FTP, SMTP, etc.
- ▶ **<username>[:<password>]@** ermöglicht die optionale Angabe eines Benutzernamens und Kennworts.¹
- ▶ **<fqdn>** ist der vollqualifizierte Domain Name², der das Ziel auf Schicht 3 identifiziert.
- ▶ **<port>** ermöglicht die optionale Angabe einer vom jeweiligen well-known Port abweichenden Portnummer für das Transportprotokoll.
- ▶ **/<path>** ermöglicht die Angabe eines Pfads auf dem Ziel relativ zur Wurzel </> der Verzeichnisstruktur.
- ▶ **?<query>** ermöglicht die Übergabe von Variablen in der Form **<variable>=<value>**. Mehrere Variablen können mittels & konateniert werden.
- ▶ **#<fragment>** ermöglicht es einzelne Fragmente bzw. Abschnitte in einem Dokument zu referenzieren.

Beispiele:

- ▶ <http://www.tum.de>
- ▶ <https://vcs.net.in.tum.de/svn/grnvss15/users/ne23puw/exam/IN0010-20150612-0000.pdf>
- ▶ <https://www.mymail.alsoinsecure/mybox?user=student&password=nolongeryourscret>

¹ Wenn Sie auf diese Art Benutzername und Kennwort angeben, wissen Sie hoffentlich, dass Sie Ihre Anmeldeinformationen genausogut via Rundfunk bekanntgeben könnten...

² Innerhalb von URLs (bzw. URIs) wird der terminierende . eines FQDNs i. d. R. weggelassen.

Aber ich tippe doch nur google.de ein und es funktioniert trotzdem!

Es funktioniert in den meisten Fällen aus folgenden Gründen:

1. Sie tippen das in Ihrem Webbrowser ein. Ihr Webbrowser weiß, dass mit höchster Wahrscheinlichkeit eine Verbindung über HTTP(S) und nicht über FTP oder gar SMTP (Email) gewünscht gefragt ist.
2. Google hat A bzw. AAAA Records für **google.de** gesetzt, so dass sich der Aufruf genauso wie mit **www.google.de** verhält. Das ist nicht selbstverständlich: Vergleichen Sie <http://in.tum.de> und <http://ei.tum.de>.
3. Der Domain Name, der nicht mal ein FQDN ist, wird stillschweigend als solcher interpretiert.

Übrigens:

- ▶ Dass es sich um einen „Webserver“ handelt, entscheidet sich nicht daran, dass der FQDN **www.webserver.de**. lautet.
- ▶ Der „Webserver“ könnte genauso gut unter dem FQDN **mail.mydomain.de**. erreichbar sein.
- ▶ Nicht einmal die Portnummer entscheidet hierüber:
 - ▶ Natürlich ist es üblich, dass ein „Webserver“ über TCP 80 und ein Mailserver über TCP 25 erreichbar ist (well-known Ports).
 - ▶ Es hindert uns aber nichts¹ daran, einen „Webserver“ auf TCP 25 und einen Mailserver auf TCP 80 erreichbar zu machen.
 - ▶ Ob und wann das sinnvoll ist, sei dahingestellt.

¹ Root- bzw. Administratorrechte vorausgesetzt, da man andernfalls unter den gängigen Betriebssystemen keinen listening Socket < 1024 anlegen kann.

HyperText Transfer Protocol (HTTP)

Das im Internet am häufigsten zur Datenübertragung zwischen Client und Server genutzte Protokoll ist [Hyper Text Transfer Protocol \(HTTP\)](#):

- ▶ HTTP definiert, welche Anfragen ein Client stellen darf und wie Server darauf zu antworten haben.
- ▶ Mit einem HTTP-Kommando wird höchstens ein „Objekt“ (Text, Grafik, Datei, etc.) übertragen.
- ▶ Kommandos werden als ASCII-kodierter Text interpretiert, d. h. es handelt sich um ein textbasiertes Protokoll.
- ▶ Eingehende HTTP-Verbindungen werden auf dem well-known Port TCP 80 erwartet.
- ▶ Bei HTTP 1.0 wird nach jedem Anfrage/Antwort-Paar die TCP-Verbindung wieder abgebaut.

Frage: Was ist die offensichtliche Problematik beim Abruf einer Webseite, welche aus vielen Teilen zusammengesetzt ist (z. B. eine große Anzahl kleiner Grafiken)?

Für jedes Objekt muss eine neue TCP-Verbindung aufgebaut werden, was entsprechend Zeit kostet und bei vielen kleinen Objekten einen entsprechenden Overhead bedeutet.

Verbesserung mit HTTP 1.1:

- ▶ TCP-Verbindungen überdauern mehrere Anfragen.
- ▶ HTTP kann dennoch als zustandslos¹ angesehen werden, da die einzelnen Anfragen voneinander unabhängig sind.

HTTP Message Types und Headers

HTTP unterscheidet grundsätzlich zwischen zwei Nachrichtentypen: [Request](#) und [Response](#)

1. Request (vom Client zum Server), enthält

- ▶ eine [Method](#), welche die vom Client gewünschte Aktion beschreibt, z. B. Übertragung einer bestimmten Resource vom Server zum Client,
- ▶ Pfad und Query-Parameter des [Uniform Resource Locator \(URL\)](#), welcher die angefragte Resource näher beschreibt und
- ▶ eine Reihe weiterer Headerfelder, in denen unter anderem die folgenden Informationen enthalten sein können:
 - ▶ FQDN des angefragten Hosts¹
 - ▶ Zeichensatz und Encoding, in dem die Antwort erwartet wird
 - ▶ Von wo eine Anfrage kam (z. B. Weiterleitung von einer anderen Webseiten über den Referrer²)
 - ▶ Den User-Agent, also die verwendete Client-Software

2. Response (vom Server zum Client), enthält

- ▶ eine Status-Line Status (numerischer Code + Text zur Angabe von Fehlern),
- ▶ einen Response Header mit ggf. weiteren Optionen und
- ▶ den mittels CRLF (Carriage Return Line Feed) abgetrennten [Body](#), welcher die eigentlichen Daten enthält.

¹ Über dieses Feld kann der Server bei so genannten Virtual Hosts unabhängig von der IP entscheiden, welche Daten ausgeliefert werden sollen

² In HTTP in falscher Schreibweise als Referer[sic]

HTTP Methods und Response Codes

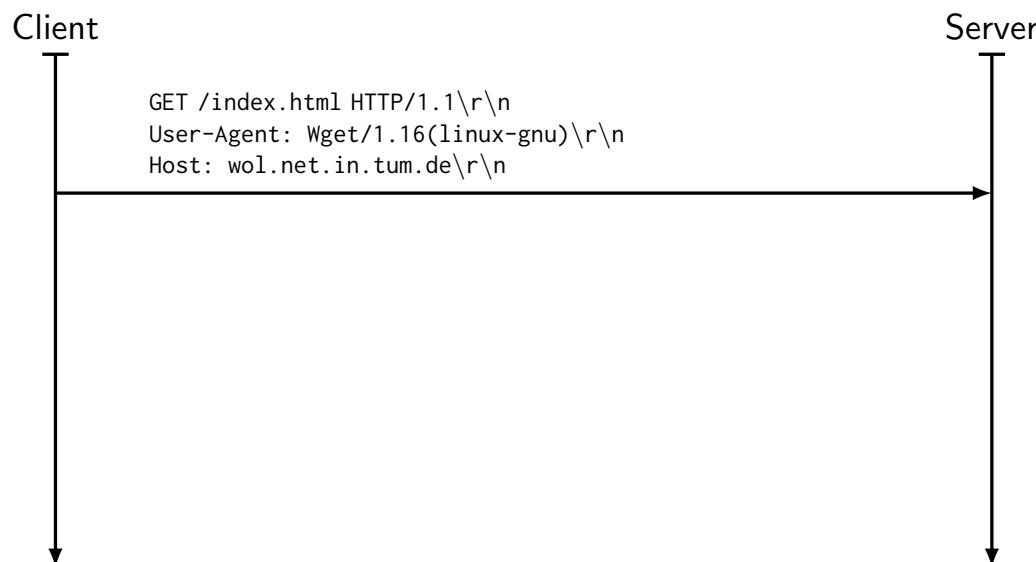
Clients verwenden eine übersichtliche Anzahl unterschiedlicher Kommandos (Methods):

- ▶ **GET** – Anfrage zur Übertragung eines bestimmten Objekts vom Server
- ▶ **HEAD** – Anfrage zur Übertragung des Headers eines bestimmten Objekts (z. B. bestehen Webseiten aus mehreren Sektionen, wovon einer als Header bezeichnet wird)
- ▶ **PUT** – Übertragung eines Objekts vom Client zum Server, welches ggf. ein bereits existierendes Objekt überschreibt
- ▶ **POST** – Übertragung eines Objekts vom Client zum Server, welches ggf. an ein bereits existierendes Objekt angehängt wird (z. B. Anhängen von Text)
- ▶ **DELETE** – Löschen eines Objekts vom Server
- ▶ **LINK / UNLINK** – Herstellung bzw. Entfernung einer Beziehung zwischen zwei unterschiedlichen Objekten

Die Antworten des Servers geben mittels der Status-Line das Ergebnis der Anfrage an, z. B.

- ▶ **200** – OK
- ▶ **400** – Bad Request
- ▶ **401** – Unauthorized
- ▶ **403** – Forbidden
- ▶ **404** – Not Found
- ▶ etc.

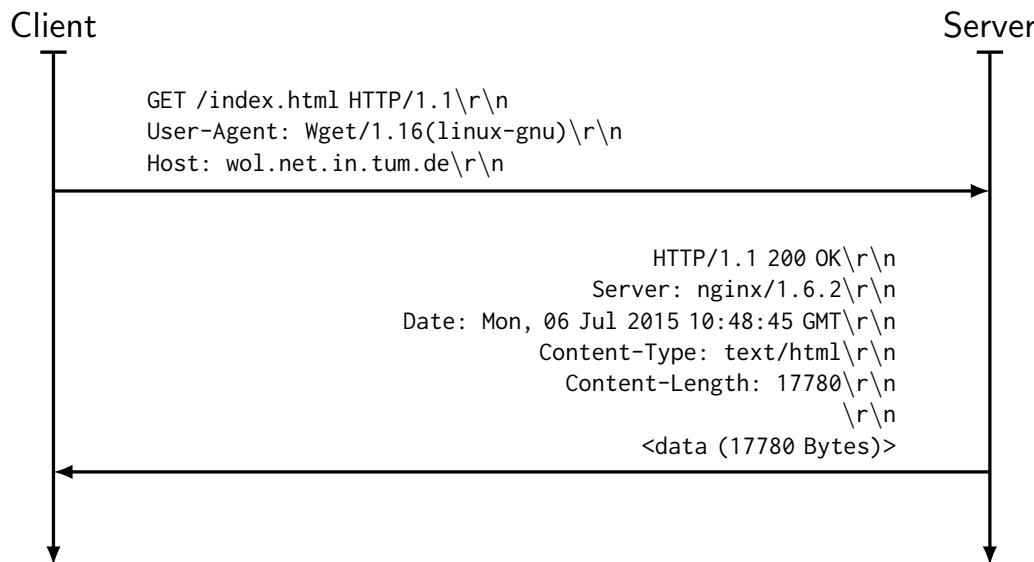
Beispiel: Zugriff auf <http://vcs.net.in.tum.de/svn/grnvss15/pub>



- ▶ Client fordert mittel **GET** eine bestimmte Resource relativ zum Document Root an.
- ▶ Unter anderem sendet der Client den benutzten User-Agent, d. h. den verwendeten Browser.
- ▶ **Host** gibt noch einmal explizit den FQDN des Webservers an.¹

¹ Auf den ersten Blick ist dies überflüssig, da dieser ja bereits mittels DNS in eine IP-Adresse aufgelöst wurde und so das Ziel auf der Netzwerkschicht identifiziert. Das nochmalige Senden ermöglicht es allerdings, mehrere Webserver mit unterschiedlichen FQDNs bzw. URLs unter derselben IP-Adresse und Portnummer zugänglich zu machen.

Beispiel: Zugriff auf <http://vcs.net.in.tum.de/svn/grnvss15/pub>



- ▶ Der Server antwortet mit dem entsprechenden Status Code.
- ▶ Zusätzlich werden verschiedene Optionen angegeben², hier insbesondere der Content-Type. Häufig werden auch das Content-Encoding und der Zeichensatz angegeben.
- ▶ Am Ende folgen die (entsprechend kodierten) Daten.

² Aus Platzgründen sind nicht alle Optionen abgebildet.

Verschlüsselung bei HTTP

- ▶ HTTP selbst kennt keine Mechanismen zur Verschlüsselung übertragener Daten.
- ▶ Es besteht aber die Möglichkeit, zwischen Transport- und Anwendungsschicht entsprechende Protokolle zu verwenden.

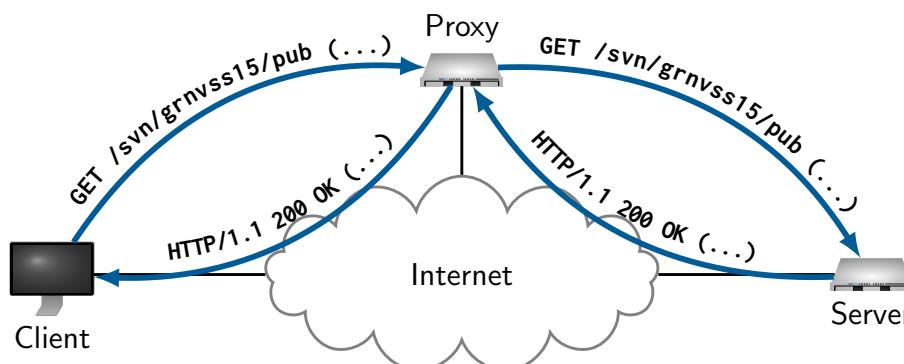
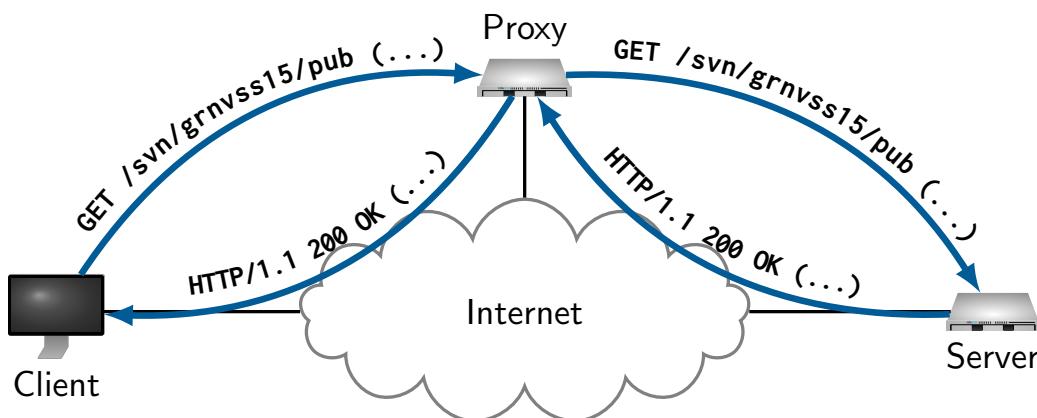
Hyper Text Transfer Protokol Secure (HTTPS)

- ▶ Nutzt [Transport Layer Security \(TLS\)](#), ein Verschlüsselungsprotokoll oberhalb der Transportschicht.
- ▶ TLS ver- und entschlüsselt den Datentransfer.
- ▶ HTTP selbst bleibt unverändert.
- ▶ Zur Unterscheidung von unverschlüsselten Verbindungen wird TCP 443 anstelle von TCP 80 verwendet.
- ▶ Für TLS ist HTTP nur ein Datenstrom.

HTTP Proxy

Manchmal ist es sinnvoll oder notwendig, dass ein Client keine Ende-zu-Ende-Verbindung zu einem HTTP-Server aufbaut, sondern einen dazwischenliegenden **Proxy** verwendet:

- ▶ Anstelle den Zielserver direkt zu kontaktieren, werden HTTP-Anfragen an einen HTTP-Proxy geschickt.
- ▶ Dieser nimmt die Anfragen entgegen, baut seinerseits eine neue Verbindung zum eigentlich Zielserver (oder einem weiteren Proxy) auf, und stellt die Anfrage anstelle des Clients.
- ▶ Die entsprechende Antwort wird dem zunächst dem Proxy gesendet, welcher sie (hoffentlich unmodifiziert) dem Client zustellt.
- ▶ Zur Unterscheidung von normalen HTTP-Servern verwenden Proxies häufig andere Portnummern wie z. B. TCP 3128.



- ▶ Der Client selbst bleibt hinter dem Proxy (weitgehend) verborgen, d. h. der Webserver sieht lediglich den Proxy, nicht aber die Clients.¹
- ▶ Der Proxy kann Anfragen cachen, d. h. bei mehrfachen identischen Anfragen (auch unterschiedlicher Clients) können Inhalte aus einem Cache geliefert werden.
 - ▶ Sind die Inhalte noch aktuell?
 - ▶ Wie und wann müssen Inhalte aktualisiert werden?
- ▶ Proxies können auch **transparent** arbeiten, d. h. ohne Wissen der Clients.
 - ▶ In Firmennetzwerken häufig anzutreffen.
 - ▶ In einigen Fällen lassen sich Proxies sogar dazu „missbrauchen“ TLS-verschlüsselte Verbindungen zu überwachen.²

¹ Davon unbetroffen sind natürlich Informationen, die der Client mittels POST an den Server übermittelt.

² Dies setzt voraus, dass Clients das SSL-Zertifikat des Proxies als vertrauenswürdig einstufen, was sich im Rahmen administrierter Netzwerke leicht bewerkstelligen lässt.

Simple Mail Transfer Protocol (SMTP)

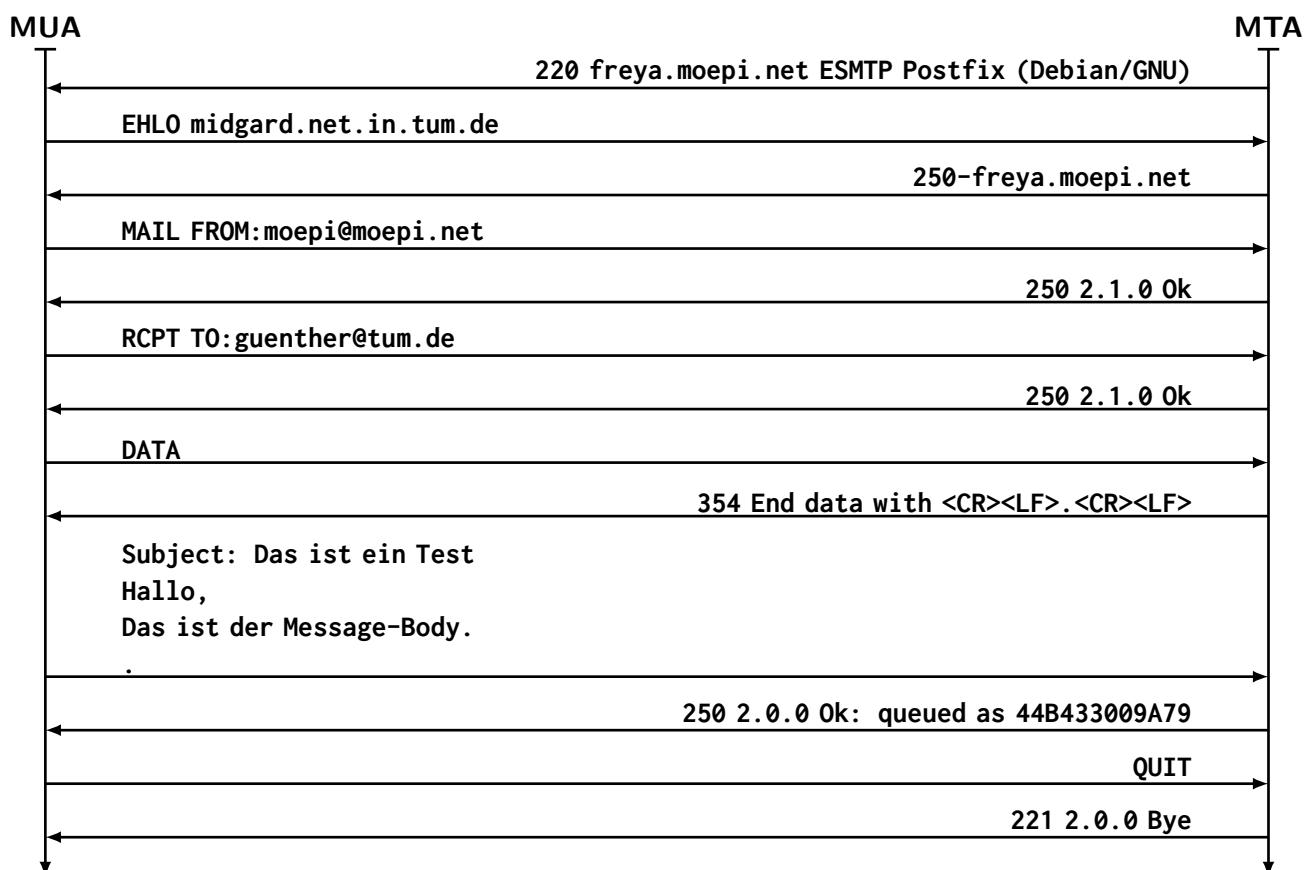
Das Simple Mail Transfer Protocol (SMTP) ist ein textbasiertes Protokoll und dient

- ▶ dem Versenden von Emails, d. h. dem Transport vom Mail User Agent (MUA) zu einem Mail Transfer Agent (MTA), sowie
- ▶ dem Transport von Emails zwischen MTAs.

In Empfangsrichtung werden i. d. R.

- ▶ Post Office Protocol (POP) oder
- ▶ Internet Message Access Protocol (IMAP) verwendet.

Beispiel: SMTP



Beispiel: SMTP (Fortsetzung) Nach Erhalt der Email versucht der MTA diese zuzustellen:

- ▶ Die Empfängeradresse `guenther@tum.de` enthält der FQDN des Ziels.
- ▶ Mittels DNS werden die MX-Records der Domain `tum.de`. identifiziert:

```
tum.de. 3600 IN MX 100 postrelay2.lrz.de.  
tum.de. 3600 IN MX 100 postrelay1.lrz.de.
```

Die beiden Einträge enthalten jeweils die TTL, Typ des DNS-Records, eine Präferenz (kleinere Werte bedeuten höhere Präferenz) sowie den FQDN des jeweiligen Mailservers.

- ▶ Der MTA wird nun eine weitere SMTP-Verbindung zu einem der beiden Mailserver aufbauen und versuchen, die Email weiterzureichen.
 - ▶ Falls erfolgreich, wird der MTA die Nachricht löschen. Der Absender wird nicht benachrichtigt.
 - ▶ Falls der Zielserver nicht erreichbar ist, wird der MTA es in regelmäßigen Zeitabständen wieder versuchen, nach einiger Zeit eine Delay-Notification an Absender zurücksenden und den Client im Fall einer Aufgabe benachrichtigen.
 - ▶ Falls der Zielserver die Annahme der Nachricht verweigert, wird der Absender mit dem entsprechenden Grund benachrichtigt.

Die Email wird auf dem Mailserver dem Empfänger zur Verfügung gestellt:

- ▶ POP3 ermöglicht den „Abruf“ einer Email, welche im Anschluss vom Server gelöscht wird.
- ▶ IMAP4 ermöglicht die Synchronisation einer Mailbox zwischen dem Server und einem oder mehreren Mailclients.

Email (Beispiel):

```
From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de Wed Jul 1 19:20:20 2015  
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>  
Delivered-To: moepi@moepi.net  
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])  
    by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0  
    for <moepi@moepi.net>; Wed, 1 Jul 2015 19:18:36 +0200 (CEST)  
Received: from postforw2.mail.lrz.de (lxmhs62.srv.lrz.de [IPv6:2001:4ca0:0:116::a9c:63e])  
    by forwout2.mail.lrz.de (Postfix) with ESMTP id 3mM8Qy1SDPz8q  
    for <moepi@moepi.net>; Wed, 1 Jul 2015 19:18:34 +0200 (CEST)  
(...)  
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])  
    (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))  
    (Client did not present a certificate)  
    by postrelay1.lrz.de (Postfix) with ESMTPS  
    for <guenther@tum.de>; Wed, 1 Jul 2015 19:18:33 +0200 (CEST)  
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])  
    by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;  
    Wed, 1 Jul 2015 19:18:30 +0200 (CEST)  
Message-ID: <559420E5.2000101@net.in.tum.de>  
Date: Wed, 01 Jul 2015 19:18:29 +0200  
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbGx1bm3DvGxsZXI=?=  
    <gallenmu@net.in.tum.de>  
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0  
MIME-Version: 1.0  
To: Stefan <mailto@example.com>  
CC: =?UTF-8?B?U3RlcGhhbiBHW7xudGhlcg==?= <guenther@tum.de>,  
    Johannes Naab <naab@net.in.tum.de>  
Subject: [GRNVS-next] Antrittsgespräch  
References: <21ed7f299038e5f6fd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>  
In-Reply-To: <554B3F8A.7090403@gmx.de>  
Content-Type: text/plain; charset=utf-8  
Content-Transfer-Encoding: 8bit  
  
Hi Stefan,  
wir haben fuer Freitag 11 Uhr einen Termin mit (...)
```



Hinweise:

- ▶ Ein MTA akzeptiert Emails i. d. R. nur für die eigenen Domains, nicht aber für fremde Ziel-Domains. Andernfalls spricht man von einem **Open Relay**, der schnell zum Versand von Spam missbraucht wird und auf Blacklists landet.
- ▶ Authentifizierung zwischen MTAs ist unüblich da technisch nicht umsetzbar.¹
- ▶ Sowohl SMTP als auch POP3 und IMAP können mittels TLS verschlüsselt werden. Die Portnummern ändern sich dabei (vgl. HTTP und HTTPS).
- ▶ Die Verbindungen zwischen MTAs können **opportunistisch verschlüsselt** werden, d. h. es wird kein Wert darauf gelegt die Gegenseite zu authentifizieren. Es wird lediglich der Transportweg verschlüsselt. Der Nutzer bzw. MUA hat hierauf keinen Einfluss.

¹ Ausnahme stellen MTAs dar, welche als sog. **Smarthost** oder **Relay-Host** arbeiten und Emails von anderen (bekannten) MTAs entgegennehmen, da diese selbst nicht in der Lage sind, Emails erfolgreich an andere MTAs zuzustellen. Beispiele sind MTAs mit dynamisches IP-Adressen, welche meist keine PTR-Records besitzen und infolge dessen häufig von anderen MTAs zurückgewiesen werden (Spamschutz).

File Transfer Protocol (FTP) [8]

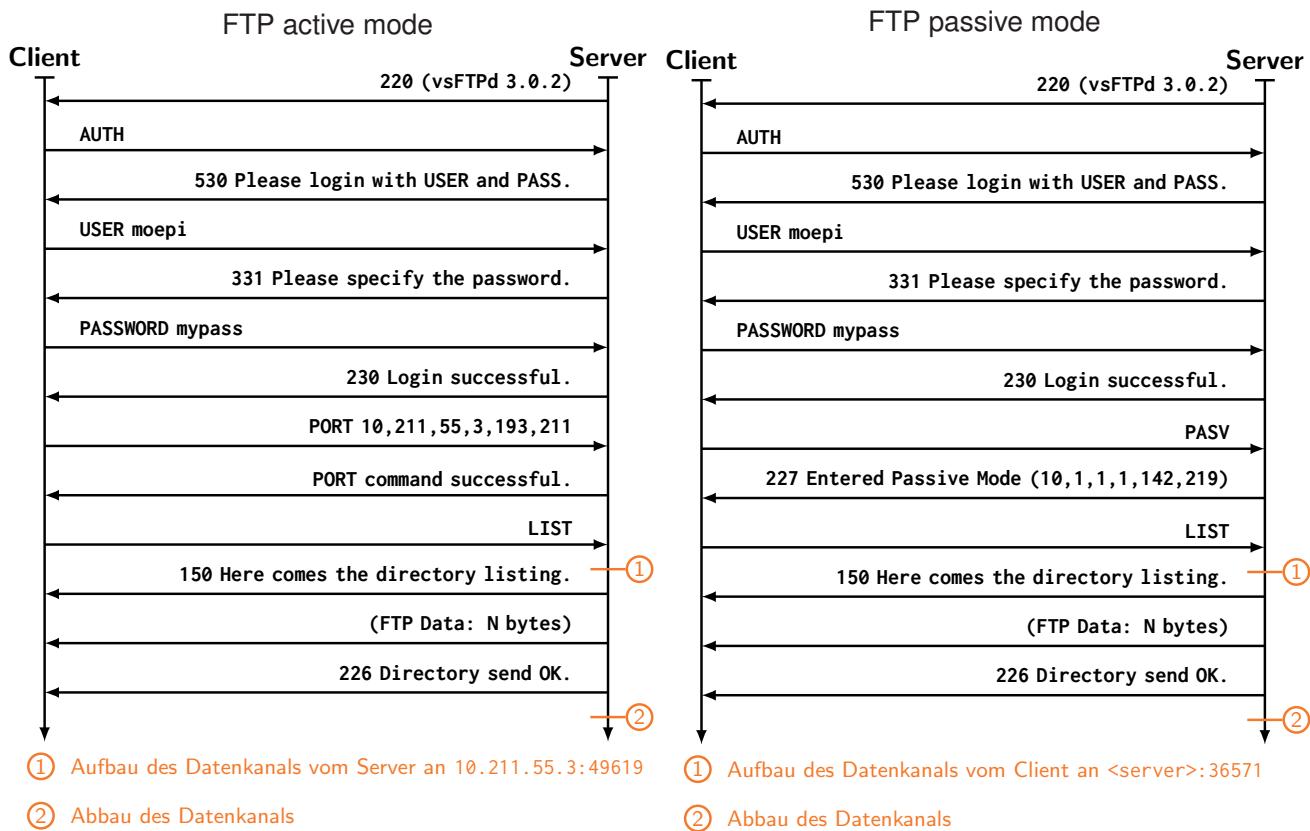
Das **File Transfer Protocol (FTP)** ist ein weiteres Protokoll zum Transfer von Daten (Text wie Binaerdaten). Unterschiede zu HTTP:

- ▶ FTP nutzt zwei getrennte TCP-Verbindungen:
 1. Kontrollkanal zur Übermittlung von Befehlen und Statuscodes zwischen Client und Server.
 2. Datenkanal zur Übertragung der eigentlichen Daten Daten.
- ▶ Der Kontrollkanal bleibt über mehrere Datentransfers hinweg bestehen, d. h. FTP ist **statefull**.
- ▶ FTP erfordert grundsätzlich eine Art von Authentifizierung (anonymer Zugang mittels Benutzername **anonymous** und beliebigem Passwort sofern konfiguriert)

FTP arbeitet entweder im **active** oder **passive mode**:

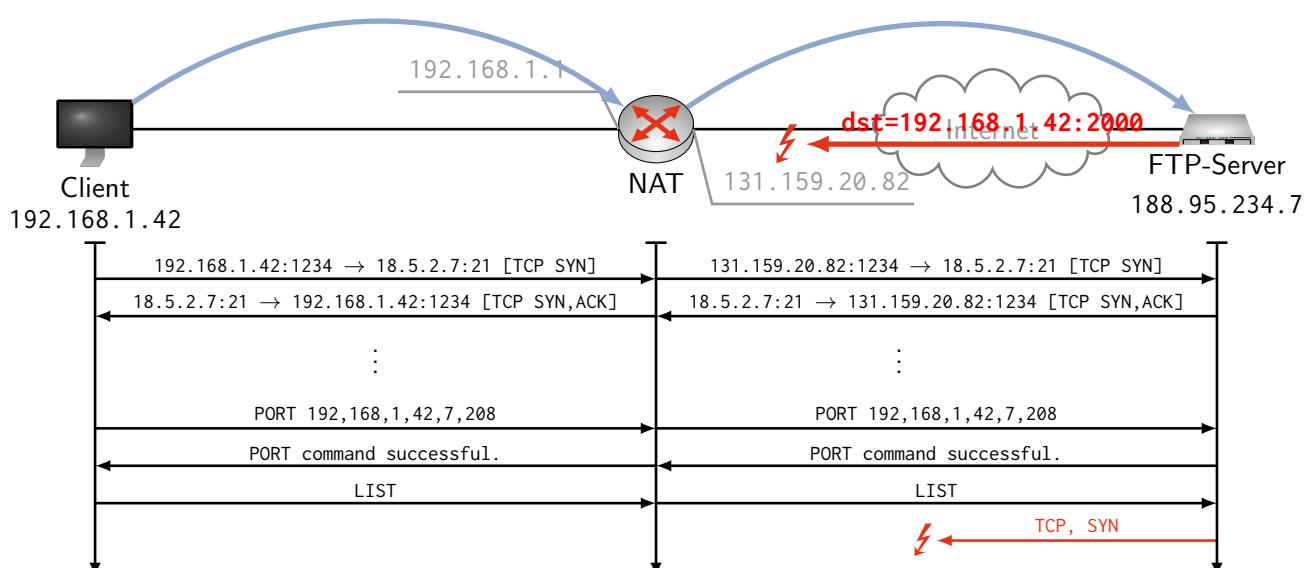
- ▶ In beiden Fällen baut der Client den Kontrollkanal zum Server auf TCP 21 auf.
- ▶ Im **active mode** teilt der Client mittels des **PORT**-Kommandos dem Server eine zufällige Portnummer mit, auf der der Server vom Quellport TCP 20 eine neue TCP-Verbindung zum Client aufbaut, die als Datenkanal verwendet wird.
- ▶ Im **passive mode** sendet der Client das Kommando **PASV** über den Kontrollkanal und erhält vom Server IP-Adresse und Portnummer, zu der der Client eine zweite TCP-Verbindung aufbauen soll, die wiederum als Datenkanal verwendet wird.

Beispiel:



FTP und NAT

Was ist das Problem mit FTP active mode und NAT?

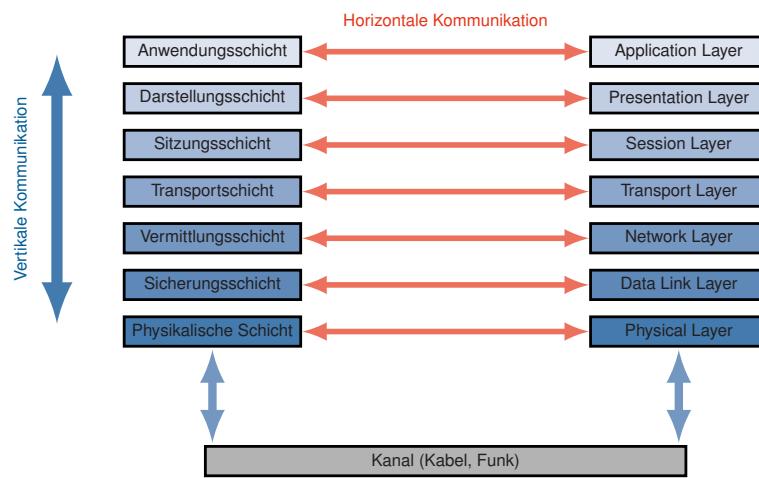


- Der Server versucht infolge des LIST-Kommandos eine Verbindung zu 192.168.1.42:2000 – wie zuvor über das PORT-Kommando ausgehandelt – aufzubauen.
- Das schlägt allein schon wegen der privaten IP-Adresse fehl.
- Selbst Wenn die Adresse öffentlich erreichbar wäre, hätte das NAT keinen passenden Eintrag für eine eingehende Verbindung auf TCP 2000.

Lösungen:

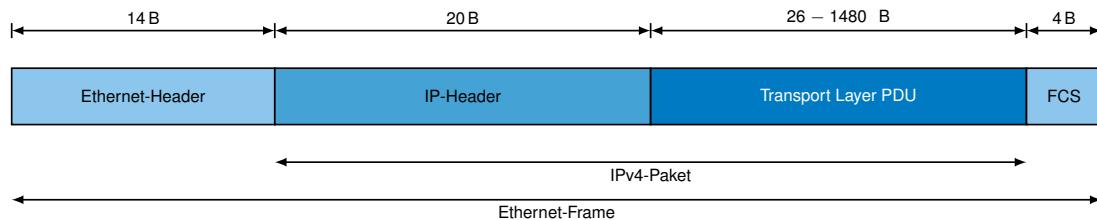
1. Die NAT-Implementierung wird so erweitert, dass sie FTP unerstüttzt.
 - ▶ NAT müsste die die L7-PDU auf das Vorhandensein von PORT prüfen und bei Auftreten sowohl die private IP-Adresse durch eine öffentlich gültige ersetzen als auch einen Eintrag in der NAT-Tabelle für den entsprechenden Port vornehmen.
2. FTP passive mode
 - ▶ Da hier der Server keine Verbindung zum Client aufbaut, sondern dieser eine zweite Verbindung zum Server aufbaut, sollte es mit NAT vorerst keine Probleme geben.
 - ▶ Problematisch wird es, wenn der Server selbst hinter einem NAT steht, das lediglich TCP 21 an eine bestimmte private Adresse weiterleitet.
 - ▶ Ebenfalls problematisch ist der Einsatz von **Firewalls**, die nur Verbindungen zu bestimmten Portnummern erlauben.

Zusammenfassung: ISO/OSI Modell



- ▶ Das **ISO/OSI Modell** unterteilt den Kommunikationsvorgang in 7 Schichten
- ▶ Es spezifiziert, welche Dienste in den verschiedenen Schichten zu erbringen sind
- ▶ Es wird jedoch keine Implementation vorgegeben
- ▶ Die n-te Schicht des Quellsystems kommuniziert nie direkt mit der n-ten Schicht des Ziels
- ▶ Kommunikation erfolgt stets erst vertikal im Stack, dann horizontal im Kanal und abschließend wieder vertikal
- ▶ Das Internet ist über einen TCP/IP-Stack realisiert, welcher weniger Schichten umfasst

Zusammenfassung: ISO/OSI Modell



- ▶ Den Nutzdaten wird auf jeder Ebene ein entsprechender Header vorangestellt
- ▶ Die tatsächlich transportierten Daten beinhalten damit für jede aktive Schicht einen Header

Zusammenfassung: Schichten

Physikalische Schicht

Der von der ersten Schicht angebotene Dienst ist die Punkt-zu-Punkt-Übermittlung von reinen Bits in Form von physikalisch messbaren Signalen.

- ▶ Generierung von Signales aus Bits (Impulsformung), welche auf Zielseite wiedererkannt werden müssen (Detektion)
- ▶ Auftragen der Signale auf eine Trägerwelle (Modulation)
- ▶ Als Trägermedium werden i. d. R. elektromagnetische Wellen genutzt
- ▶ Teilweise ausgleichen der inhärenten Unzuverlässigkeit des Kanals mithilfe einer Kanalkodierung

Sicherungsschicht

Die Sicherungsschicht übernimmt die Abstraktion eines physischen Kanals auf eine logische Direktverbindung.

- ▶ Erkennen von Übertragungsfehlern, nach Möglichkeit Korrektur
- ▶ Verhindern einer Überforderung des Zielsystems bei der Kommunikation (Flusskontrolle)
- ▶ Regelung des Medienzugriffs auf ein gemeinsam genutztes Kommunikationssystem

Zusammenfassung: Schichten

Vermittlungsschicht

Die Vermittlungsschicht verbindet Systeme über beliebig viele Direktverbindungen hinweg.

- ▶ Ermöglichung von Kommunikation über Direktverbindungen und Subnetze
- ▶ Bereitstellung einer eindeutigen und logischen Adressierung von Hosts
- ▶ Bestimmung möglichst optimaler Vermittlungspfade zwischen kommunizierenden Hosts (Routing)

Transportschicht

Die Transportschicht realisiert eine Ende-zu-Ende-Kommunikation zwischen Prozessen auf unterschiedlichen Hosts.

- ▶ Flusskontrolle zur Verhinderung von Überlast beim Empfänger
- ▶ Staukontrolle zur Vermeidung von Überlastsituationen im Netz
- ▶ Multiplexing von Datenströmen verschiedener Anwendungen bzw. ihrer Instanzen
- ▶ Bereitstellung verbindungsloser bzw. -orientierter Transportmechanismen

Zusammenfassung: Schichten

Sitzungsschicht

Die Sitzungsschicht erlaubt die Etablierung eines gemeinsamen Kommunikationszustandes, der mehrere Verbindungen auf der Transportschicht umfassen kann.

- ▶ Stellt Synchronisationspunkte für die Kommunikation zur Verfügung
- ▶ Ermöglicht damit das Suspendieren und Wiederaufnehmen von Kommunikationen
- ▶ Realisiert einen Koordinationsmechanismus für Kommunikationspartner

Darstellungsschicht

Die Darstellungsschicht ermöglicht die Bereitstellung eines abstrakten Formats zur Repräsentation der übertragenen Daten.

- ▶ Abstraktion von anwendungsspezifischer Syntax
- ▶ Bereitstellung von Datenkompression
- ▶ Ver- und Entschlüsselung der Kommunikationsdaten
- ▶ Umwandlung der Kommunikationsdaten

Zusammenfassung: Schichten

Anwendungsschicht

Die Anwendungsschicht beinhaltet alle Protokolle, welche direkt mit Anwendungen interagieren und deren Datentransport realisieren.

- ▶ Stellen einen Dienst für den User zur Verfügung
- ▶ Besitzen kein gemeinsames Dienste-Interface, da ihr Einsatzzweck sehr unterschiedlich ist

Die durch das ISO/OSI Modell realisierten Abstraktionen ermöglichen es Prozessen auf verschiedenen Systemen, transparent über ein Netz zu kommunizieren, ohne sich mit dem eigentlichen Übermittlungsvorgang auseinanderzusetzen zu müssen.

Die Schichtarchitektur selbst erlaubt, dass die Technologien einzelner Segmente ausgetauscht werden können, ohne dass Änderungen am restlichen Stack vorgenommen werden müssen (z. B. kabelgebundene Kommunikation vs. WLAN).

Zusätzlich ist es möglich, Adapter zur Verfügung zu stellen, welche die gleichzeitige Nutzung verschiedener Technologien während ein und demselben Datenaustausch ermöglichen (z. B. WLAN Access Point).

Bibliography I

- [1] The JSON Data Interchange Format, 2013. <http://www.ecma-international.org/publications/standards/Ecma-404.htm>.
- [2] R. Braden. Requirements for Internet Hosts – Communication Layers, 1989. <https://tools.ietf.org/html/rfc1122>.
- [3] T. Bray. The JavaScript Object Notation (JSON) Data Interchange Format, 2014. <https://tools.ietf.org/html/rfc7159>.
- [4] R. Elz and R. Bush. Clarifications to the DNS Specifications, 1997. <https://tools.ietf.org/html/rfc2181>.
- [5] S. Josefsson. The Base16, Base32, and base64 data encodings, 2006. <https://tools.ietf.org/html/rfc4648>.
- [6] P. Moackapetris. Domain Names – Concepts and Facilities, 1987. <https://tools.ietf.org/html/rfc1034>.
- [7] P. Moackapetris. Domain Names – Implementation and Specification, 1987. <https://tools.ietf.org/html/rfc1035>.
- [8] J. Postel and J. Reynolds. File Transfer Protocol (FTP), 1985. <https://tools.ietf.org/html/rfc959>.