Netzwerke

Theorie

Client-Server-Modell

Server: Ein Computer, der Informationen zur Verfügung stellt.

Client-Server-Modell

Server: Ein Computer, der Informationen zur Verfügung stellt.

Client: Ein Computer, der auf diese Informationen zugreift.

Client-Server-Modell

Server: Ein Computer, der Informationen zur Verfügung stellt.

Client: Ein Computer, der auf diese Informationen zugreift.

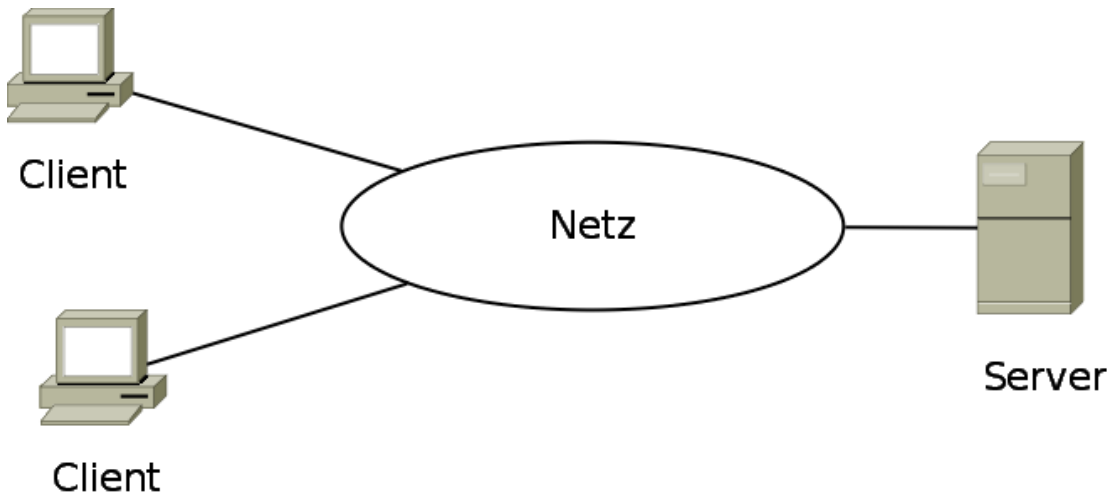


Abbildung 1: Netzwerk mit einem Server und zwei Clients

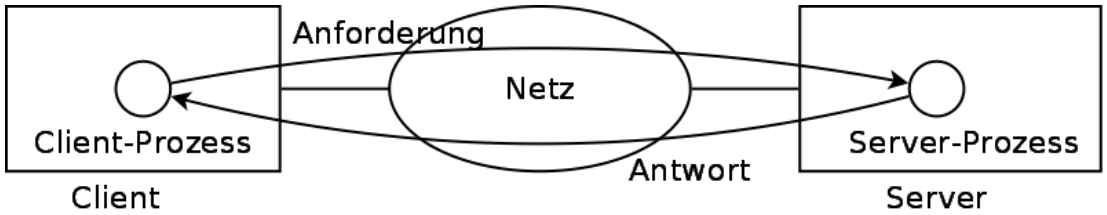


Abbildung 2: Anforderungen und Antworten im Client-Server-Modell

Der Client-Prozess sendet über das Netzwerk eine Nachricht (Anforderung, Request) an den Server-Prozess und wartet dann auf eine Antwort.

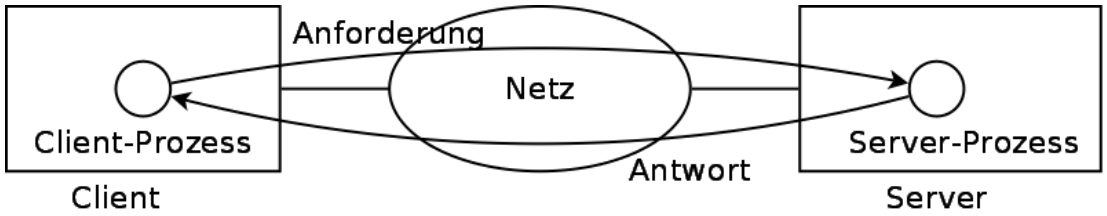


Abbildung 2: Anforderungen und Antworten im Client-Server-Modell

Der Client-Prozess sendet über das Netzwerk eine Nachricht (Anforderung, Request) an den Server-Prozess und wartet dann auf eine Antwort.

Wenn der Server-Prozess die Anforderung erhält, führt er die angeforderte Aufgabe aus und sendet eine Antwort zurück.

Jeder Teilnehmer (Computer) kann prinzipiell mit jedem anderen Computer im Netzwerk kommunizieren.

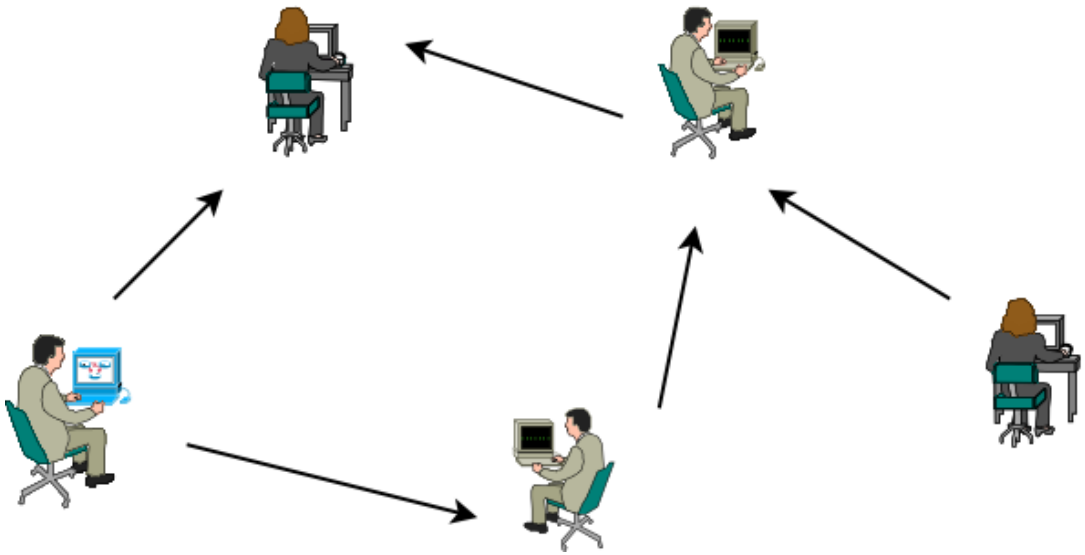
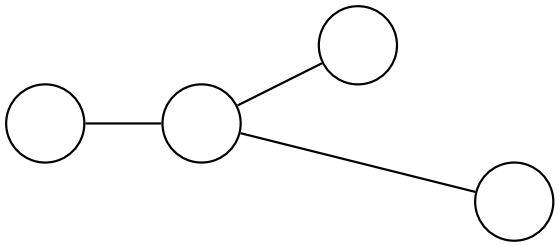
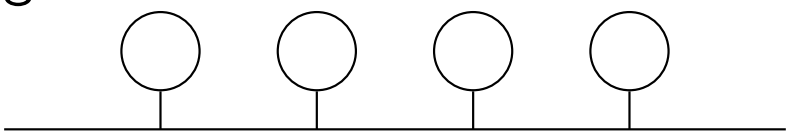


Abbildung 3: Peer-to-Peer-Modell

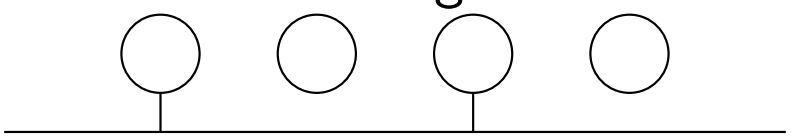
Punkt-zu-Punkt-Verbindungen: Jeweils *zwei* Rechner sind durch einen Kanal verbunden. (*Unicast-Verbindung*)



Broadcast-Verbindungen: *Alle* Rechner im Netz verwenden einen gemeinsamen Kanal.



Multicast-Verbindungen: Eine *Teilmenge* der Rechner im Netz verwenden einen gemeinsamen Kanal.



Ausdehnung	Name (Beispiel)
1 m–10 m	*P*ersonal *A*rea *N*etwork (PC-Arbeitsplatz)
10 m–1 km	*L*ocal *A*rea *N*etwork (Gebäude, Campus)
1 km–10 km	*M*etropolitan *A*rea *N*etwork (Stadt)
10 km–1000 km	*W*ide *A*rea *N*etwork (Land, Kontinent)
1000 km–10 000 km	Internet (Planet)

Verbindungsnetze

In grösseren Netzen sind die angeschlossenen Computer (*Hosts*) über ein *Verbindungsnetz* (*Subnetz*) verbunden.

Das Verbindungsnetz besteht aus *Übertragungskanälen* (Glasfaser, Kupferdrähte, Funk- und Satellitenverbindungen) und *Vermittlungseinheiten* (*Router*)

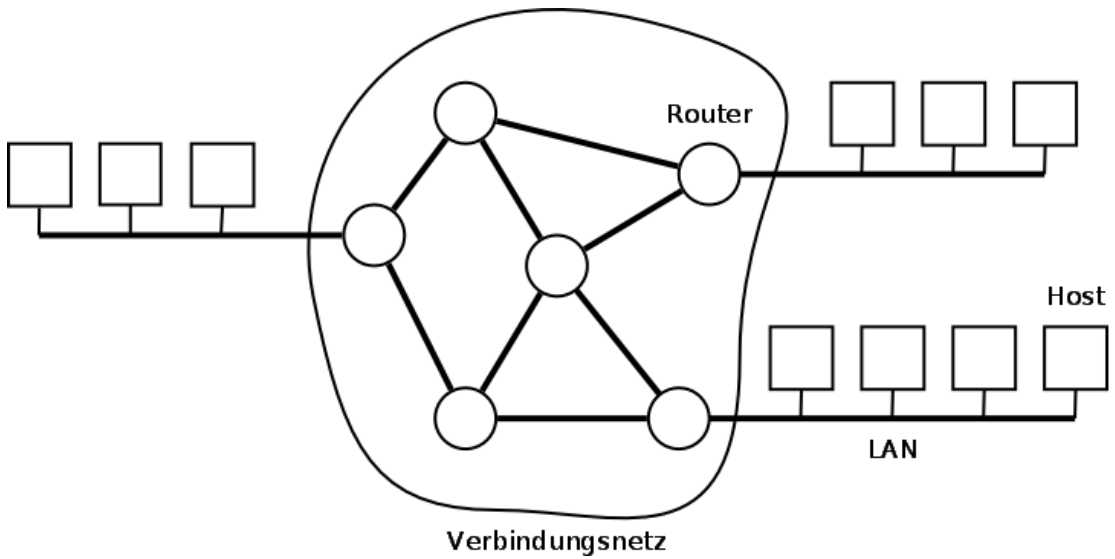


Abbildung 4: Subnetz

Motivation für Schichtenmodelle

Netzwerke sind hochkomplexe Gebilde.

Die Zerlegung der Gesamtaufgabe in unabhängige Teilaufgaben (Schichten) bringt folgende Vorteile:

- Reduktion der Komplexität
- Teile des Systems können durch bessere Technologien ersetzt werden, ohne dass das Gesamtsystem verändert werden muss.

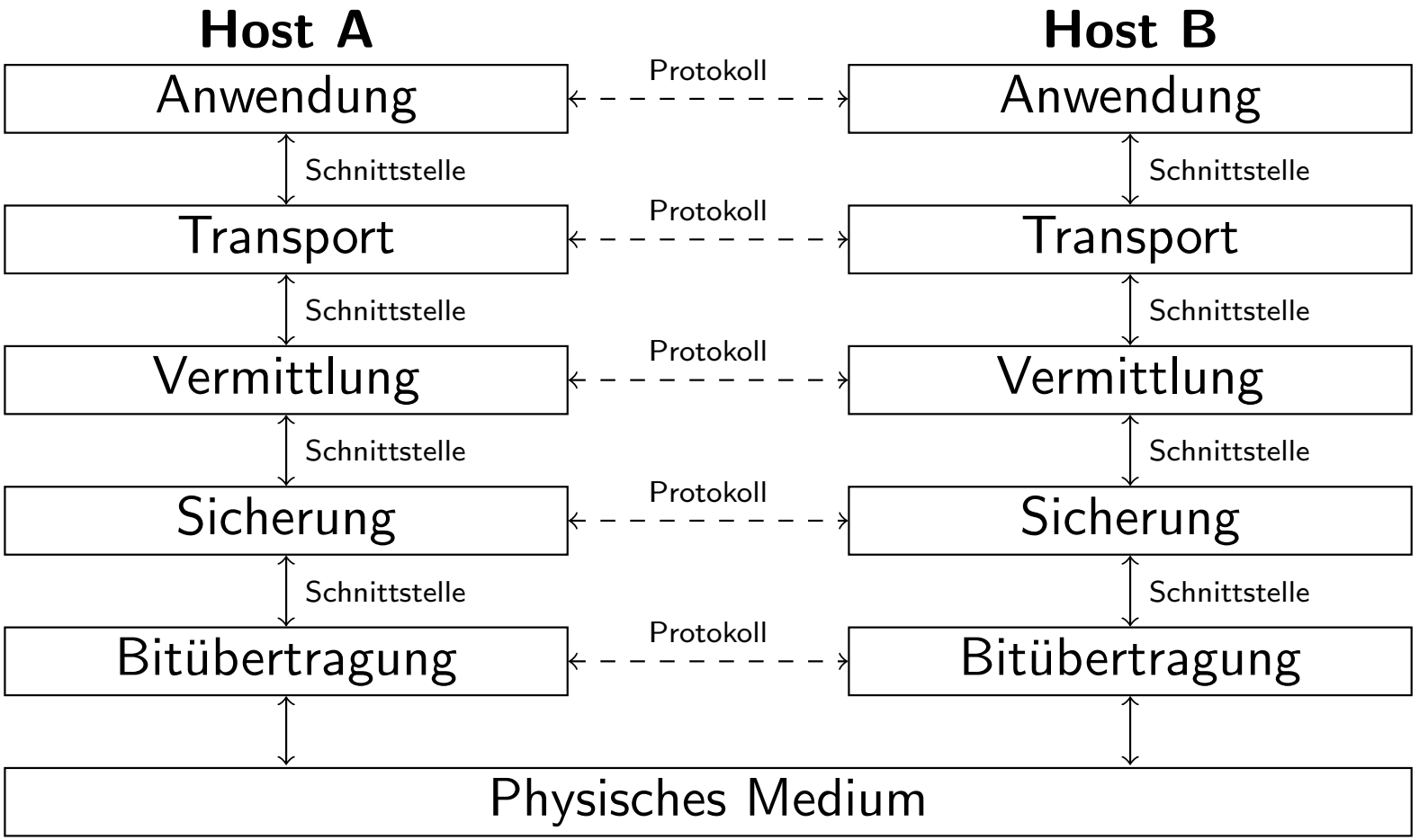
Protokolle, Dienste und Schnittstellen

Ein *Protokoll* ist eine Vereinbarung über den genauen Ablauf der Kommunikation zwischen zwei Kommunikationspartnern.

Ein *Dienst* ist eine Menge von Funktionen, die eine Schicht der darüber liegenden Schicht zur Verfügung stellt.

Eine *Schnittstelle* definiert, *wie* eine höhere Schicht auf die Dienste der darunterliegenden Schicht zugreifen kann.

Das im Unterricht verwendete Schichtenmodell



Die Anwendungsschicht

Anwendungsprozesse

- Mail
- World Wide Web
- Domain Name Service
- ...

Die von der Anwendungsschicht übertragenen Daten werden oft *Nachrichten* genannt.

Die Transportschicht

Die *Transportschicht* zerlegt die Daten in kleinere Einheiten und stellt einen zuverlässigen oder unzuverlässigen Transportdienst von der Quelle zum Ziel zur Verfügung.

Die von der Transportschicht übertragenen Dateneinheiten werden auch *Segmente* genannt.

Vermittlungsschicht

Die *Vermittlungsschicht* sorgt für eine geeignete Wegwahl (Routing) im Subnetz und trifft Massnahmen bei dessen Überlastung.

Die von der Vermittlungsschicht übertragenen Daten werden *Pakete* genannt.

Die Sicherungsschicht

Die *Sicherungsschicht* sorgt für eine möglichst fehlerfreie Übertragung der Daten zwischen zwei Übertragungseinheiten.

Die von der Sicherungsschicht übertragenen Daten werden *Rahmen* genannt.

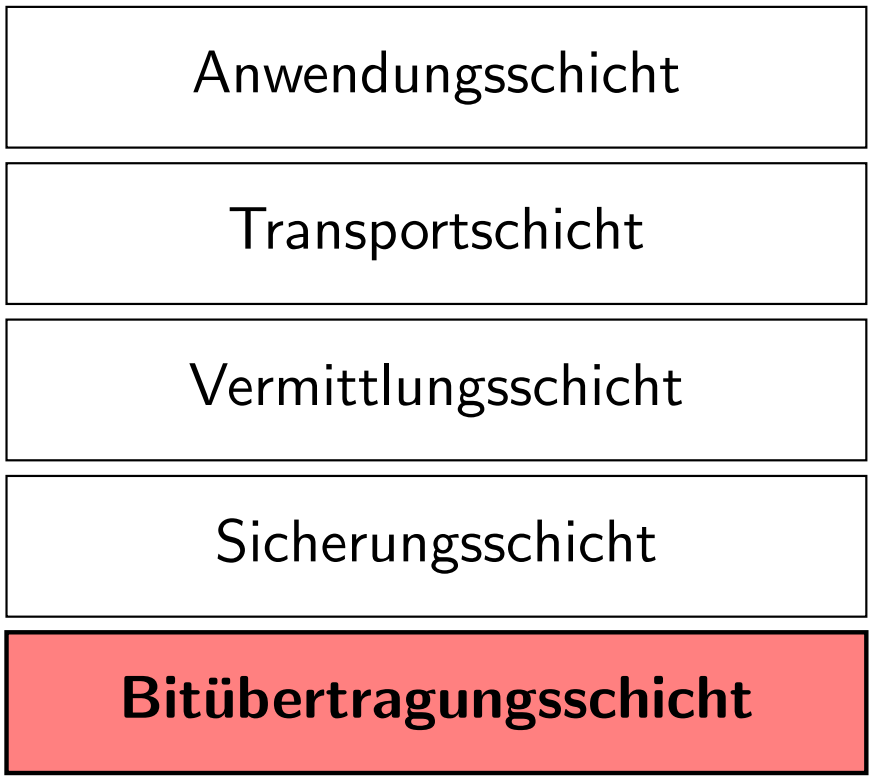
Die Bitübertragungsschicht

Die *Bitübertragungsschicht* sendet die einzelnen Bits über einen Kanal.

Präfix	Abkürzung	Faktor
Nano	n	10^{-9}
Mikro	μ	10^{-6}
Milli	m	10^{-3}
Kilo	k	10^{3}
Mega	M	10^{6}
Giga	G	10^{9}
Tera	T	10^{12}
Peta	P	10^{15}

- b steht für Bit; B für Byte
- Übertragungsraten werden in Bit pro Sekunde angegeben.
- $2^{10} = 1024 \approx 1000 = 10^{3}$

Standortbestimmung



Signalübertragung

Informationen können durch die Veränderung einer physikalischen Grösse (Stromstärke, Spannung, Lichtpulse) übertragen werden. Dies kann durch eine Funktion $y = f(t)$ modelliert werden.

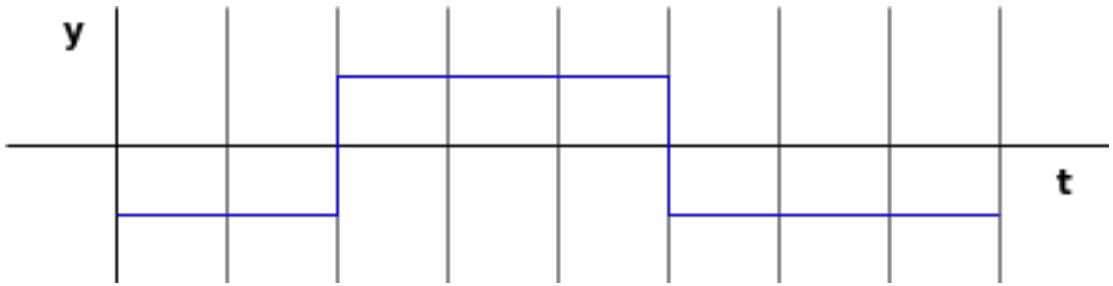


Abbildung 5: Bitmuster 00111000 als $y = f(t)$

Bemerkung: Die senkrechten blauen Linien dürften eigentlich nicht gezeichnet werden.

Fourieranalyse

Der französische Mathematiker *Jean-Baptiste Fourier* (1768–1830) konnte beweisen, dass jede „vernünftige“ periodische Funktion $y = f(t)$ als (möglicherweise unendliche) Summe von Sinus- und Cosinusfunktionen dargestellt werden kann.

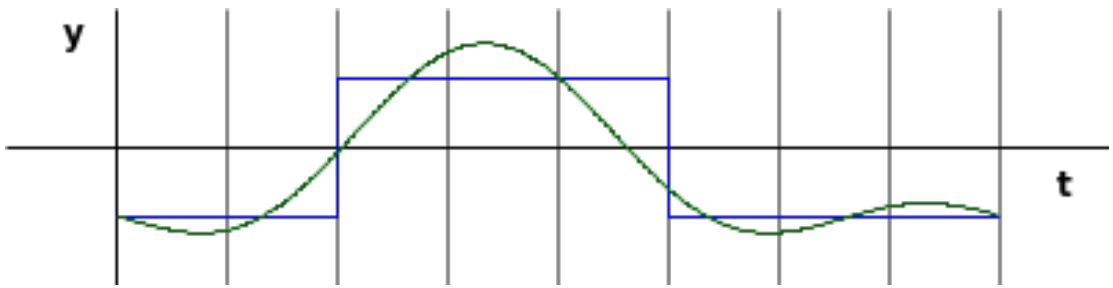


Abbildung 6: $y = f(t)$ und trigonometrische Näherungsfunktion

$$f(t) \approx -0.33 - \cos(t) + 0.58\sin(t) + 0.33\cos(2t) - 0.58\sin(2t)$$

Fortsetzung des Beispiels

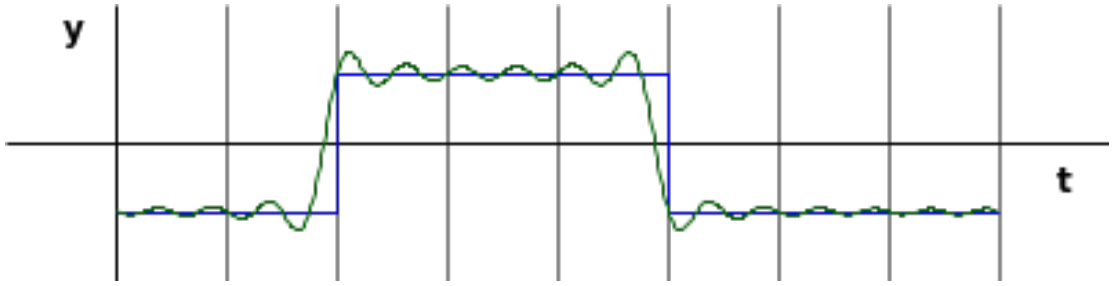


Abbildung 7: Terme bis $\cos(16t)$ bzw. $\sin(16t)$

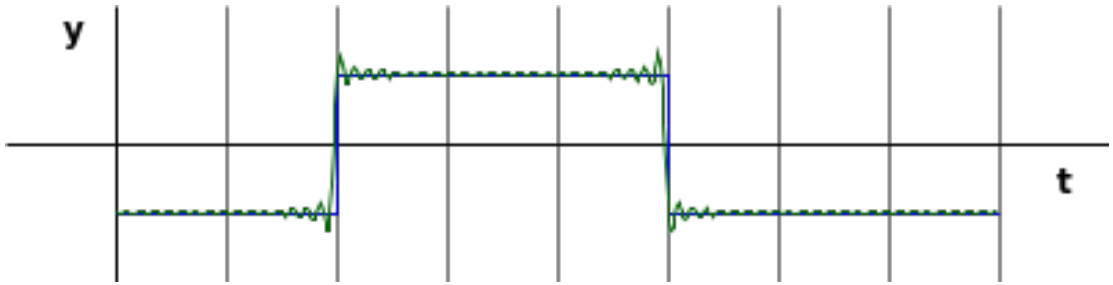


Abbildung 8: Terme bis $\cos(64t)$ bzw. $\sin(64t)$

Das Abtasttheorem von Nyquist und Shannon

Damit ein Signal aus den Abtastwerten rekonstruiert werden kann, muss die Abtastrate (*sampling rate*, f_{samp}) grösser sein als das Doppelte der im Signal auftretenden Maximalfrequenz f_{max}:

$$f_{\text{samp}} > 2 \cdot f_{\text{max}}$$

Besteht ein Signal aus V Stufen, dann gilt:

$$\text{maximale Datenübertragungsrate} = 2B \cdot \log_2 V \quad \text{(in Bit/s)}$$

Beispiel: In den früher üblichen Telefonleitungen wurde die Maximalfrequenz künstlich durch einen Tiefpassfilter auf 3 kHz begrenzt. Damit ein Sprachsignal aus den Abtastwerten wieder rekonstruiert werden kann, muss man es mindestens 6000 Mal pro Sekunde abtasten.

Analoge Bandbreite

Elektroingenieure definieren die (analoge) *Bandbreite* als das Frequenzspektrum

$$B = f_{\text{max}} - f_{\text{min}} \,,$$

in dem Signale ohne grössere Dämpfung übertragen werden. B wird in Schwingungen pro Sekunde gemessen. Die Masseinheit ist Hertz (Hz).

Beispiele:

- analoges Telefonnetz: 3 kHz
- analoge Fernsehkanäle: 6 MHz
- Lichtwellenleiter: 1 GHz ($< 1\,\text{km}$)

Signal-Rausch-Verhältnis

Molekülbewegungen bewirken Rauschen (*noise*) in einem Kommunikationskanal.

Das *Signal-Rausch-Verhältnis* (SNR=*Signal to Noise Ratio*) ist das Verhältnis der mittleren Nutz- und Rauschsignalleistung S/N.

Da die Nutzleistung oft sehr viel grösser als die Rauschleistung ist, wird S/N logarithmisch dargestellt:

$$\text{SNR} = 10 \cdot \log_{10}(S/N) \quad (\text{in dB} = \textit{Dezibel})$$

Beispiele:

- Gib $S/N = 1000$ in Dezibel an:

Signal-Rausch-Verhältnis

Molekülbewegungen bewirken Rauschen (*noise*) in einem Kommunikationskanal.

Das *Signal-Rausch-Verhältnis* (SNR=*Signal to Noise Ratio*) ist das Verhältnis der mittleren Nutz- und Rauschsignalleistung S/N.

Da die Nutzleistung oft sehr viel grösser als die Rauschleistung ist, wird S/N logarithmisch dargestellt:

$$\text{SNR} = 10 \cdot \log_{10}(S/N) \quad (\text{in dB} = \textit{Dezibel})$$

Beispiele:

- Gib $S/N = 1000$ in Dezibel an: $\text{SNR} = 30\,\text{dB}$

Signal-Rausch-Verhältnis

Molekülbewegungen bewirken Rauschen (*noise*) in einem Kommunikationskanal.

Das *Signal-Rausch-Verhältnis* (SNR=*Signal to Noise Ratio*) ist das Verhältnis der mittleren Nutz- und Rauschsignalleistung S/N.

Da die Nutzleistung oft sehr viel grösser als die Rauschleistung ist, wird S/N logarithmisch dargestellt:

$$\text{SNR} = 10 \cdot \log_{10}(S/N) \quad (\text{in dB} = \textit{Dezibel})$$

Beispiele:

- Gib $S/N = 1000$ in Dezibel an: $\text{SNR} = 30\,\text{dB}$
- Wie gross ist S/N für $\text{SNR} = -20\,\text{dB}$?

Signal-Rausch-Verhältnis

Molekülbewegungen bewirken Rauschen (*noise*) in einem Kommunikationskanal.

Das *Signal-Rausch-Verhältnis* (SNR=*Signal to Noise Ratio*) ist das Verhältnis der mittleren Nutz- und Rauschsignalleistung S/N.

Da die Nutzleistung oft sehr viel grösser als die Rauschleistung ist, wird S/N logarithmisch dargestellt:

$$\text{SNR} = 10 \cdot \log_{10}(S/N) \quad (\text{in dB} = \textit{Dezibel})$$

Beispiele:

- Gib $S/N = 1000$ in Dezibel an: $\text{SNR} = 30\,\text{dB}$
- Wie gross ist S/N für $\text{SNR} = -20\,\text{dB}$? $S/N = 0.01$

Digitale Bandbreite

Informatiker definieren die den Begriff der (digitalen) *Bandbreite* als die maximale Datenrate eines Kanals. Die (digitale) Bandbreite wird in Bit/s gemessen.

Das Gesetz von Shannon

Claude Shannon veröffentlichte 1948 einen Arbeit, in der er eine Obergrenze für die maximale Datenübertragungsrate eines Kanals mit einer Bandbreite B und einem Signal-Rausch-Verhältnis von S/N angibt:

$$\text{Maximale Anzahl Bit/Sekunde} = B \cdot \log_2(1 + S/N)$$

Beispiel: Welche maximale Datenübertragungsrate lässt sich auf einer Telefonverbindung ($B = 3\,\text{kHz}$) mit einem SNR von 20 dB erreichen?

Das Gesetz von Shannon

Claude Shannon veröffentlichte 1948 einen Arbeit, in der er eine Obergrenze für die maximale Datenübertragungsrate eines Kanals mit einer Bandbreite B und einem Signal-Rausch-Verhältnis von S/N angibt:

$$\text{Maximale Anzahl Bit/Sekunde} = B \cdot \log_2(1 + S/N)$$

Beispiel: Welche maximale Datenübertragungsrate lässt sich auf einer Telefonverbindung ($B = 3\,\text{kHz}$) mit einem SNR von 20 dB erreichen?

$\text{SNR} = 20\,\text{dB} \Rightarrow S/N = 100$

Das Gesetz von Shannon

Claude Shannon veröffentlichte 1948 einen Arbeit, in der er eine Obergrenze für die maximale Datenübertragungsrate eines Kanals mit einer Bandbreite B und einem Signal-Rausch-Verhältnis von S/N angibt:

$$\text{Maximale Anzahl Bit/Sekunde} = B \cdot \log_2(1 + S/N)$$

Beispiel: Welche maximale Datenübertragungsrate lässt sich auf einer Telefonverbindung ($B = 3\,\text{kHz}$) mit einem SNR von 20 dB erreichen?

$\text{SNR} = 20\,\text{dB} \Rightarrow S/N = 100$

$3000 \cdot \log_2(1 + 100) \approx 20\,000\,\text{Bit/s}$

Typen von Punkt-zu-Punkt-Verbindungen

- *Vollduplexverbindung:* Die Daten können in beide Richtungen übertragen werden. („zweispurige Strasse“)

  $A \rightleftharpoons B$

- *Halbduplexverbindung:* Die Daten können in beide Richtungen übertragen werden aber bei jeder Übertragung immer nur in einer Richtung. („enge einspurige Strasse“)

  $A \longrightarrow B \ (t = 0\,s)$
  $A \longleftarrow B \ (t = 1\,s)$
  $A \longrightarrow B \ (t = 2\,s)$

- *Simplexverbindung:* Die Daten können nur in einer Richtung übertragen werden. („Einbahnstrasse“)

  $A \longrightarrow B$

Twisted-Pair-Kabel

Zwei in der Regel 1 mm dicke isolierte Kupferkabel werden miteinander verdrillt (*Twisted-Pair-Kabel*).

Parallele Kabel bilden eine gute Antenne. Sie senden und empfangen unerwünschte Signale. Durch das Verdrillen der Kabel heben sich die störenden Einflüsse grösstenteils auf.

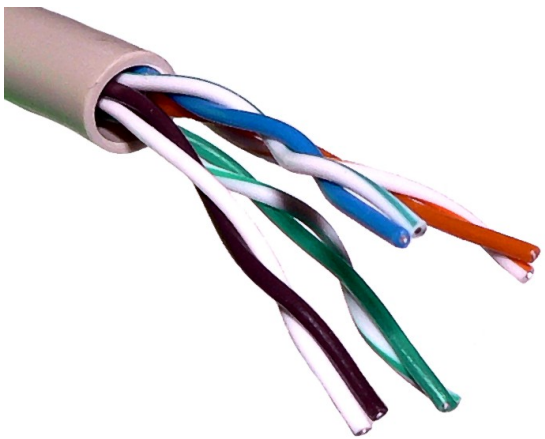


Abbildung 9: Twisted-Pair-Kabel (ohne Abschirmung)

Koaxialkabel

Ein Kupferkern wird von einer Isolationsschicht umgeben, die wiederum von einem Aussenleiter umgeben wird. Das Kabel wird von einem Schutzmantel umgeben.

Moderne Koaxialkabel haben eine Bandbreite von mehreren GHz und werden häufig in Stadtnetzen (MANs) eingesetzt.

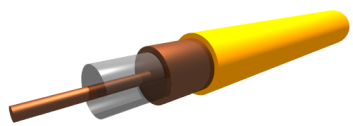


Abbildung 10: Koaxialkabel

Trägerfrequenzanlagen

Auf den niederfrequenten Wechselstrom (50 Hz) in elektrischen Leitungen wird ein hochfrequentes Datensignal aufgesetzt.

Probleme:

- Störungen durch Ein- und Ausschalten elektrischer Geräte
- Dämpfung hoher Frequenzen in Stromkabeln
- Ohne Verdrillung verhalten sich die Kabel wie Antennen

Dennoch sind Datenraten bis zu mehreren 100 MBit/s möglich.

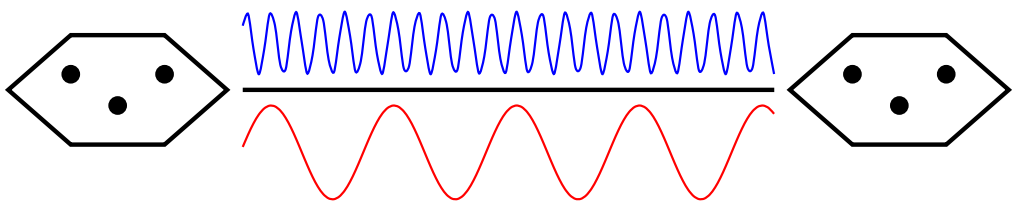


Abbildung 11: schematische Darstellung einer Trägerfrequenzanlage

Glasfaserkabel

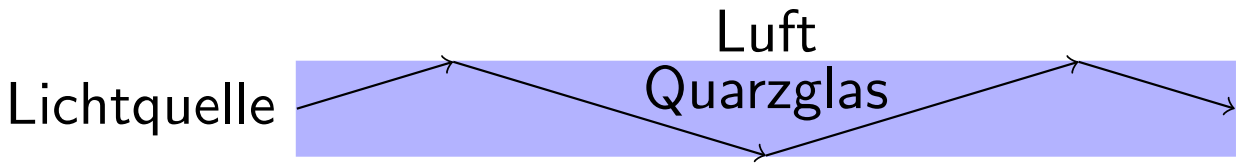


Abbildung 12: Totalreflexion

Ein Glaskern wird von einem Glasmantel mit einem niedrigeren Brechungsindex umgeben. Damit bleibt das Licht im Glaskern.

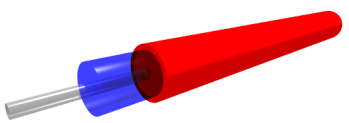


Abbildung 13: Glasfaser

Mikrowellenübertragung

Ab Frequenzen von 1 MHz wird es schwierig, elektromagnetische Signale mittels Twisted-Pair- oder Koaxialkabel zu übertragen.

Mikrowellen sind elektromagnetische Wellen im Frequenzbereich von 1 GHz bis 300 GHz.

Mikrowellen verlaufen in einer geraden Linie.

Im Gegensatz zu Funkwellen (10 kHz bis 1 GHz) können Mikrowellen keine Gebäude durchdringen.

Regen dämpft die Übertragungsleistung.

Einsatzgebiete für Mikrowellenübertragung

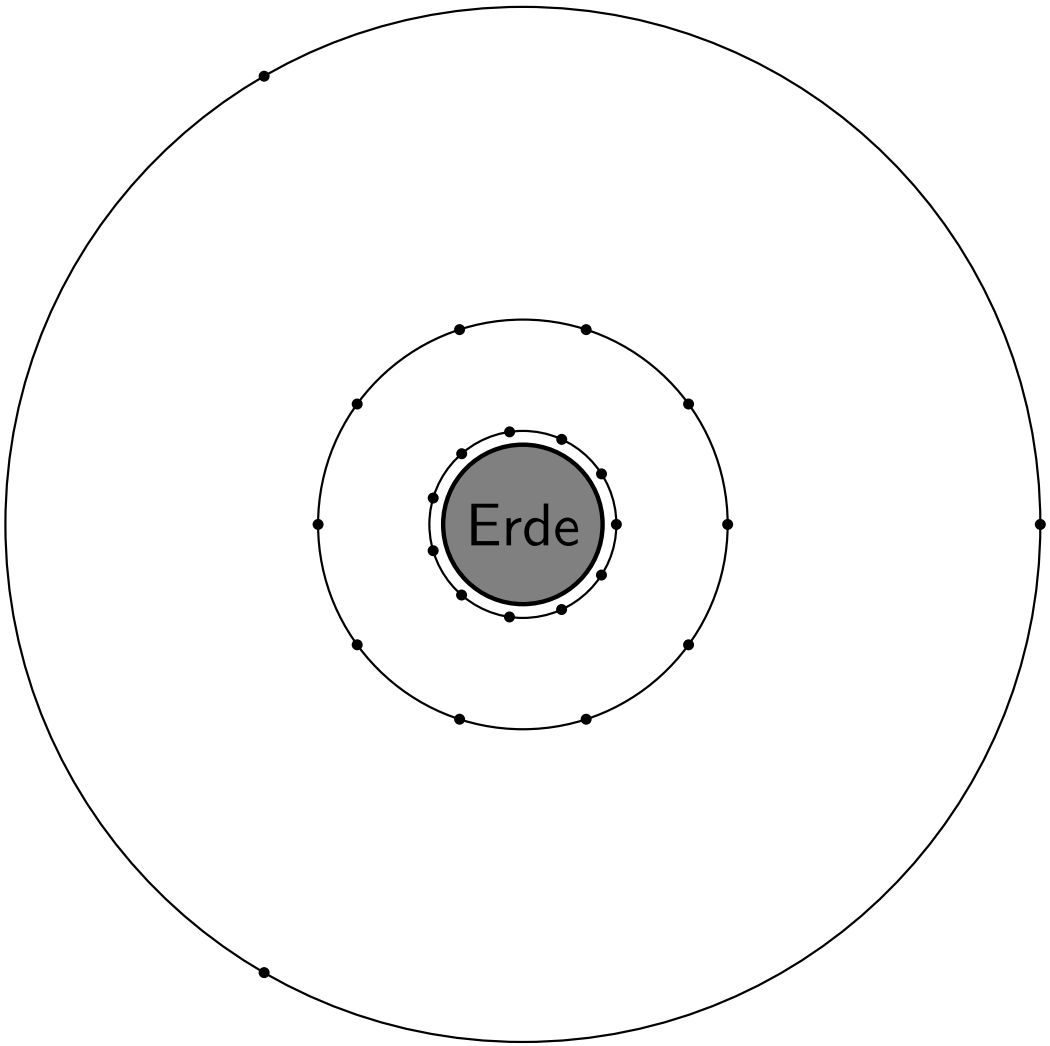
- Mobilfunk
- Amateurfunk
- Richtfunk
- Bluetooth
- WLAN
- Satellitenempfang
- GPS
- Funketiketten (RFID)
- Funkthermometer
- Funk-Kopfhörer
- Mikrowellenherde(!)

Kommunikationssatelliten

Klassifikation aufgrund der Höhe h:

- *GEO-Satelliten* (Geostationary Earth Orbit):
  $h \approx 35\,000\,\text{km}$
- *MEO-Satelliten* (Medium Earth Orbit):
  $5\,000\,\text{km} < h < 15\,000\,\text{km}$
- *LEO-Satelliten* (Low Earth Orbit):
  $200\,\text{km} < h < 2\,000\,\text{km}$

Abstandsverhältnisse (Massstab 1:12 500)



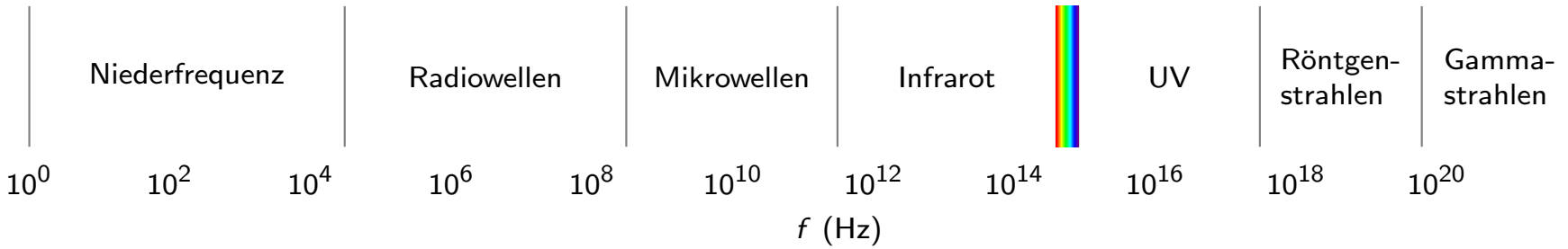
Latenz bei Satellitenverbindungen

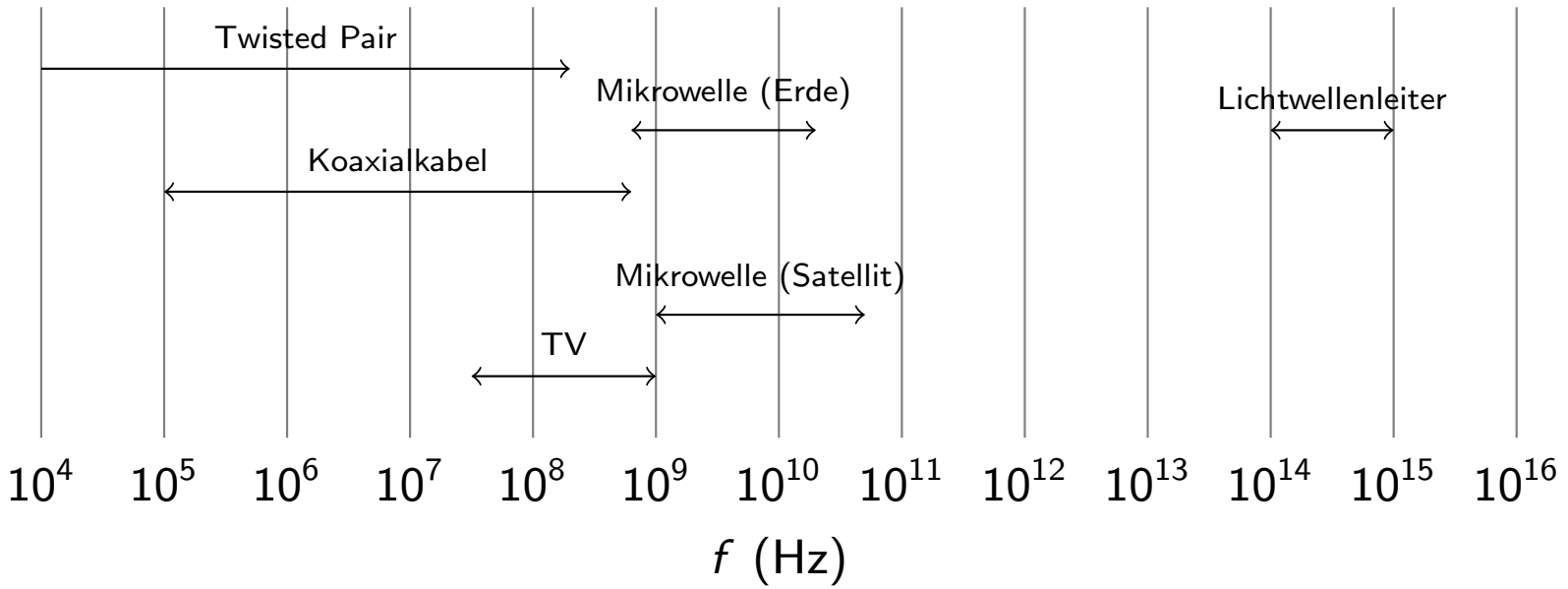
Aufgrund der grossen Entfernungen zwischen Bodenstationen und Satelliten gewinnt die Signallaufzeit an Bedeutung.

Die *Latenzzeit* ist die Zeit, die ein Signal braucht, um vom Sender zum Satellit und von dort zum Empfänger zu gelangen.

Typ	Latenzzeit (in ms)
GEO	250–300
MEO	35–80
LEO	1–7

Das elektromagnetische Spektrum





Modulation

Modulation: Ein Nutzsignal (Daten, Sprache) wird so transformiert, dass es von einem Träger(signal) übertragen werden kann.

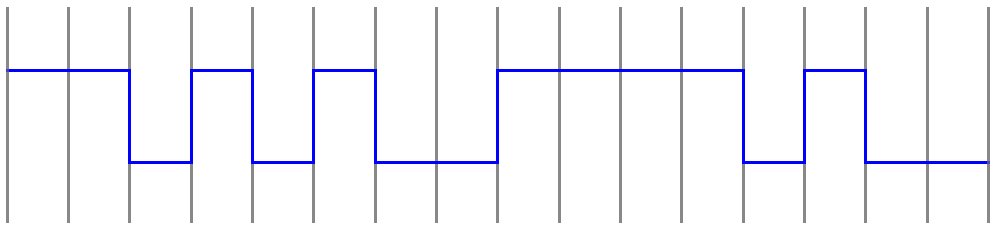
Modulation

Modulation: Ein Nutzsignal (Daten, Sprache) wird so transformiert, dass es von einem Träger(signal) übertragen werden kann.

Beispiel: Morsezeichen (Punkt, Strich) werden durch kurze und lange Lichtimpulse einer Taschenlampe übertragen.

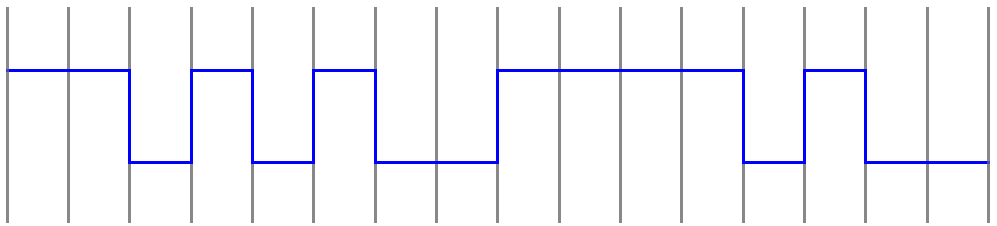
Non-Return-to-Zero-Codierung (NRZ)

Bitfolge: 1101010011110100



Non-Return-to-Zero-Codierung (NRZ)

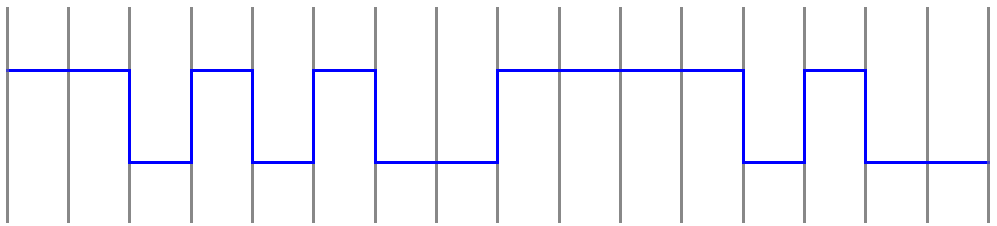
Bitfolge: 1101010011110100



Vorteil:

Non-Return-to-Zero-Codierung (NRZ)

Bitfolge: 1101010011110100

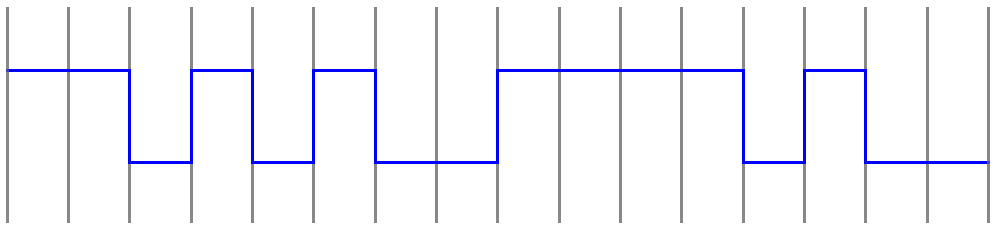


Vorteil:

einfache Realisierung (z. B. $0 \Rightarrow -1\,\mathrm{V}$, $1 \Rightarrow +1\,\mathrm{V}$)

Non-Return-to-Zero-Codierung (NRZ)

Bitfolge: 1101010011110100



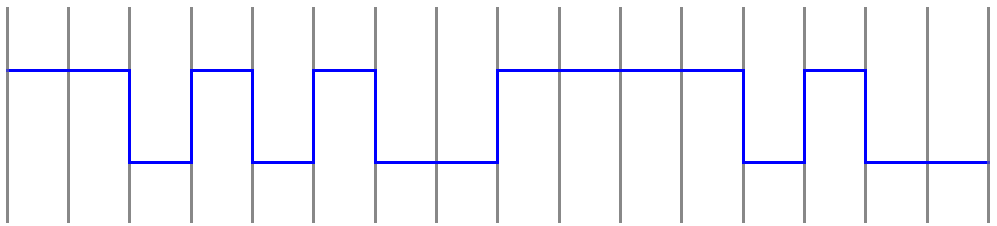
Vorteil:

einfache Realisierung (z. B. $0 \Rightarrow -1\,\mathrm{V}$, $1 \Rightarrow +1\,\mathrm{V}$)

Nachteil:

Non-Return-to-Zero-Codierung (NRZ)

Bitfolge: 1101010011110100



Vorteil:

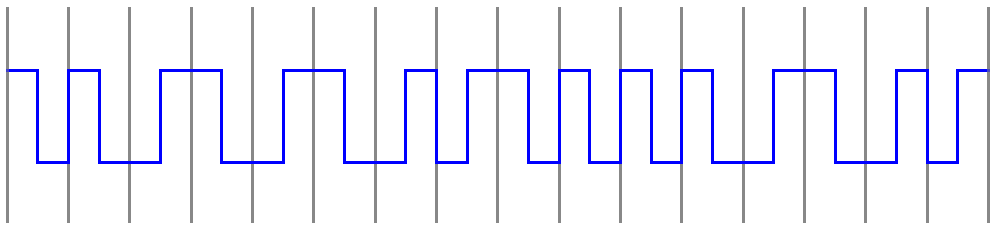
einfache Realisierung (z. B. $0 \Rightarrow -1\,\mathrm{V}$, $1 \Rightarrow +1\,\mathrm{V}$)

Nachteil:

Synchronisationsprobleme bei langen Folgen gleicher Bits

Manchester-Codierung

Bitfolge: 1101010011110100



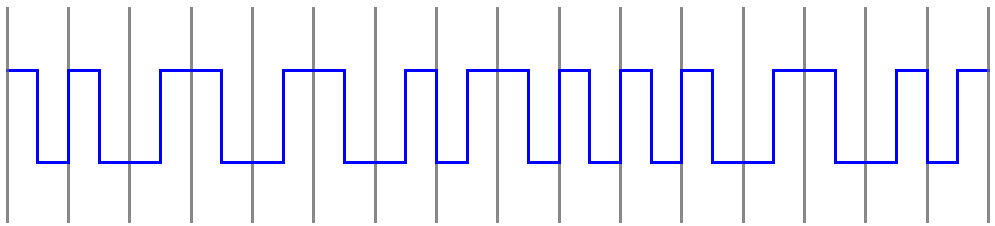
0: steigende Flanke

1: fallende Flanke

(es gibt auch die umgekehrte Version)

Manchester-Codierung

Bitfolge: 1101010011110100



0: steigende Flanke

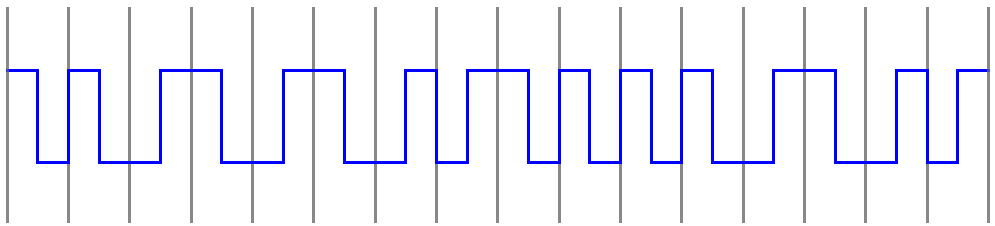
1: fallende Flanke

(es gibt auch die umgekehrte Version)

Vorteil:

Manchester-Codierung

Bitfolge: 1101010011110100



0: steigende Flanke

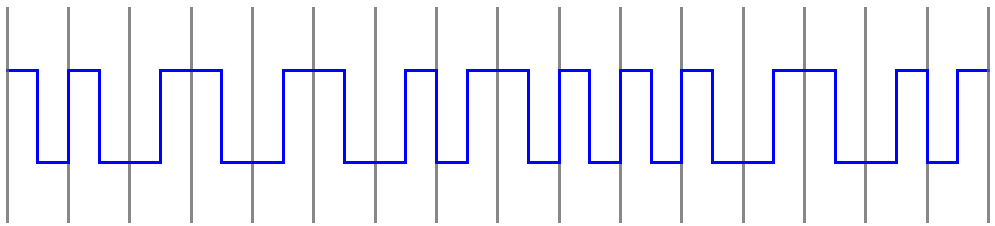
1: fallende Flanke

(es gibt auch die umgekehrte Version)

Vorteil: einfachere Synchronisation

Manchester-Codierung

Bitfolge: 1101010011110100



0: steigende Flanke

1: fallende Flanke

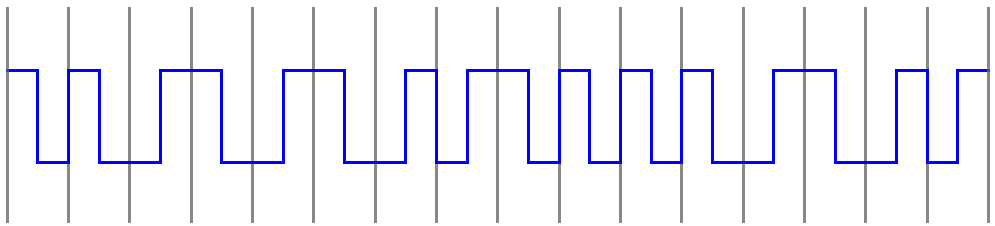
(es gibt auch die umgekehrte Version)

Vorteil: einfachere Synchronisation

Nachteil:

Manchester-Codierung

Bitfolge: 1101010011110100



0: steigende Flanke

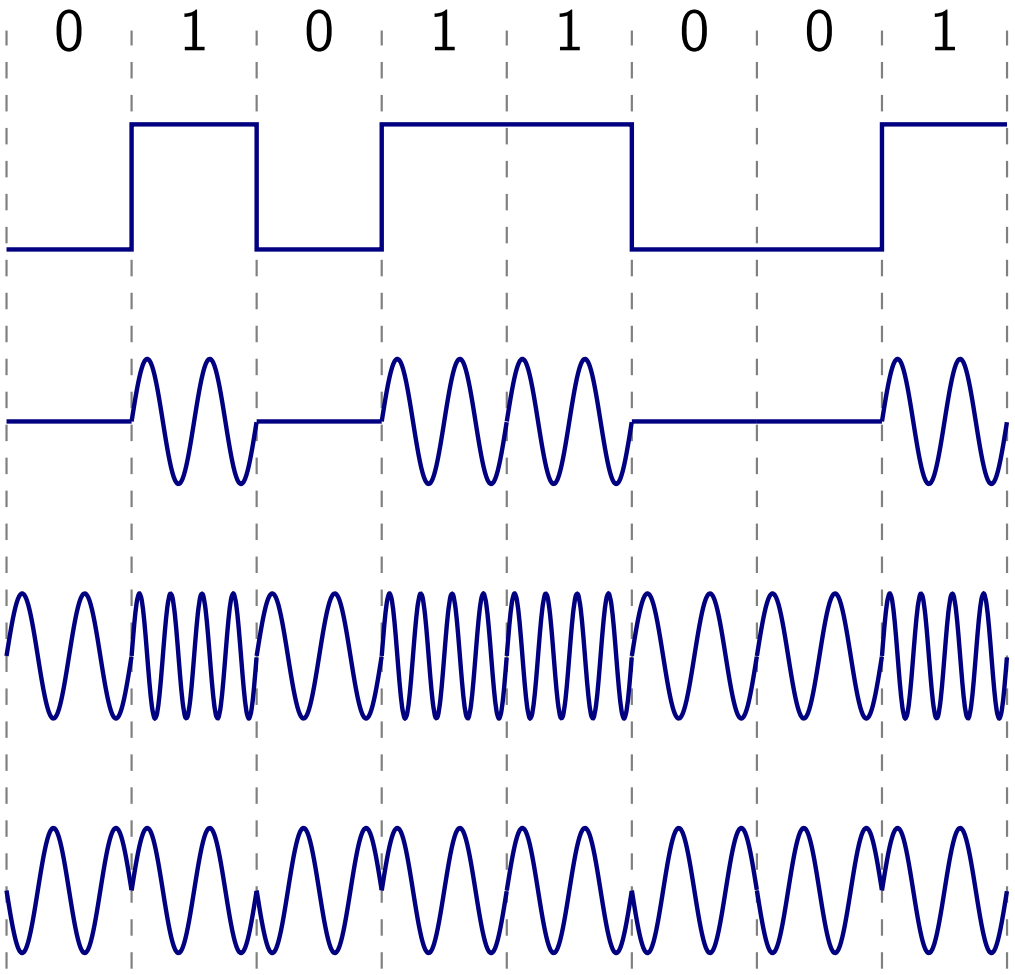
1: fallende Flanke

(es gibt auch die umgekehrte Version)

Vorteil: einfachere Synchronisation

Nachteil: benötigt doppelte Frequenz bzw. Bandbreite

Digitale Modulationsverfahren für ein Trägersignal



Binärsignal

Amplitudenmodulation

Frequenzmodulation

Phasenmodulation

Der Grundgedanke

Multiplexing: Verfahren zur Bündelung mehrerer Signale, um sie gemeinsam über ein Medium zu übertragen.

Demultiplexing: Das Trennen der gebündelten Signale beim Empfänger.

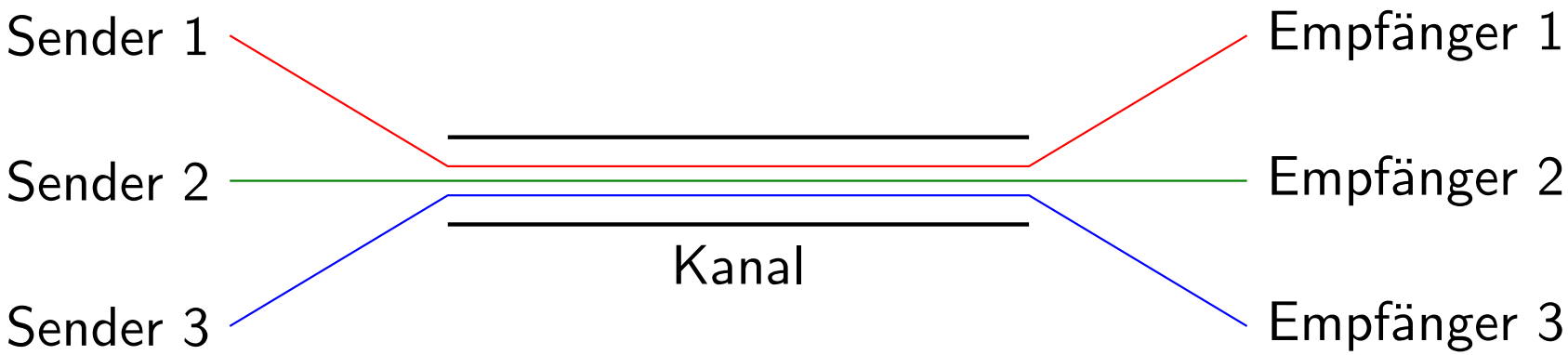


Abbildung 14: Multiplexing und Demultiplexing

Zeitmultiplexverfahren

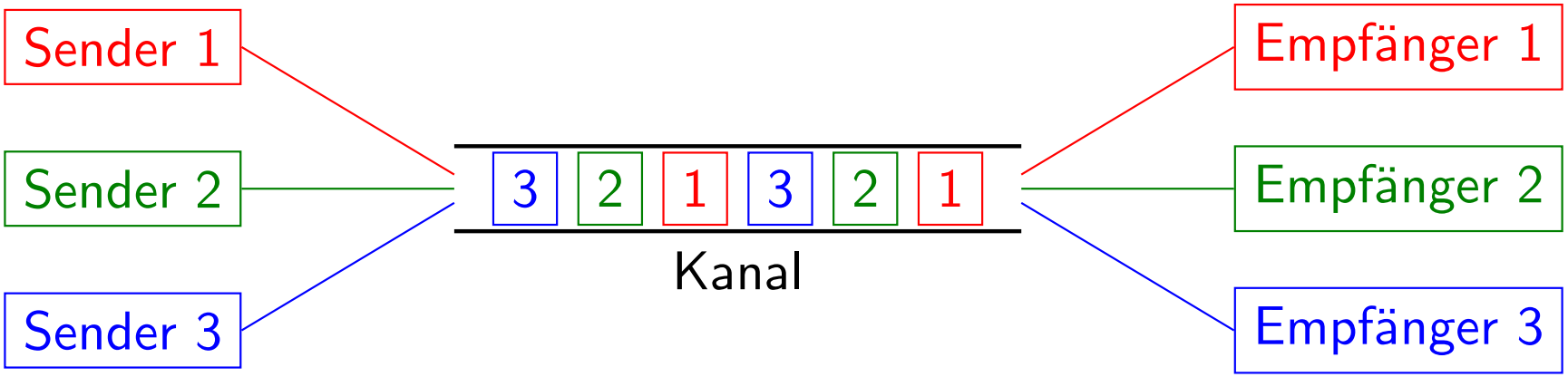


Abbildung 15: Zeitmultiplexing

Frequenzmultiplexverfahren

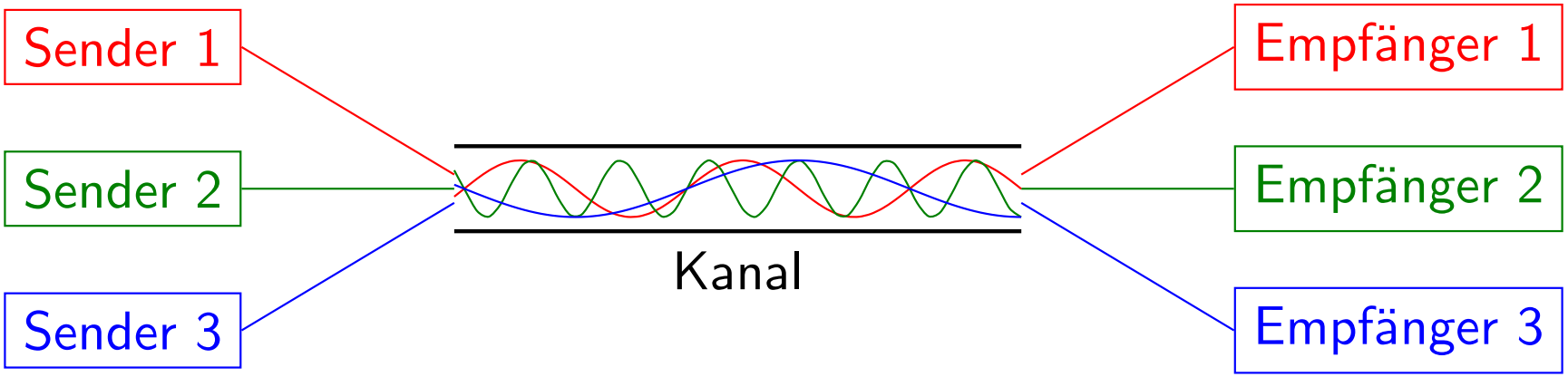
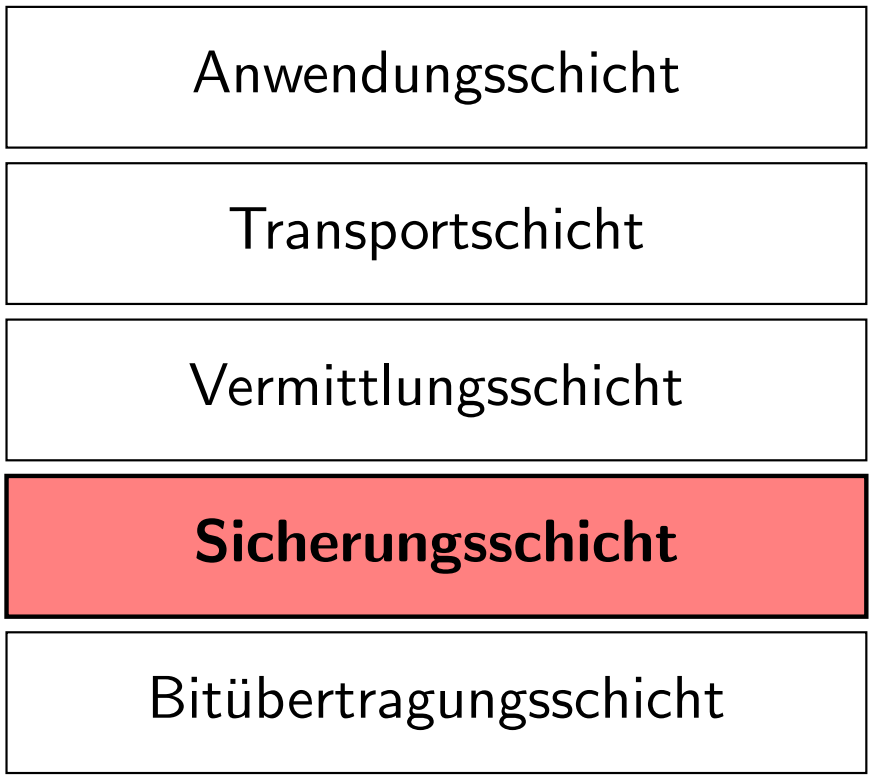


Abbildung 16: Frequenzmultiplexing

Standortbestimmung



Die Aufgaben der Sicherungsschicht

1. Dienstleister für die Vermittlungsschicht
2. Behandlung von Übertragungsfehlern
3. Regulierung des Datenflusses

Pakete und Rahmen

Rahmen kapseln die Pakete der Vermittlungsschicht:

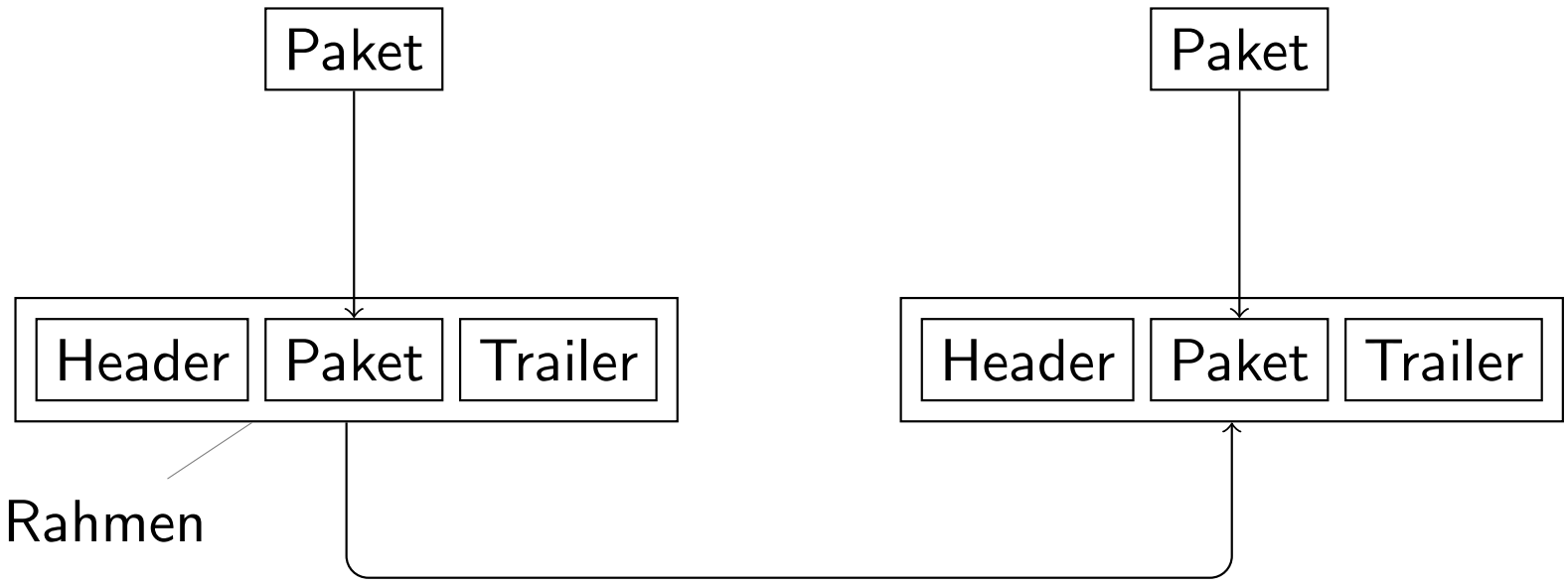


Abbildung 17: Rahmenbildung

Byte-Stuffing

Die Rahmen werden durch ein spezielles Byte (*Flagbyte*) gekennzeichnet.

Abbildung 18: Flagbytes begrenzen Rahmen

Problem: Flagbytes *in* den Nutzdaten

Byte-Stuffing

Die Rahmen werden durch ein spezielles Byte (*Flagbyte*) gekennzeichnet.

Abbildung 18: Flagbytes begrenzen Rahmen

Problem: Flagbytes *in* den Nutzdaten

Lösung: Vor jedes Flagbyte in den Nutzdaten setzt man ein *Escape-Byte* (*Byte-Stuffing*).

Byte-Stuffing

Die Rahmen werden durch ein spezielles Byte (*Flagbyte*) gekennzeichnet.

Abbildung 18: Flagbytes begrenzen Rahmen

Problem: Flagbytes *in* den Nutzdaten

Lösung: Vor jedes Flagbyte in den Nutzdaten setzt man ein *Escape-Byte* (*Byte-Stuffing*).

nächstes Problem: Escape-Bytes in den Nutzdaten

Byte-Stuffing

Die Rahmen werden durch ein spezielles Byte (*Flagbyte*) gekennzeichnet.

Abbildung 18: Flagbytes begrenzen Rahmen

Problem: Flagbytes *in* den Nutzdaten

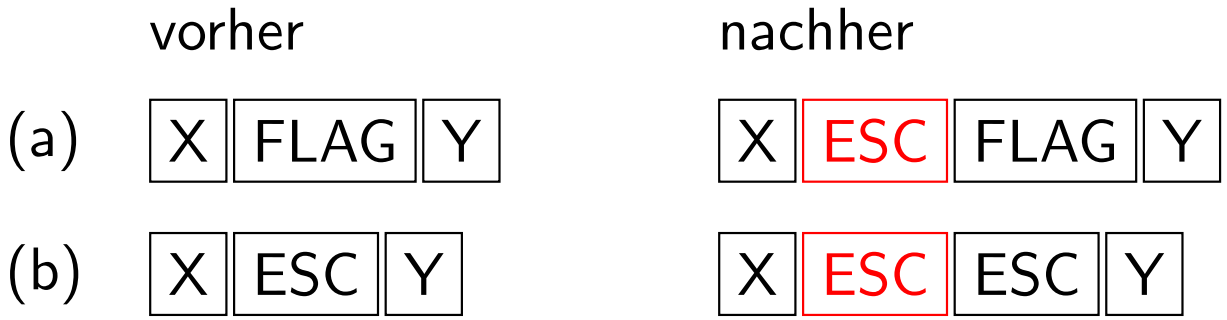
Lösung: Vor jedes Flagbyte in den Nutzdaten setzt man ein *Escape-Byte* (*Byte-Stuffing*).

nächstes Problem: Escape-Bytes in den Nutzdaten

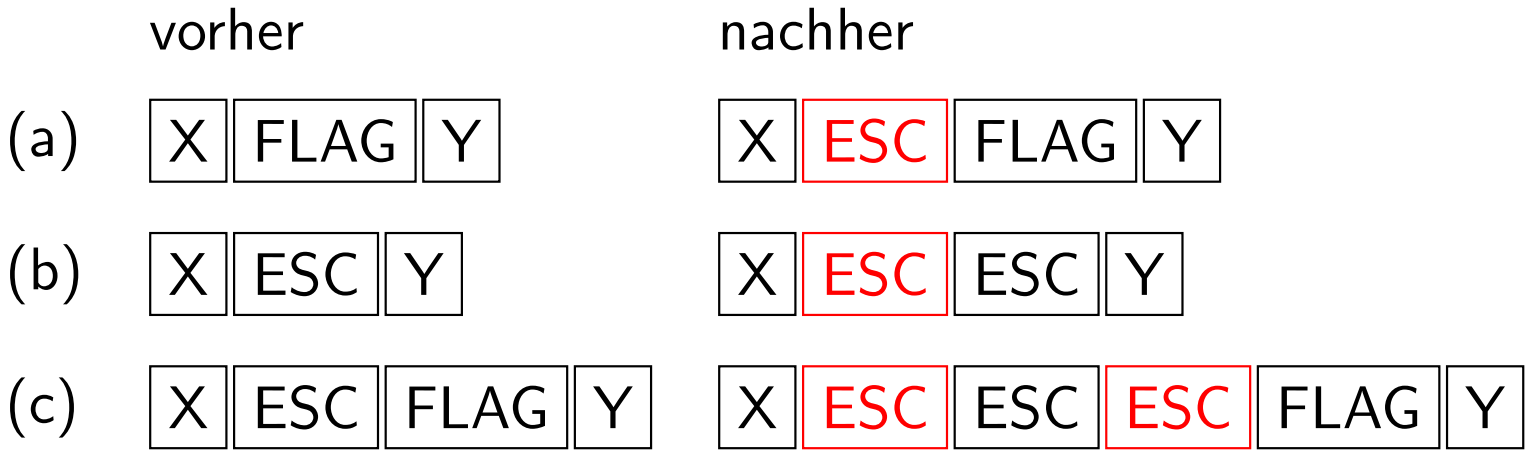
Lösung: Vor jedes Escape-Byte in den Nutzdaten setzt man ein weiteres Escape-Byte.

Beispiele zum Byte-Stuffing

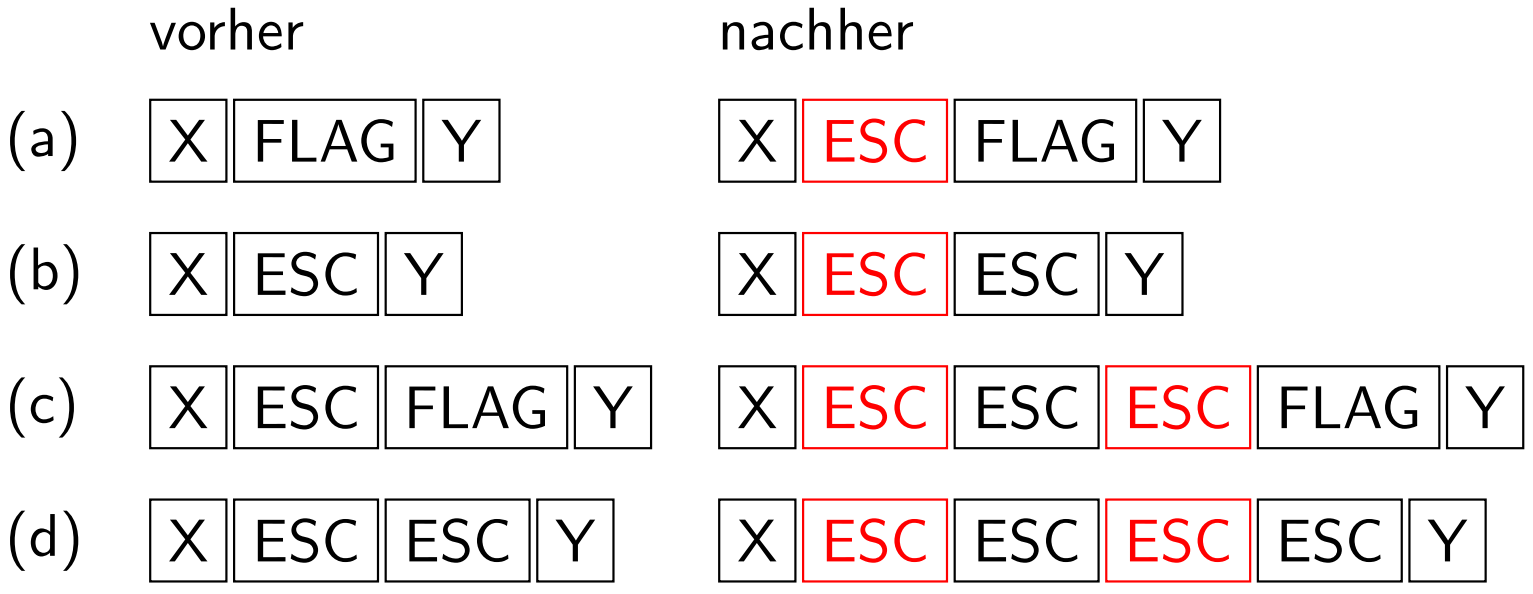
Beispiele zum Byte-Stuffing



Beispiele zum Byte-Stuffing



Beispiele zum Byte-Stuffing



Bit-Stuffing

Es gibt Bitströme, deren Länge kein Vielfaches von 8 ist.

In dieser Situation kann ein spezielles Bitmuster (`01111110`) zur Rahmenbegrenzung eingesetzt werden.

Um zu verhindern, dass dieses Muster (6 Einsen in Folge) in den Nutzdaten auftritt, wird nach jeder Folge von 5 Einsen ein Nullbit eingefügt.

Beispiel:

`011111111011111100011110` wird zu

`011111011101111100011110`

Der Empfänger muss nur das Nullbit nach jeder Folge von 5 Einsen wieder entfernen, um die Originaldaten wiederherzustellen.

Modell eines Informationsübertragungssystems

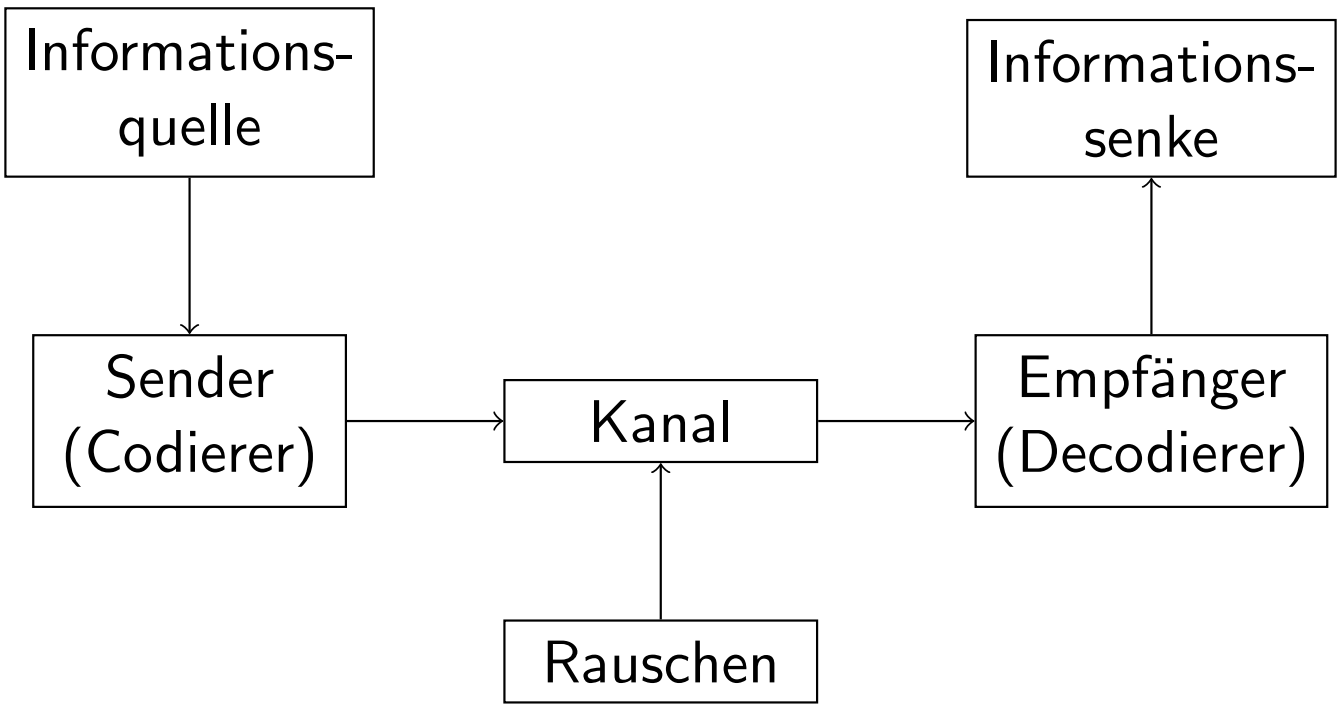


Abbildung 19: Modell der Informationsübertragung

Codes

Wort w: Folge von Binärziffern $w = 10011$

Länge $|w|$ eines Worts w: Anzahl der Ziffern von w $|101| = 3$

Code C: Menge von Wörtern $C = \{0, 10, 110, 1110\}$

Blockcode: Code mit Wörtern gleicher Länge $C = \{01, 10, 11\}$

Distanz d zweier Codewörter: Anzahl der Positionen, in denen sich zwei Wörter gleicher Länge unterscheiden. $d(0110, 0011) = 2$

Distanz eines Blockcodes C: kleinstmögliche Distanz von zwei Codewörtern in C.

Der Code $C = \{1000, 0111, 1010\}$ hat die Distanz 1

Redundanz

Redundanz beschreibt das Vorhandensein von Daten ohne Informationsgehalt.

Aus dem folgenden Text wurden zufällig 20% der Zeichen entfernt. Dennoch ist er einigermassen verständlich. Darüber hinaus sollte die „Geschichte“ bekannt sein.

E war einmal ein kleines Mädche, dasmer ein rotsKppche trg. Daumhiess es be allen eutn nr ”Rotkppche.Eins Tage sgteie Mutter u Knd: ”Hie ist uchenund ne Flace Wein,bringe sie er kraken Grssmutter Aber geh nct om g ab!” Di Grossmutr wohntn einemHäschen i Wad. Rotkäppce ging fort, ud al esduc eWldgng, bentees dem Wolf.Roäppchusse nich, ass der Wolf böe war, und erzählte ih on der krknGrossmuter

Verfahren

Fügt man einem Code auf geeignete Weise Redundanz hinzu, lassen sich

(a) Fehler erkennen

(b) Fehler korrigieren

Das gewählte Verfahren hängt vom Einsatzgebiet ab:

- sehr zuverlässige Kanäle: Fehlererkennung (Glasfaser)
- stark verrauschte Kanäle: Fehlerkorrektur (WLAN)
- hohe Latenz: Fehlerkorrektur (Satellit)

Paritätsbit

Einem Codewort wird ein weiteres Bit hinzugefügt. Dieses Bit wird so gewählt, dass das neue Codewort eine gerade Anzahl Einsen hat.

Beispiele:

- $1101 \Rightarrow 11011$
- $1010 \Rightarrow 10100$

4 Infobits + 1 Kontrollbit $\rightarrow (5, 4)$-Code (Coderate: 80%)

Falls man eine Bitfolge mit einer ungeraden Anzahl Einsen erhält, *muss* bei der Übertragung ein Fehler aufgetreten sein.

Andererseits ist eine gerade Anzahl Einsen keine Garantie für Fehlerfreiheit, denn mit dem Paritätsbit lassen sich z. B. weder zwei noch vier Bitfehler erkennen. Bei drei oder fünf Fehlern kommt es zudem noch darauf an, ob das Paritätsbit betroffen ist.

Wie viele Fehler kann ein Code erkennen?

Der Code $C = \{000, 111\}$ hat offensichtlich den Abstand 3.

Ausgehend vom Wort 000 können wir bis zu zwei Bitfehler erkennen. Bei drei Bitfehlern entsteht das andere gültige Codewort, so dass ein allfälliger Fehler nicht mehr erkennbar wäre. Dasselbe gilt für 111.

Allgemein: Wenn ein Code mindestens d Bitfehler *erkennen* soll, muss er den Abstand $d + 1$ haben.

Beachte: Da der Abstand $d + 1$ eines Codes ein Minimalabstand ist, kann es auch einzelne Codewörter geben bei denen sich mehr als d Fehler erkennen lassen. Diese höhere Erkennungsleistung kann aber nicht für alle Elemente garantiert werden.

Polynome mit Binärkoeffizienten

Ein Polynom in den Koeffizienten a_n, a_{n-1}, ..., a_1, a_0 ist ein Ausdruck der folgenden Form:

$$a_n x^n + a_{n-1} x^{n-1} + \ldots a_1 x + a_0$$

Stammen die Koeffizienten aus dem endlichen Körper $\mathbb{F}_2 = \{0, 1\}$ mit den Operationen

$+$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

so vereinfachen sich die Berechnungen mit diesen Polynomen, weil keine Überträge nötig sind und sich die Addition wie die Subtraktion verhält.

Addition (= Subtraktion):

$$\begin{array}{rl} & 1 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x + 0 \\ \pm & 1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x + 1 \\ \hline & 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x + 1 \end{array}$$

Polynomdivision:

$$\begin{array}{l} (1 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x + 1) : (1 \cdot x^2 + 0 \cdot x + 1) = 1 \cdot x + 1 \\ \underline{1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x} \\ \qquad 1 \cdot x^2 + 1 \cdot x + 1 \\ \qquad \underline{1 \cdot x^2 + 0 \cdot x + 1} \\ \qquad\qquad 1 \cdot x + 0 \quad \text{(Rest)} \end{array}$$

CRC-Polynome – Die Voraussetzungen

CRC steht für *Cyclic Redundancy Check* (Zyklische Redundanzprüfung).

Bitfolgen werden als Polynome mit den Koeffizienten 0 und 1 aufgefasst, wobei Addition und Subtraktion ohne Übertrag durchgeführt werden. (XOR)

An die zu versendende Bitsequenz wird eine Folge von n Bits angehängt, so dass damit die Division durch ein bestimmtes Polynom den Rest 0 ergibt.

Der Empfänger prüft dann, ob er bei der Division durch dasselbe Polynom ebenfalls den Rest 0 erhält. Wenn nicht, dann kann er sicher sein, dass bei der Übertragung Fehler passiert sind.

CRC-Generatorpolynome

Das Divisorpolynom wird *erzeugendes Polynom* (*generator polynom*) genannt.

CRC-Generatorpolynome

Das Divisorpolynom wird *erzeugendes Polynom* (*generator polynom*) genannt.

Für verschiedene Anwendungszwecke gibt es unterschiedliche erzeugende Polynome. Hier eine Auswahl:

CRC-Generatorpolynome

Das Divisorpolynom wird *erzeugendes Polynom* (*generator polynom*) genannt.

Für verschiedene Anwendungszwecke gibt es unterschiedliche erzeugende Polynome. Hier eine Auswahl:

- $x^4 + x + 1$ (CRC-4) kann 15-Bit-Rahmen prüfen

CRC-Generatorpolynome

Das Divisorpolynom wird *erzeugendes Polynom* (*generator polynom*) genannt.

Für verschiedene Anwendungszwecke gibt es unterschiedliche erzeugende Polynome. Hier eine Auswahl:

- $x^4 + x + 1$ (CRC-4) kann 15-Bit-Rahmen prüfen
- $x^5 + x^2 + 1$ (CRC-5) kann 31-Bit-Rahmen prüfen (USB)

CRC-Generatorpolynome

Das Divisorpolynom wird *erzeugendes Polynom* (*generator polynom*) genannt.

Für verschiedene Anwendungszwecke gibt es unterschiedliche erzeugende Polynome. Hier eine Auswahl:

- $x^4 + x + 1$ (CRC-4) kann 15-Bit-Rahmen prüfen
- $x^5 + x^2 + 1$ (CRC-5) kann 31-Bit-Rahmen prüfen (USB)
- $x^5 + x^4 + x^2 + 1$ kann 15-Bit-Rahmen prüfen (Bluetooth)

CRC-Generatorpolynome

Das Divisorpolynom wird *erzeugendes Polynom* (*generator polynom*) genannt.

Für verschiedene Anwendungszwecke gibt es unterschiedliche erzeugende Polynome. Hier eine Auswahl:

- $x^4 + x + 1$ (CRC-4) kann 15-Bit-Rahmen prüfen
- $x^5 + x^2 + 1$ (CRC-5) kann 31-Bit-Rahmen prüfen (USB)
- $x^5 + x^4 + x^2 + 1$ kann 15-Bit-Rahmen prüfen (Bluetooth)
- $x^{16} + x^{12} + x^5 + 1$ (CRC-16) kann 32 767-Bit-Rahmen prüfen.

CRC-Generatorpolynome

Das Divisorpolynom wird *erzeugendes Polynom* (*generator polynom*) genannt.

Für verschiedene Anwendungszwecke gibt es unterschiedliche erzeugende Polynome. Hier eine Auswahl:

- $x^4 + x + 1$ (CRC-4) kann 15-Bit-Rahmen prüfen
- $x^5 + x^2 + 1$ (CRC-5) kann 31-Bit-Rahmen prüfen (USB)
- $x^5 + x^4 + x^2 + 1$ kann 15-Bit-Rahmen prüfen (Bluetooth)
- $x^{16} + x^{12} + x^5 + 1$ (CRC-16) kann 32 767-Bit-Rahmen prüfen.

Von der Länge der prüfbaren Bitfolge muss noch die Länge des Restpolynoms (z. B. 4 Bit bei CRC-4) abgezogen werden, um die Länge des Nutzdatenfeldes zu bestimmen.

CRC-Beispiel (Sender)

Der Sender möchte die Bitfolge 01011001 an den Empfänger senden. Beide Parteien haben sich auf das erzeugende Polynom $x^4 + x + 1$ (CRC-4) geeinigt. Die Rahmenlänge beträgt 15 Bit.

CRC-Beispiel (Sender)

Der Sender möchte die Bitfolge 01011001 an den Empfänger senden. Beide Parteien haben sich auf das erzeugende Polynom $x^4 + x + 1$ (CRC-4) geeinigt. Die Rahmenlänge beträgt 15 Bit.

Der Sender stellt einen Bitrahmen der Länge 15 zusammen, indem er rechts vier Nullbits für den möglichen Rest anhängt und die eigentliche Nachricht mit Nullen auffüllt:

000010110010000

CRC-Beispiel (Sender)

Der Sender möchte die Bitfolge 01011001 an den Empfänger senden. Beide Parteien haben sich auf das erzeugende Polynom $x^4 + x + 1$ (CRC-4) geeinigt. Die Rahmenlänge beträgt 15 Bit.

Der Sender stellt einen Bitrahmen der Länge 15 zusammen, indem er rechts vier Nullbits für den möglichen Rest anhängt und die eigentliche Nachricht mit Nullen auffüllt:

000010110010000

CRC-Beispiel (Sender, Fortsetzung)

000010110010000

CRC-Beispiel (Sender, Fortsetzung)

000010110010000
10011

Setze das erzeugende Polynom 10011 ($x^4 + x + 1$) bündig von links unter die erste 1

CRC-Beispiel (Sender, Fortsetzung)

```
000010110010000
    10011
    -----
```

Setze das erzeugende Polynom 10011 ($x^4 + x + 1$) bündig von links unter die erste 1

„Subtrahiere“ mit XOR und hole die übrigen Ziffern nach unten.

CRC-Beispiel (Sender, Fortsetzung)

```
000010110010000
    10011
    -----
    00101010000
```

Setze das erzeugende Polynom 10011 ($x^4 + x + 1$) bündig von links unter die erste 1

„Subtrahiere“ mit XOR und hole die übrigen Ziffern nach unten.

Wiederhole die letzten beiden Schritte so lange bis der Rest feststeht.

CRC-Beispiel (Sender, Fortsetzung)

```
000010110010000
    10011
    -----
    00101010000
      10011
```

Setze das erzeugende Polynom 10011 ($x^4 + x + 1$) bündig von links unter die erste 1

„Subtrahiere“ mit XOR und hole die übrigen Ziffern nach unten.

Wiederhole die letzten beiden Schritte so lange bis der Rest feststeht.

CRC-Beispiel (Sender, Fortsetzung)

```
000010110010000
    10011
    -----
    00101010000
      10011
      -----
```

Setze das erzeugende Polynom 10011 ($x^4 + x + 1$) bündig von links unter die erste 1

„Subtrahiere“ mit XOR und hole die übrigen Ziffern nach unten.

Wiederhole die letzten beiden Schritte so lange bis der Rest feststeht.

CRC-Beispiel (Sender, Fortsetzung)

```
000010110010000
    10011
    -----
00101010000
   10011
   -----
001100000
```

Setze das erzeugende Polynom 10011 ($x^4 + x + 1$) bündig von links unter die erste 1

„Subtrahiere“ mit XOR und hole die übrigen Ziffern nach unten.

Wiederhole die letzten beiden Schritte so lange bis der Rest feststeht.

CRC-Beispiel (Sender, Fortsetzung)

```
000010110010000
    10011
    -----
    00101010000
      10011
      -----
      001100000
        10011
```

Setze das erzeugende Polynom 10011 ($x^4 + x + 1$) bündig von links unter die erste 1

„Subtrahiere“ mit XOR und hole die übrigen Ziffern nach unten.

Wiederhole die letzten beiden Schritte so lange bis der Rest feststeht.

CRC-Beispiel (Sender, Fortsetzung)

```
000010110010000
    10011
    -----
    00101010000
      10011
      -----
      001100000
        10011
        -----
```

Setze das erzeugende Polynom 10011 ($x^4 + x + 1$) bündig von links unter die erste 1

„Subtrahiere“ mit XOR und hole die übrigen Ziffern nach unten.

Wiederhole die letzten beiden Schritte so lange bis der Rest feststeht.

CRC-Beispiel (Sender, Fortsetzung)

```
000010110010000
    10011
    -----
    00101010000
      10011
      -----
      001100000
        10011
        -----
        0101100
```

Setze das erzeugende Polynom 10011 ($x^4 + x + 1$) bündig von links unter die erste 1

„Subtrahiere“ mit XOR und hole die übrigen Ziffern nach unten.

Wiederhole die letzten beiden Schritte so lange bis der Rest feststeht.

CRC-Beispiel (Sender, Fortsetzung)

```
000010110010000
    10011
    -----
    00101010000
      10011
      -----
      001100000
        10011
        -----
        0101100
         10011
```

Setze das erzeugende Polynom 10011 ($x^4 + x + 1$) bündig von links unter die erste 1

„Subtrahiere“ mit XOR und hole die übrigen Ziffern nach unten.

Wiederhole die letzten beiden Schritte so lange bis der Rest feststeht.

CRC-Beispiel (Sender, Fortsetzung)

```
000010110010000
    10011
    -----
    00101010000
      10011
      -----
      001100000
        10011
        -----
        0101100
         10011
         -----
```

Setze das erzeugende Polynom 10011 ($x^4 + x + 1$) bündig von links unter die erste 1

„Subtrahiere“ mit XOR und hole die übrigen Ziffern nach unten.

Wiederhole die letzten beiden Schritte so lange bis der Rest feststeht.

CRC-Beispiel (Sender, Fortsetzung)

```
000010110010000
    10011
    -----
    00101010000
      10011
      -----
      001100000
        10011
        -----
        0101100
         10011
         -----
         001010
```

Setze das erzeugende Polynom 10011 ($x^4 + x + 1$) bündig von links unter die erste 1

„Subtrahiere“ mit XOR und hole die übrigen Ziffern nach unten.

Wiederhole die letzten beiden Schritte so lange bis der Rest feststeht.

CRC-Beispiel (Sender, Fortsetzung)

```
000010110010000
    10011
    -----
    00101010000
      10011
      -----
      001100000
        10011
        -----
        0101100
         10011
         -----
         001010
```

Setze das erzeugende Polynom 10011 ($x^4 + x + 1$) bündig von links unter die erste 1

„Subtrahiere“ mit XOR und hole die übrigen Ziffern nach unten.

Wiederhole die letzten beiden Schritte so lange bis der Rest feststeht.

Ersetze die letzten 4 Positionen durch diesen Rest:
000010110011010

CRC-Beispiel (Empfänger)

Der Empfänger wendet nun dasselbe Verfahren an.

0000101100111010

CRC-Beispiel (Empfänger)

Der Empfänger wendet nun dasselbe Verfahren an.

```
0000101100111010
    10011
```

CRC-Beispiel (Empfänger)

Der Empfänger wendet nun dasselbe Verfahren an.

```
000010110011010
    10011
    -----
```

CRC-Beispiel (Empfänger)

Der Empfänger wendet nun dasselbe Verfahren an.

```
000010110011010
    10011
    -----
    00101011010
```

CRC-Beispiel (Empfänger)

Der Empfänger wendet nun dasselbe Verfahren an.

```
0000101100111010
    10011
    -----
    00101011010
      10011
```

CRC-Beispiel (Empfänger)

Der Empfänger wendet nun dasselbe Verfahren an.

```
000010110011010
    10011
    -----
    00101011010
      10011
      -----
```

CRC-Beispiel (Empfänger)

Der Empfänger wendet nun dasselbe Verfahren an.

```
000010110011010
    10011
    -----
    00101011010
      10011
      -----
      001101010
```

CRC-Beispiel (Empfänger)

Der Empfänger wendet nun dasselbe Verfahren an.

```
000010110011010
    10011
    -----
    00101011010
      10011
      -----
      001101010
        10011
```

CRC-Beispiel (Empfänger)

Der Empfänger wendet nun dasselbe Verfahren an.

```
000010110011010
    10011
    -----
    00101011010
      10011
      -----
      001101010
        10011
        -----
```

CRC-Beispiel (Empfänger)

Der Empfänger wendet nun dasselbe Verfahren an.

```
0000101100111010
    10011
    -----
    00101011010
      10011
      -----
      001101010
        10011
        -----
        0100110
```

CRC-Beispiel (Empfänger)

Der Empfänger wendet nun dasselbe Verfahren an.

```
000010110011010
    10011
    -----
    00101011010
      10011
      -----
      001101010
        10011
        -----
        0100110
         10011
```

CRC-Beispiel (Empfänger)

Der Empfänger wendet nun dasselbe Verfahren an.

```
000010110011010
    10011
    -----
    00101011010
      10011
      -----
      001101010
        10011
        -----
        0100110
         10011
         -----
```

CRC-Beispiel (Empfänger)

Der Empfänger wendet nun dasselbe Verfahren an.

```
000010110011010
    10011
    -----
    00101011010
      10011
      -----
      001101010
        10011
        -----
        0100110
         10011
         -----
         000000
```

CRC-Beispiel (Empfänger)

Der Empfänger wendet nun dasselbe Verfahren an.

```
000010110011010
    10011
    -----
    00101011010
      10011
      -----
      001101010
        10011
        -----
        0100110
         10011
         -----
         000000
```

Eigenschaften

Wo versagt die CRC-Methode?

Eigenschaften

Wo versagt die CRC-Methode?

Wenn der Fehler einem Vielfachen des erzeugenden Polynoms entspricht.

Eigenschaften

Wo versagt die CRC-Methode?

Wenn der Fehler einem Vielfachen des erzeugenden Polynoms entspricht.

Was leistet die CRC-Methode? Sie entdeckt . . .

- *1-Bit-Fehler*,

Eigenschaften

Wo versagt die CRC-Methode?

Wenn der Fehler einem Vielfachen des erzeugenden Polynoms entspricht.

Was leistet die CRC-Methode? Sie entdeckt . . .

- *1-Bit-Fehler*,
- *(isolierte) 2-Bit-Fehler*,

Eigenschaften

Wo versagt die CRC-Methode?

Wenn der Fehler einem Vielfachen des erzeugenden Polynoms entspricht.

Was leistet die CRC-Methode? Sie entdeckt . . .

- *1-Bit-Fehler*,
- *(isolierte) 2-Bit-Fehler*,
- *eine ungerade Anzahl von Bitfehlern*,

Eigenschaften

Wo versagt die CRC-Methode?

Wenn der Fehler einem Vielfachen des erzeugenden Polynoms entspricht.

Was leistet die CRC-Methode? Sie entdeckt . . .

- *1-Bit-Fehler*,
- *(isolierte) 2-Bit-Fehler*,
- *eine ungerade Anzahl von Bitfehlern*,
- *Burstfehler der maximalen Länge n (Grad des erzeugenden Polynoms)*.

Repetitionscodes

Einfacher fehlerkorrigierender Code: verdreifache jedes Bit.

Damit kann jeder Einzelfehler korrigiert werden

Beispiel:

$$01 \overset{\text{encode}}{\longrightarrow} 000111 \overset{\text{send}}{\longrightarrow} 000101 \overset{\text{decode}}{\longrightarrow} 01$$

Allgemein: Wenn ein Code bis zu d Bitfehler *korrigieren* soll, muss er mindestens den Abstand $2d + 1$ haben.

Hamming-Code

Der Hamming-Code ist ein fehlerkorrigierender Code, der jeden Einzelfehler korrigieren kann.

Beispiel: Codiere die Bitfolge 1101

Positionen	1	2	3	4	5	6	7
Bitfolge	1	0	1	0	1	0	1
Position 1	1		1		1		1
Position 2		0	1		0		1
Position 4				0	1		1

Simplex, unendlicher Puffer, perfekter Kanal

Sender

```
solange wahr:
  hole Paket von der Vermittlungsschicht
  verpacke es in einen Rahmen
  übergib den Rahmen der Bitübetragungsschicht
```

Empfänger

```
solange wahr:
  warte auf die Ankunft eines Rahmens
  hole den Rahmen von der Bitübetragungsschicht
  übergib das Paket im Rahmen der Vermittlungsschicht
```

Halbduplex, endlicher Puffer, perfekter Kanal

Sender

```
solange wahr:
  hole Paket von der Vermittlungsschicht
  verpacke es in einen Rahmen
  übergib den Rahmen der Bitübetragungsschicht
  warte auf eine Bestätigung
```

Empfänger

```
solange wahr:
  warte auf die Ankunft eines Rahmens
  hole den Rahmen von der Bitübetragungsschicht
  übergib das Paket im Rahmen der Vermittlungsschicht
  sende eine Bestätigung
```

Halbduplex, endlicher Puffer, verrauschter Kanal

Sender

```
n = 0
hole das erste Paket von der Vermittlungsschicht
solange wahr:
  verpacke es in einen Rahmen mit Rahmen.seq = n
  übergib den Rahmen der Bitübertragungsschicht
  starte Timer
  warte auf ein Ereignis
  wenn Ereignis == Rahmen:
    hole Rahmen von Bitübertragungsschicht
    wenn Rahmen.ack == n:
      stoppe den Timer
      hole das nächste Paket von der Vermittlungsschicht
      n = (n+1) % 2
```

Empfänger

```
n = 0
solange wahr:
  warte auf ein Ereignis
  wenn Ereignis == Rahmen:
    hole Rahmen von Bitübertragungsschicht
    wenn Rahmen.ack == n
      leite Rahmen an Vermittlungsschicht weiter
      n = (n+1) % 2
    Leerrahmen.ack = n
    leite Leerrahmen an Vermittlungsschicht weiter
```

Die Ausgangslage

Mehrere Stationen teilen sich einen Kanal. (z. B. Ethernet, WLAN)

Wie soll die Nutzung des Kanals koordiniert werden?

Statische Kanalzuordnung: Jedem Benutzer wird ein Teil der Bandbreite zugewiesen.

Gut, wenn die Anzahl der Benutzer klein und konstant ist.

Problematisch, wenn die Anzahl der Benutzer stark schwankt.

Das Aloha-System

Kommunikationsverfahren für Computer, das von Norman Abramson an der Universität von Hawaii in den 1970er-Jahren entwickelt wurde.

Zwischen den Inseln von Hawaii gab es damals keine Telefonverbindungen. Zur Verfügung stand der Kurzstreckenfunk mit zwei Kanälen (Upstream für Nachrichten an den Zentralrechner; Downstream für Nachrichten vom Zentralrechner)

Das Aloha-System (erfolgreiche Übertragung)

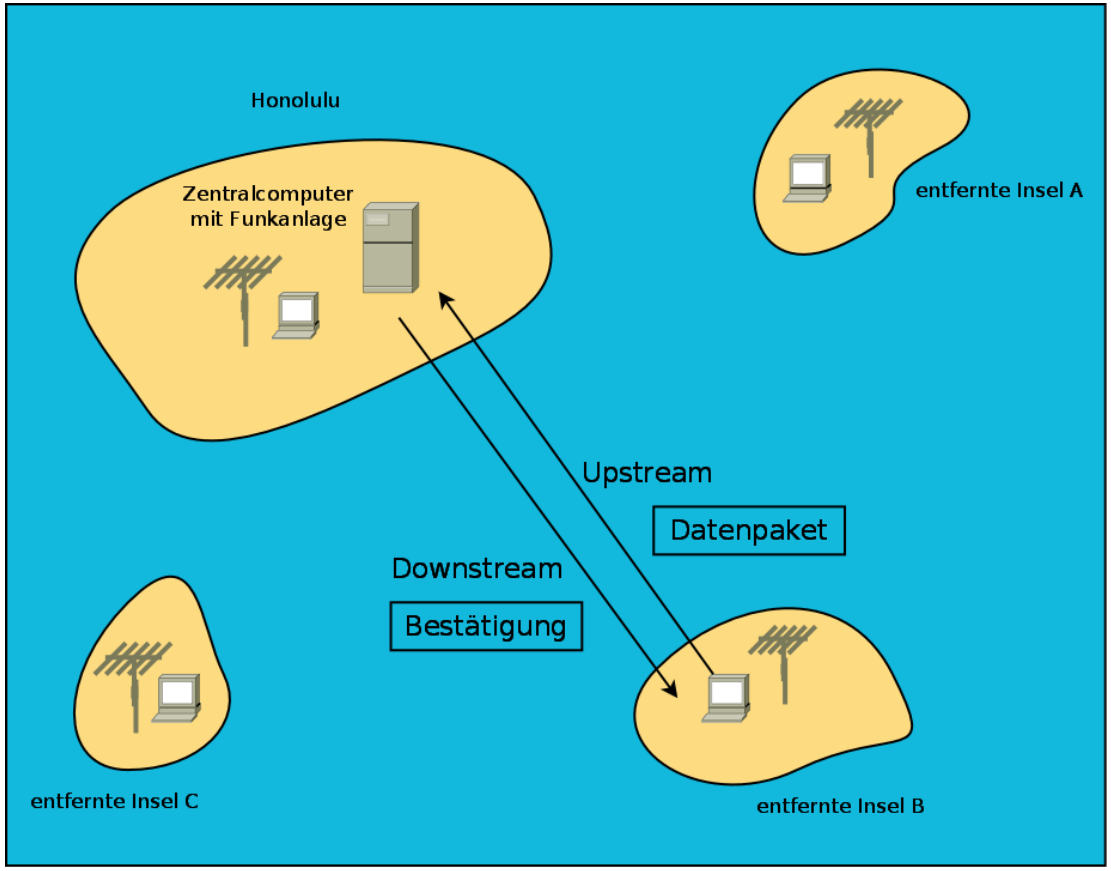


Abbildung 20: Das Aloha-System bei erfolgreiche Übertragung

Das Aloha-System (Kollision)

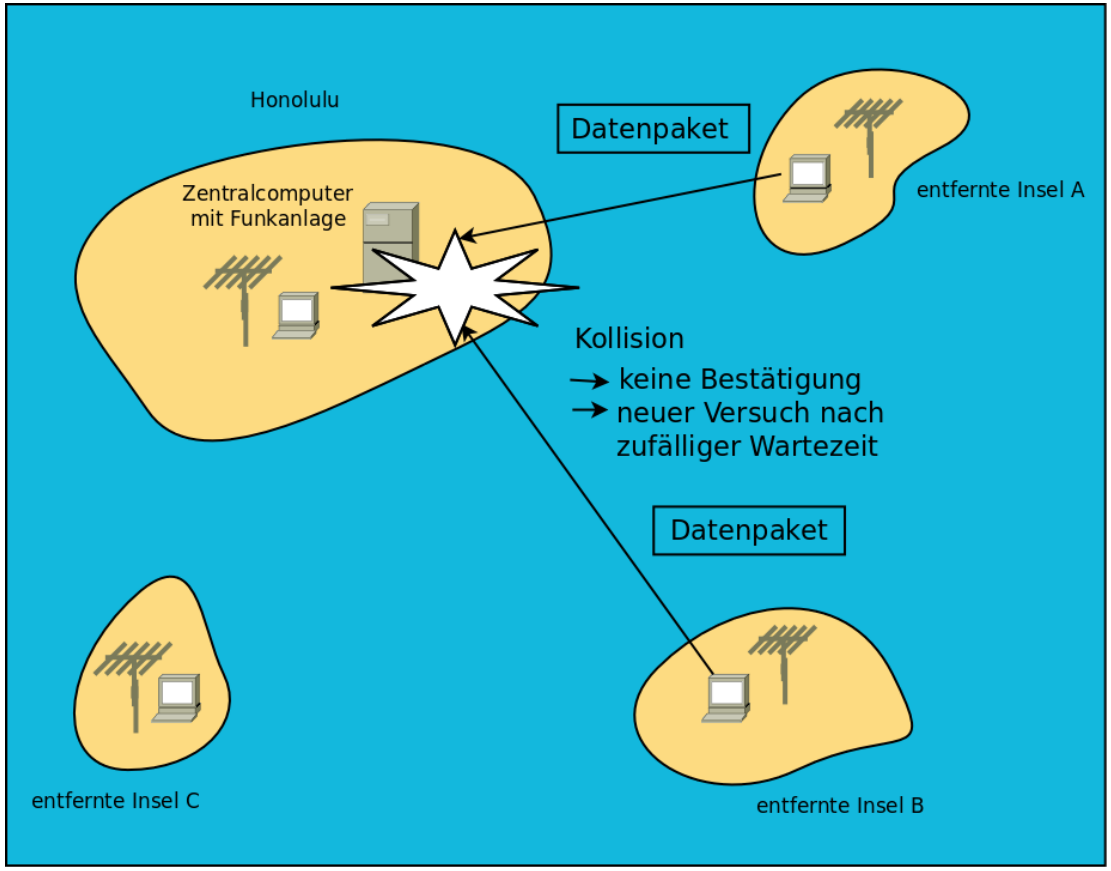


Abbildung 21: Das Aloha-System bei Kollisionen

Ethernet

Aloha-System: Broadcast-System mit zufälligem Zugriff (Multiaccess Channel, Random Access Channel)

Dieses Konzept wurde von Bob Metcalfe und David Boggs auf ein Netzwerk von Computern übertragen, die an einem langen Koaxialkabelstrang angeschlossen waren. Alle so verbundenen Computer konnten über einen Transciever elektromagnetischen Wellen senden und empfangen. Das System wurde *Ethernet* genannt.

Das System wird leistungsfähiger, wenn man vor dem Versenden von Daten überprüft, ob der Kanal überhaupt frei ist. Diesen Vorgang nennt man Trägerprüfung (carrier sense) und für das gesamte System wird das Akronym CSMA (Carrier Sense Multiple Access) verwendet.

Konfigurationsarten

Konfigurationsarten von Drahtlosnetzwerken:

- *Infrastruktur-Modus:* mit einem Access Point
- *Ad-Hoc-Modus:* direkte Kommunikation der Clients miteinander

Aufgrund der begrenzten Funkreichweite ergeben sich Schwierigkeiten, die bei kabelgebundenen Netzwerken nicht vorkommen.

Hidden-Terminal-Problem

Station A kann nicht wissen, dass auch Station C mit Station B kommunzieren will und so Interferenzen verursacht.

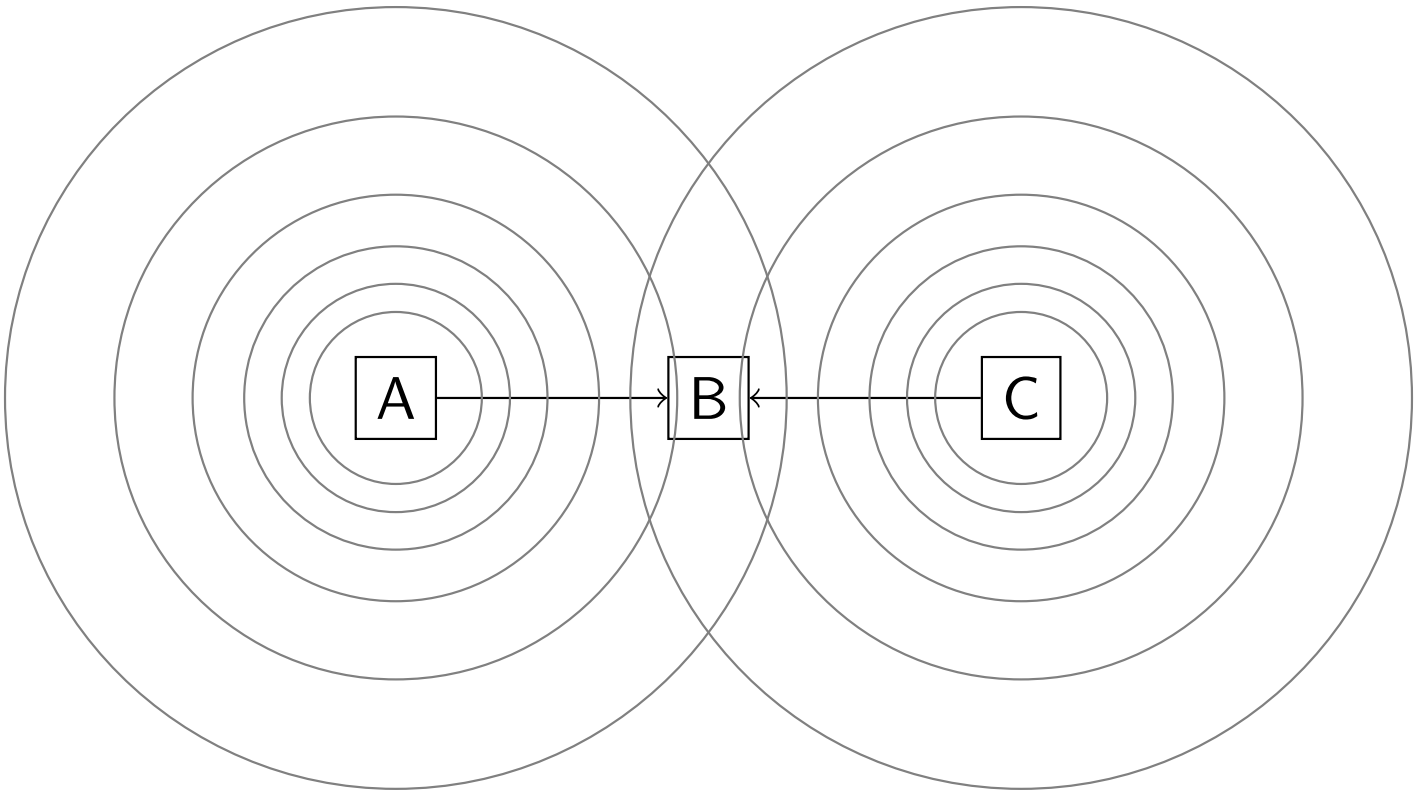


Abbildung 22: Hidden-Terminal-Problem

Exposed-Terminal-Problem

B und C nehmen an, dass sie sich gegenseitig stören, obwohl dies bei der Kommunikation B $\rightarrow$ A und C $\rightarrow$ D kein Problem ist.

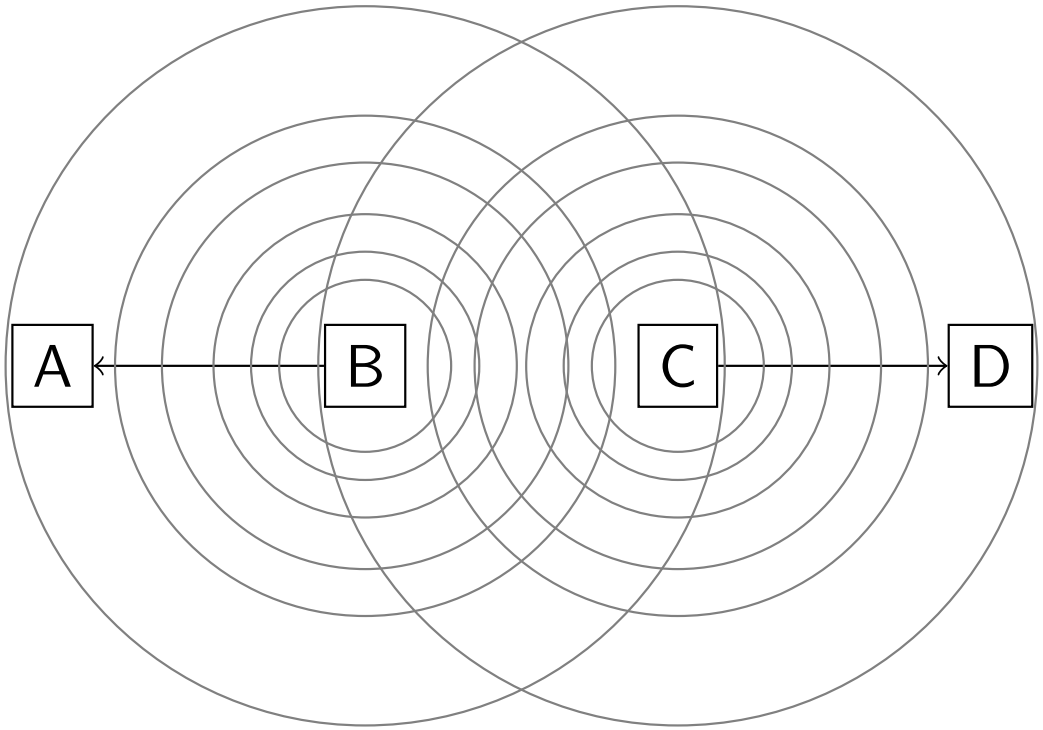


Abbildung 23: Exposed-Terminal-Problem

Die Situation in WLANs

Im Gegensatz zu Ethernet gibt es in Drahtlosnetzwerken praktisch keine Möglichkeit, Kollisionen zu erkennen, da die Funksignale stark gedämpft werden. Daher verwendet man Bestätigungen, um Kollisionen zu detektieren. Darüber hinaus muss in drahtlosen LANs, wie oben beschrieben, mit versteckten oder exponierten Stationen gerechnet werden.

Multiple Access with Collision Avoidance (MACA)

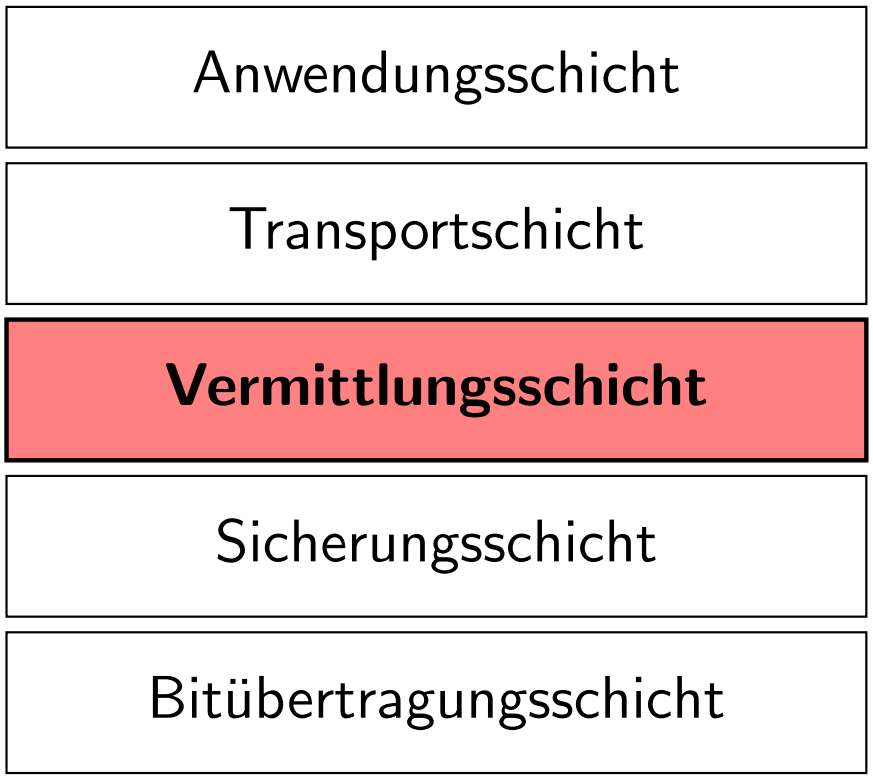
A sendet ein RTS (Request to Send) an B. Dieser kurze Steuerrahmen enthält die Länge des folgenden Datenrahmens.

B sendet darauf hin einen CTS (Clear to Send) an A zurück, der ebenfalls die Länge des Datenrahmens aus dem RTS enthält.

Nach Empfang des CTS kann A seinen Rahmen „ungestört“ an B übermitteln. Weshalb? Alle Stationen, die sich in der Reichweite von A befinden, können das RTS empfangen, und daraus ableiten, wie lange sie mit dem Senden ihrer Rahmen warten müssen. Dasselbe gilt für die Stationen, die sich in der Reichweite von B befinden.

Kollisionen können nur entstehen, wenn zwei Stationen gleichzeitig ihre RTS-Rahmen versenden. Da in diesem Fall ein CTS ausbleibt, versuchen es die beiden Stationen nach eine zufallsgesteuerten Zeitspanne erneut.

Standortbestimmung



The Big Picture

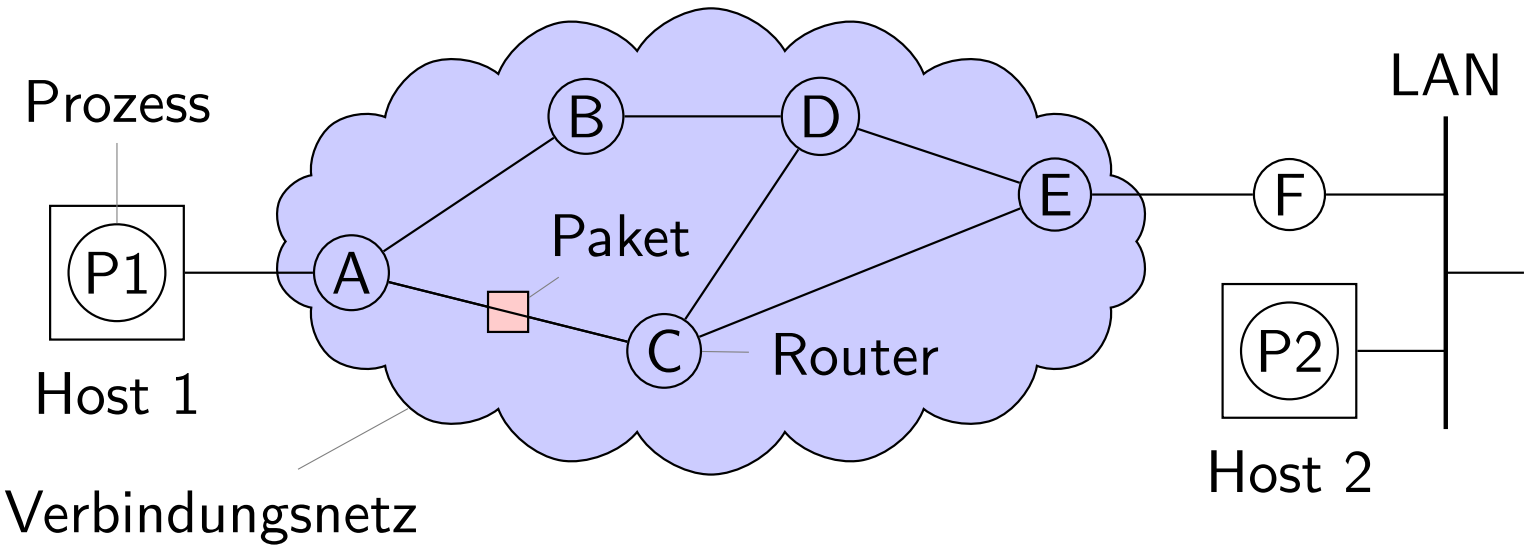


Abbildung 24: Die Situation in der Vermittlungsschicht

Store-and-forward Paketvermittlung

Ein Host sendet ein Paket an einen Router. Dieser speichert das Paket und kontrolliert die Prüfsumme. Danach wird das Paket auf die gleiche Weise auf einer der Ausgangsleitungen zum nächsten Router weitergeleitet, bis es schliesslich den Zielhost erreicht hat.

Beachte: Router können Pakete verlieren.

Dienstarten

- *Verbindungslos:* Jedes Paket wird unabhängig von den anderen befördert. (Datagramm-Netz)
- *Verbindungsorientiert*: Zuerst wird eine virtuelle Verbindung zwischen Quell- und Zielhost aufgebaut. Danach werden alle Pakete über diese Verbindung befördert. Am Ende wird die virtuelle Verbindung abgebaut. (Virtual Circuit-Netz)

Der Routing-Begriff

Router führen intern eine Tabelle die angibt, auf welcher Ausgangsleitung ein einkommendes Paket weitergeleitet werden sollen. Unter *Routing* versteht man zwei Prozesse:

- Die Bestimmung eines „guten“ Weges (Aufbau der Tabelle)
- Die konkrete Weiterleitung eines Paketes (Anwendung der Tabelle)

Graphen mit Kantengewichten

Um Routing-Algorithmen zu untersuchen, verwendet man Graphen wobei die Router durch die Knoten und die Verbindungsleitungen durch gewichtete Kanten modelliert werden. Die Kantengewichte sind normalerweise die Übertragungszeiten, können je nach Anwendungsgebiet auch andere Grössen (Kosten, Distanz) sein.

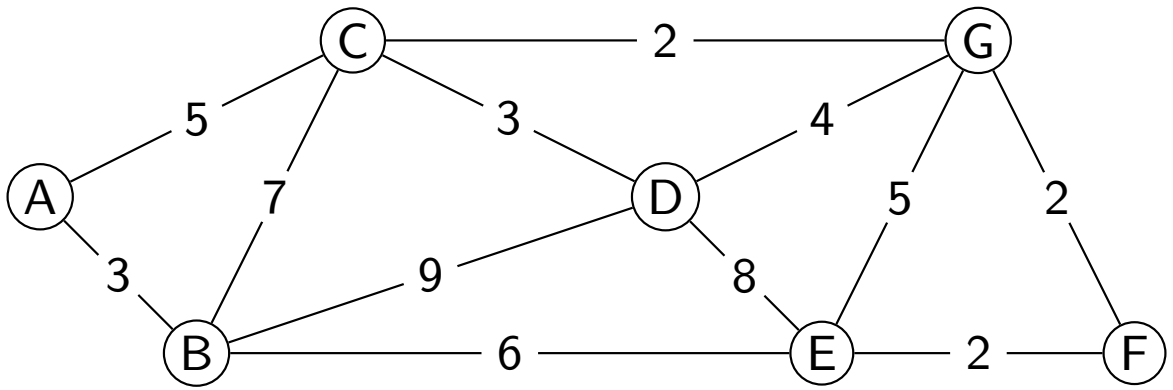


Abbildung 25: Ein ungerichteter Graph mit Kantengewichten

Das Optimalitätsprinzip von Bellmann

Wenn Router M auf dem optimalen Pfad vom Router A zum Router Z liegt, dann muss auch der Teilpfad vom Router M zum Router Z auf dem optimalen Pfad liegen.

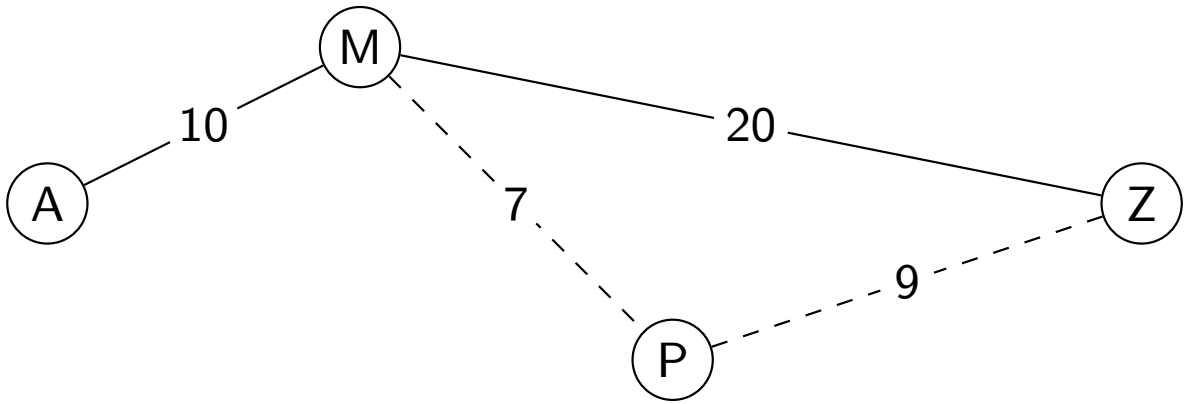


Abbildung 26: Das Optimalitätsprinzip von Bellmann

Moral: Optimale Pfade lassen sich aus optimalen Teilpfaden aufbauen.

Der Algorithmus von Dijkstra

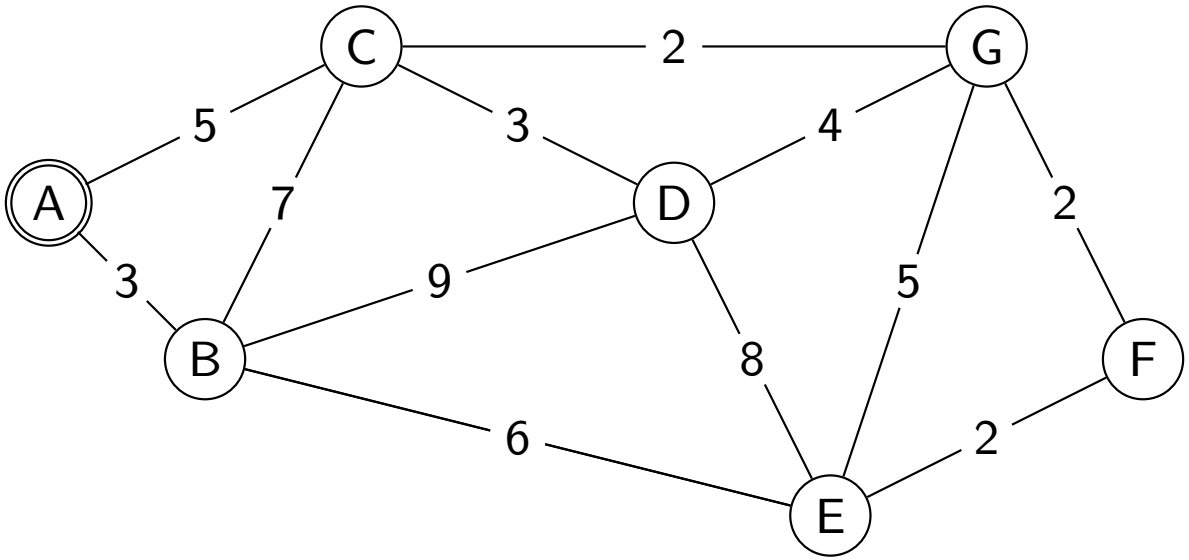
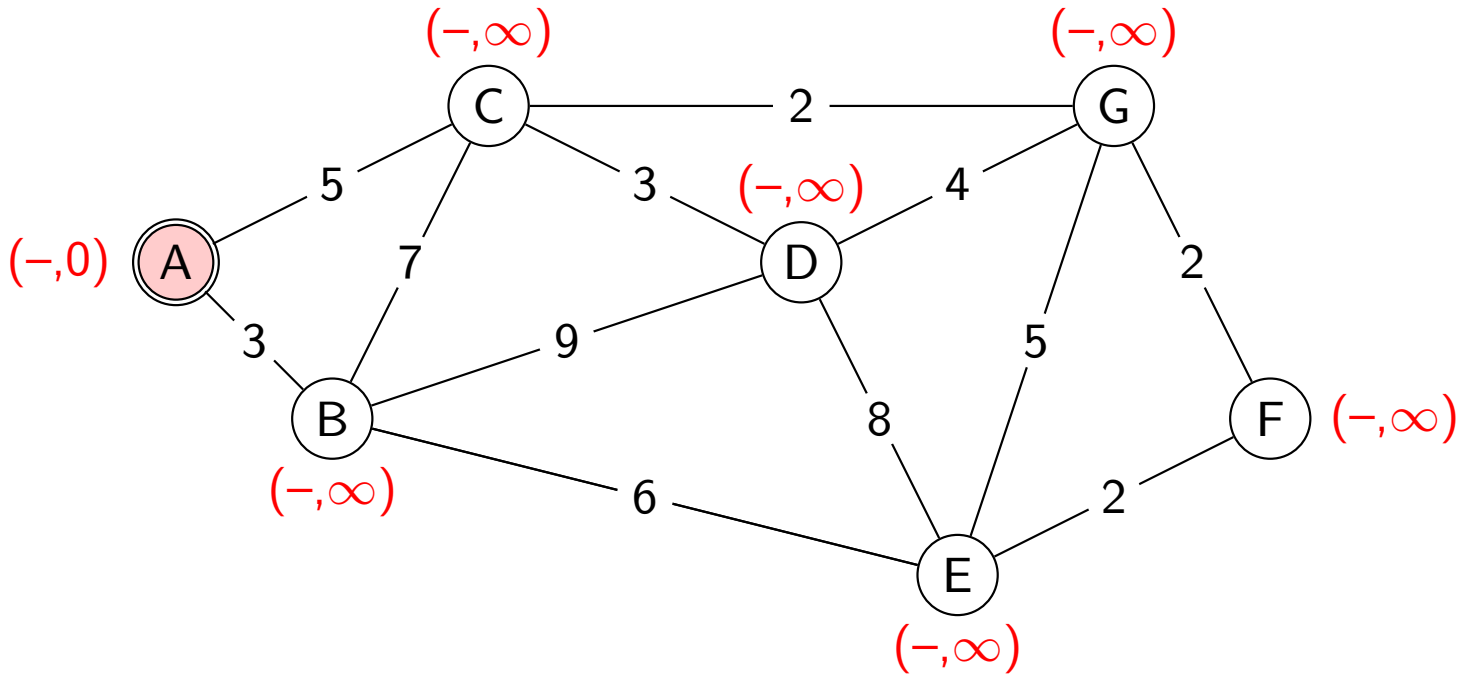


Abbildung 27: Ein Beispielgraph

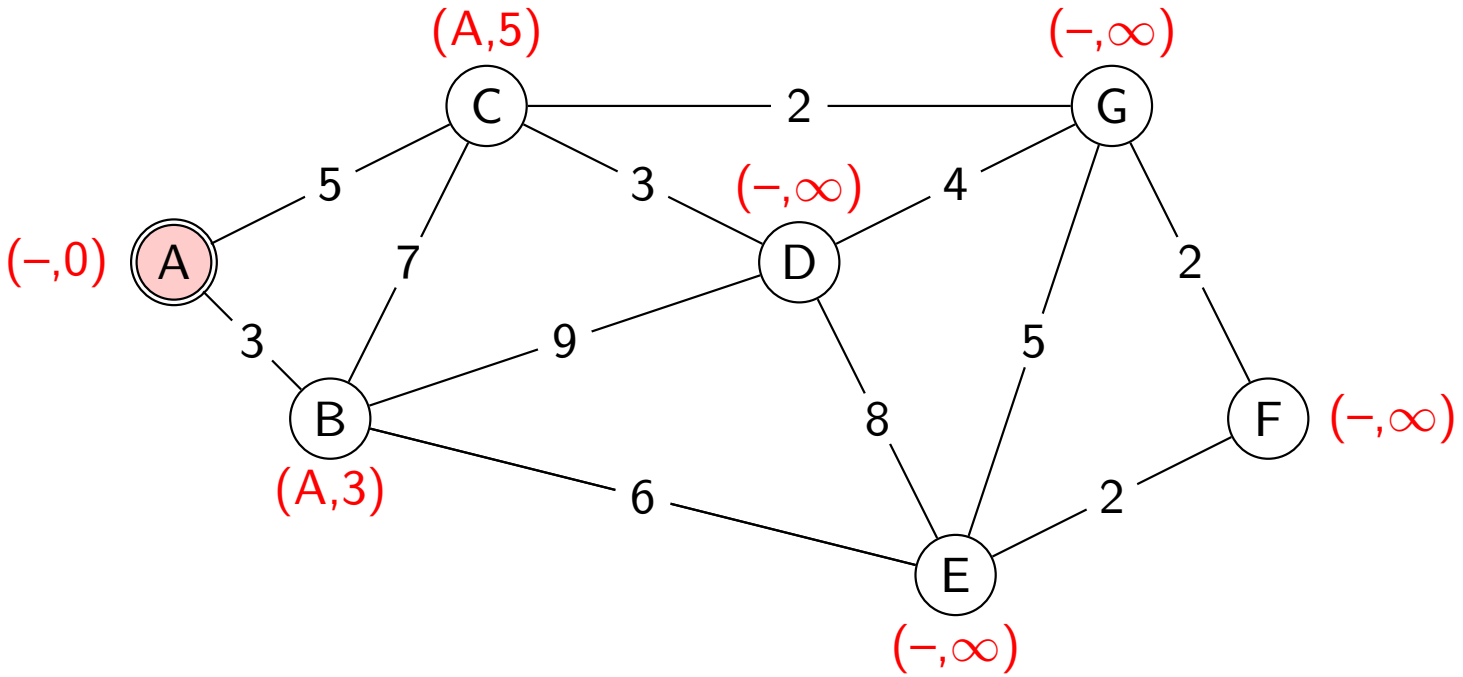
Schritt 1



Der Startknoten (A) ist der erste Arbeitsknoten (rot).

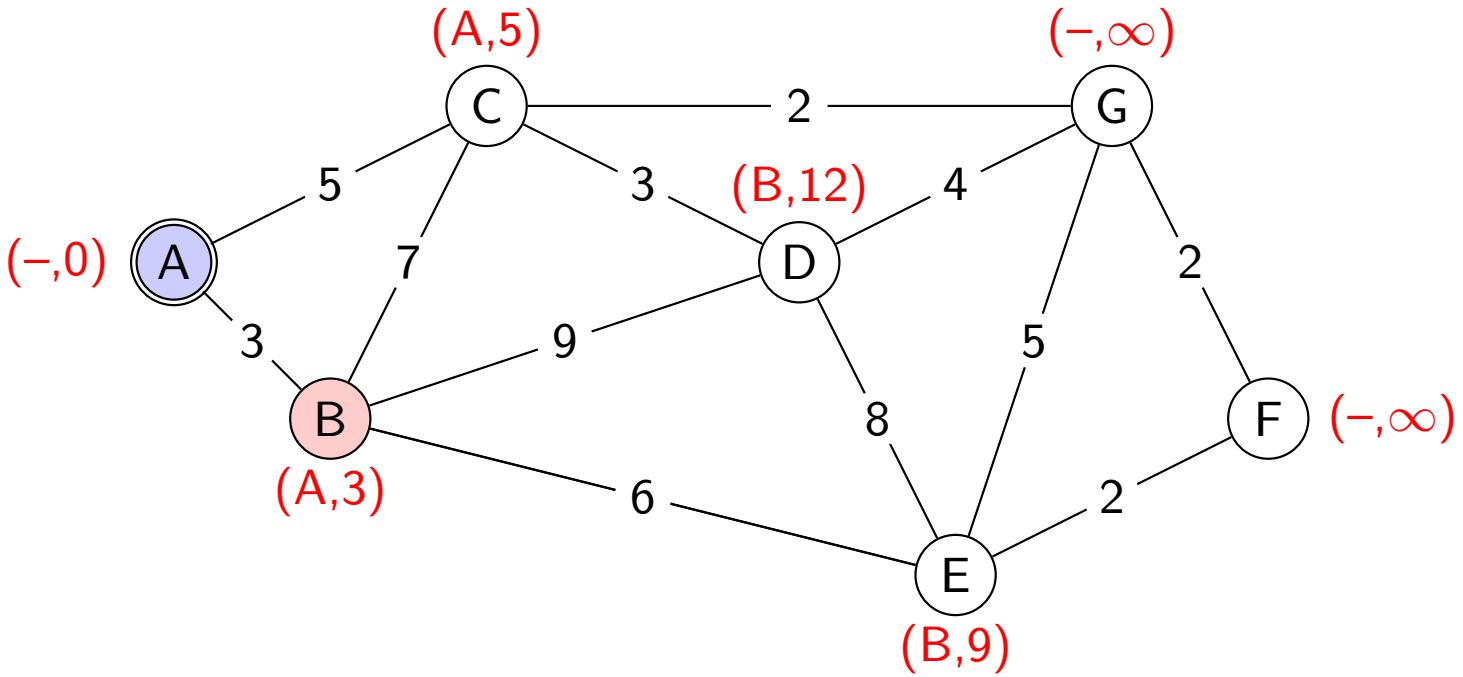
Initialisiere jeden Knoten mit der Gesamtdistanz vom Startknoten. Abgesehen vom Startknoten wird dafür ∞ gewählt. *Grund:* Falls der Graph isolierte Knoten hat, ist dies eine sinnvoller Distanzwert.

Schritt 2



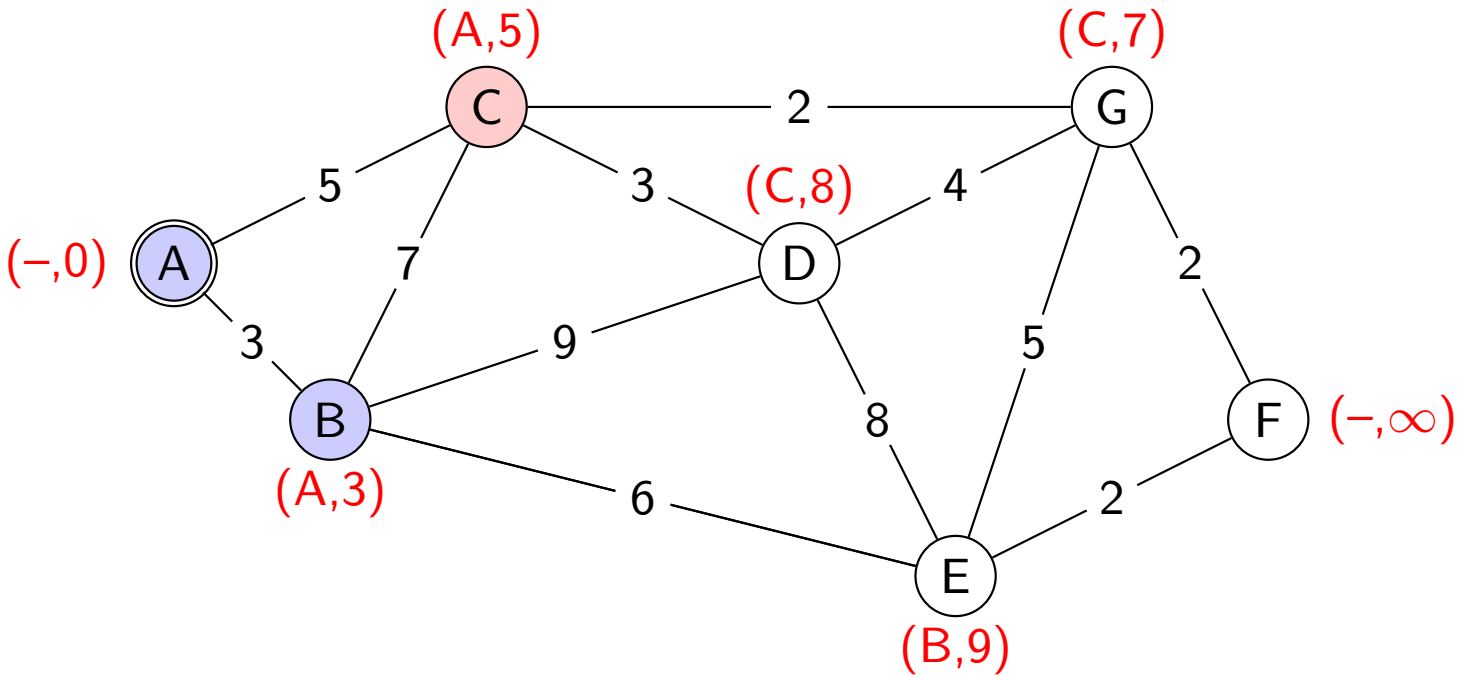
Gehe vom Arbeitsknoten A (rot) zu jedem seiner Nachbarn und untersuche, ob die Distanz zum Zielknoten „relaxiert“ (entspannt) werden kann. Falls ja, notiere den Arbeitsknoten und die neue Gesamtdistanz.

Schritt 3



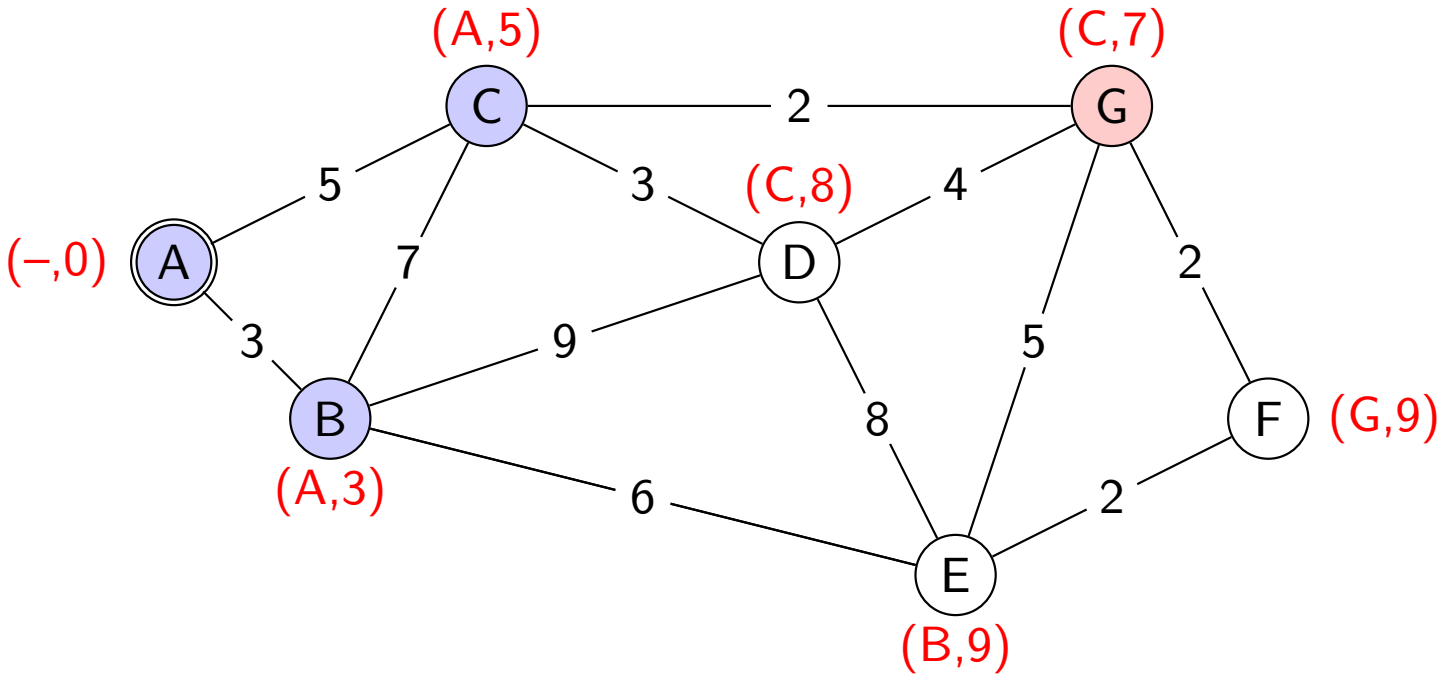
Kennzeichne den letzten Arbeitsknoten A als verarbeitet (blau) und wähle aus den übrigen Knoten denjenigen mit der kleinsten Gesamtentfernung vom Startknoten aus. Dies ist B. Relaxiere die Kanten zu allen Nachbarn des Arbeitsknotens wie im letzten Schritt.

Schritt 4



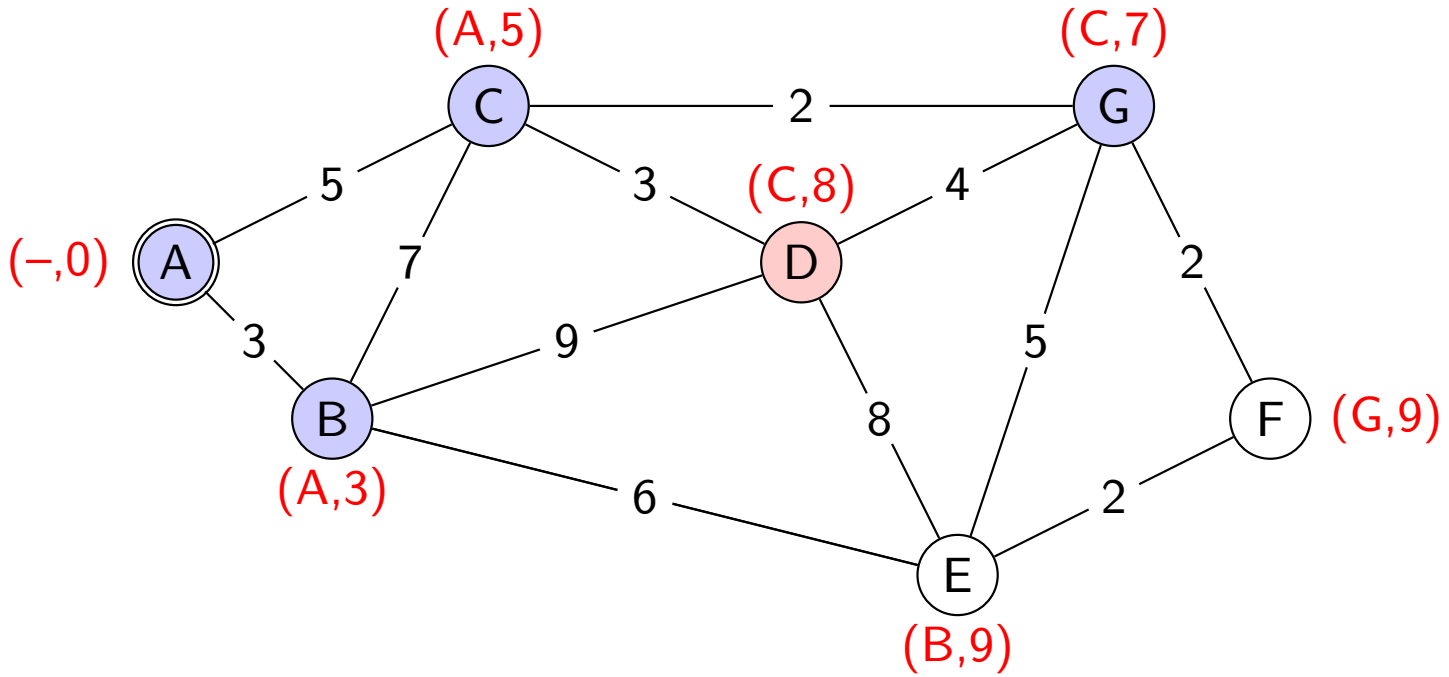
Kennzeichne den letzten Arbeitsknoten B als verarbeitet (blau) und wähle aus den übrigen Knoten denjenigen mit der kleinsten Gesamtentfernung vom Startknoten aus. Dies ist C (rot). Relaxiere die Kanten zu allen Nachbarn des Arbeitsknotens wie im letzten Schritt.

Schritt 5



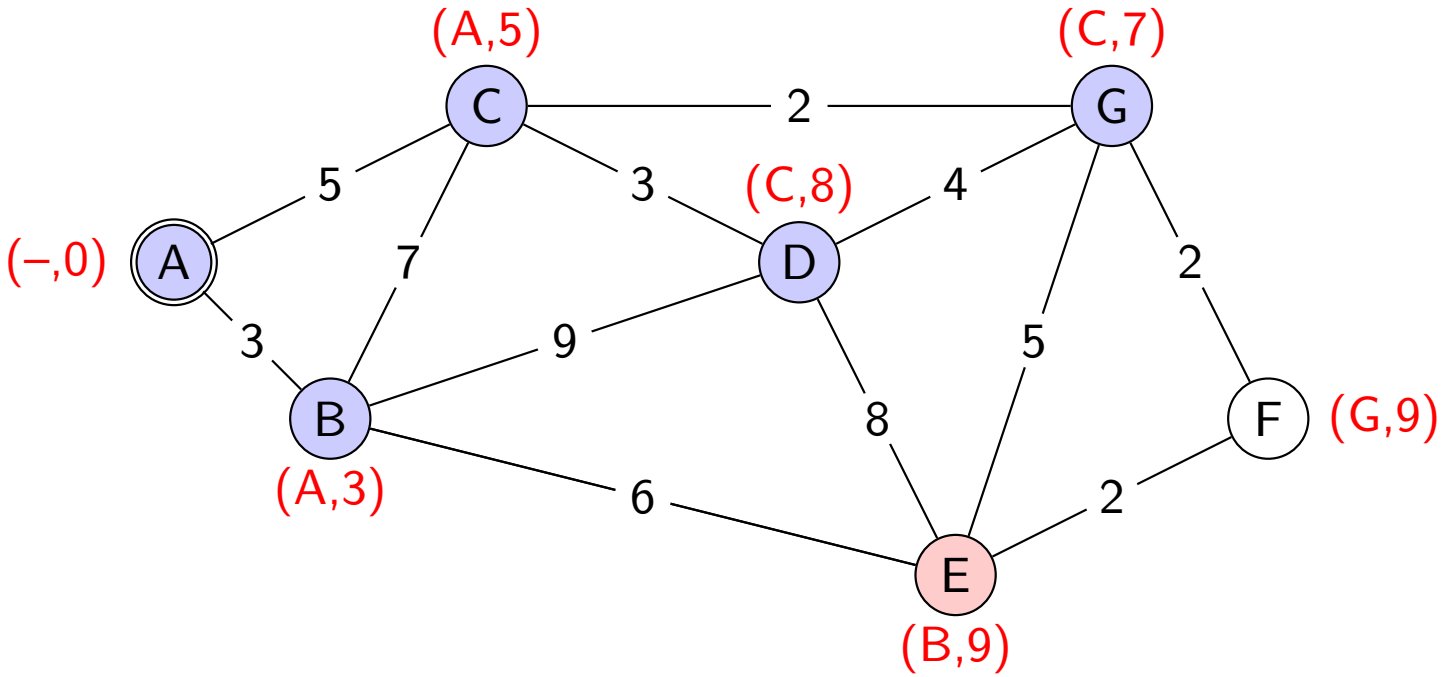
Kennzeichne den letzten Arbeitsknoten C als verarbeitet (blau) und wähle aus den übrigen Knoten denjenigen mit der kleinsten Gesamtentfernung vom Startknoten aus. Dies ist G (rot). Relaxiere wie im letzten Schritt die Kanten zu allen Nachbarn des Arbeitsknotens.

Schritt 6



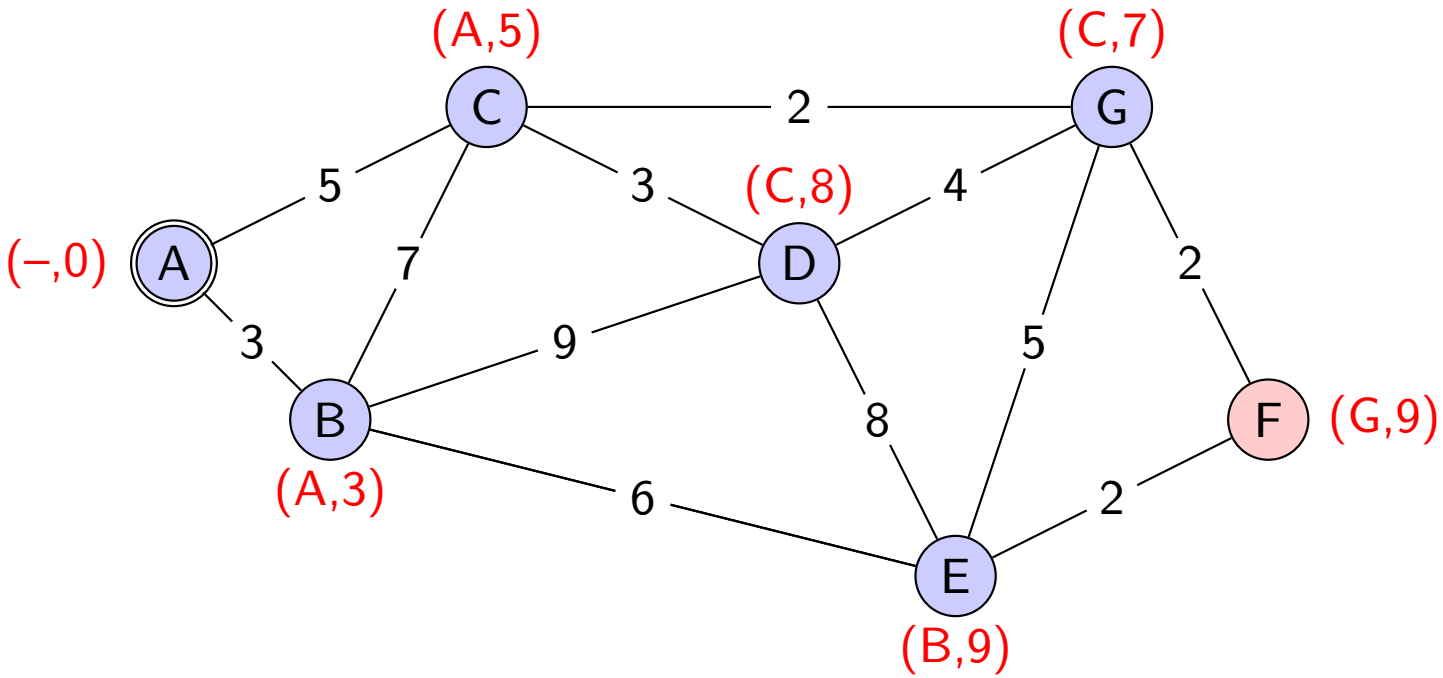
Kennzeichne den letzten Arbeitsknoten (G) als verarbeitet (blau) und wähle aus den übrigen Knoten denjenigen mit der kleinsten Gesamtentfernung vom Startknoten aus. Dies ist D (rot). Relaxiere wie im letzten Schritt die Kanten zu allen Nachbarn des Arbeitsknotens.

Schritt 7



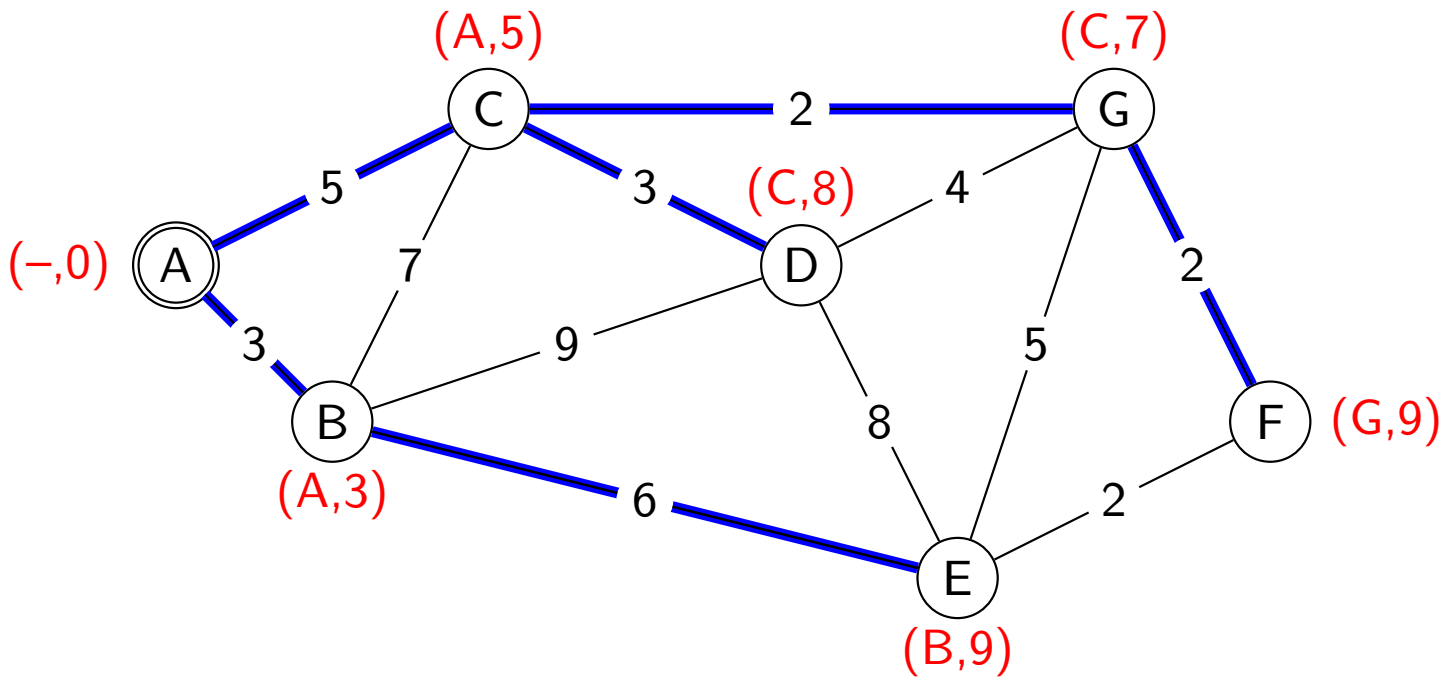
Kennzeichne den letzten Arbeitsknoten (D) als verarbeitet (blau) und wähle aus den übrigen Knoten denjenigen mit der kleinsten Gesamtentfernung vom Startknoten aus. Dies ist E (rot). Man hätte aber auch F wählen können. Relaxiere wie im letzten Schritt die Kanten zu allen Nachbarn des Arbeitsknotens.

Schritt 8



Sobald der letzte vom Startknoten erreichbare Knoten verarbeitet, d.h. relaxiert wurde, endet der Algorithmus von Dijkstra

Der Spannbaum

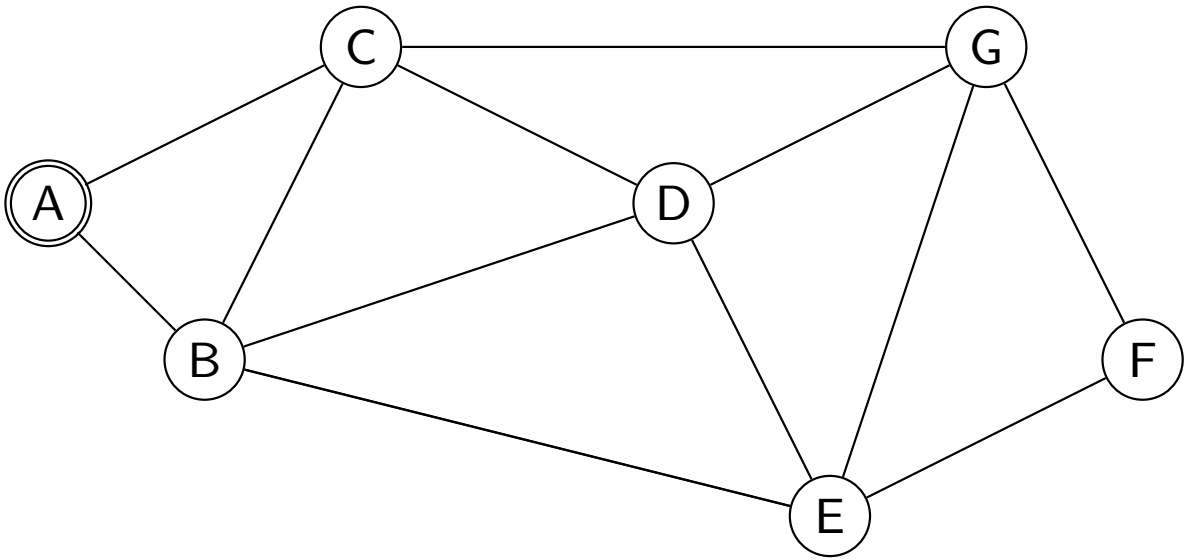


Verfolgt man gemäss der Knotenbeschriftungen die Pfade von den einzelnen Knoten zurück, erhält man einen *Spannbaum*. Das bedeutet, dass man mit diesem Baum jeden Knoten erreicht.

Auch wenn die Summe der Kantengewichte vom Wurzelknoten (A) bis zu einem anderen Knoten minimal ist, handelt es sich nicht

Flooding

Erreicht ein Paket den Router auf eine Leitung, wird es auf allen übrigen Leitungen ausgegeben.

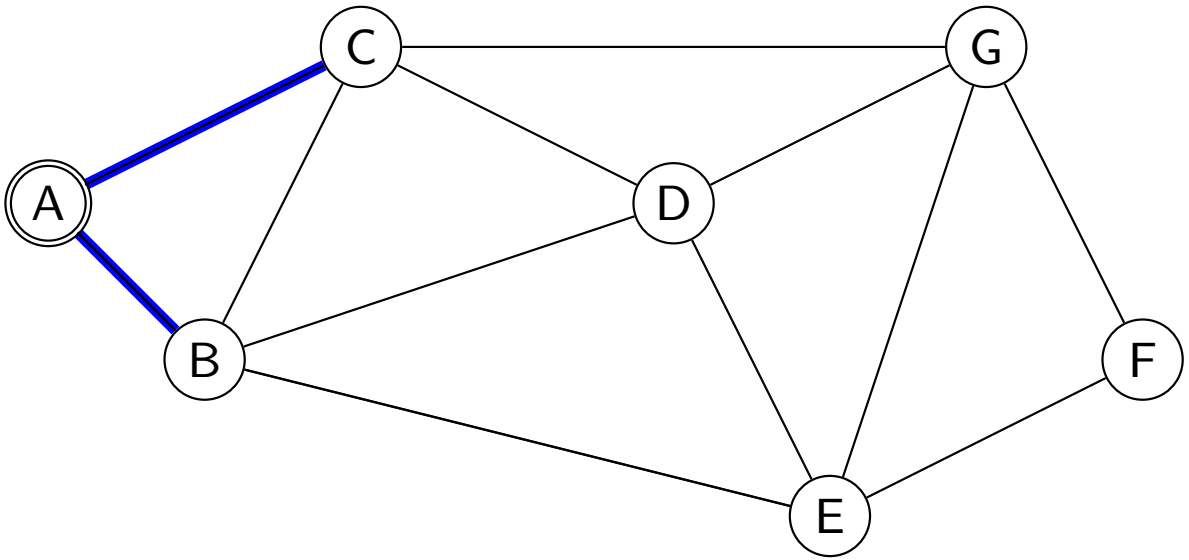


Vorteile: einfach, schnell

Nachteil: hohes Datenaufkommen

Flooding

Erreicht ein Paket den Router auf eine Leitung, wird es auf allen übrigen Leitungen ausgegeben.

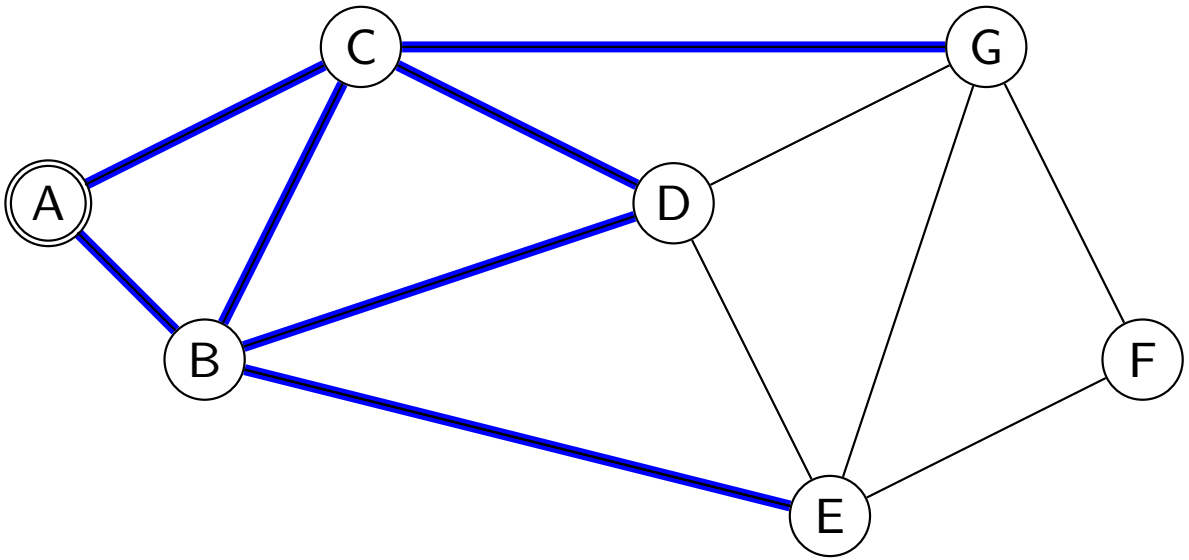


Vorteile: einfach, schnell

Nachteil: hohes Datenaufkommen

Flooding

Erreicht ein Paket den Router auf eine Leitung, wird es auf allen übrigen Leitungen ausgegeben.

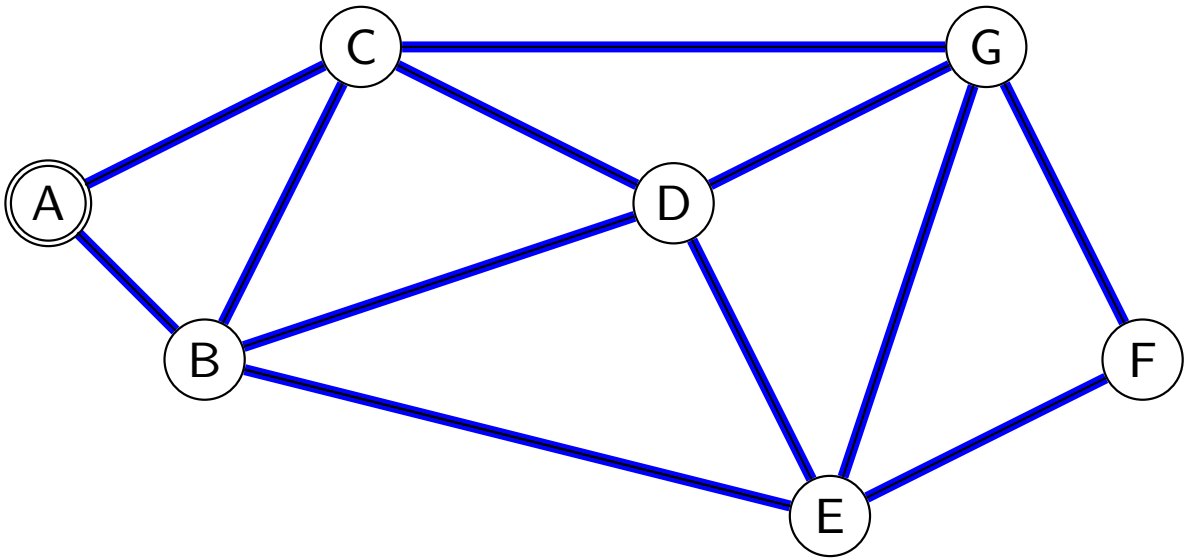


Vorteile: einfach, schnell

Nachteil: hohes Datenaufkommen

Flooding

Erreicht ein Paket den Router auf eine Leitung, wird es auf allen übrigen Leitungen ausgegeben.



Vorteile: einfach, schnell

Nachteil: hohes Datenaufkommen

Das Problem

Ist die Anzahl der von den Host ins Netz eingespeisten Pakete niedriger als die Übertragungskapazität des Netzes, werden die meisten Pakete am Bestimmungsort ankommen.

Das Problem

Ist die Anzahl der von den Host ins Netz eingespeisten Pakete niedriger als die Übertragungskapazität des Netzes, werden die meisten Pakete am Bestimmungsort ankommen.

Was geschieht, wenn das Netz an seine Kapazitätsgrenze kommt?

Das Problem

Ist die Anzahl der von den Host ins Netz eingespeisten Pakete niedriger als die Übertragungskapazität des Netzes, werden die meisten Pakete am Bestimmungsort ankommen.

Was geschieht, wenn das Netz an seine Kapazitätsgrenze kommt?

Immer mehr Pakete gehen verloren, was die Situation nur noch verschlimmert, wenn die Hosts Duplikate senden.

Das Problem

Ist die Anzahl der von den Host ins Netz eingespeisten Pakete niedriger als die Übertragungskapazität des Netzes, werden die meisten Pakete am Bestimmungsort ankommen.

Was geschieht, wenn das Netz an seine Kapazitätsgrenze kommt?

Immer mehr Pakete gehen verloren, was die Situation nur noch verschlimmert, wenn die Hosts Duplikate senden.

(Überlastung = Congestion)

Ursachen

- ungenügende Leitungskapazität im Netz
- zu langsame Router-Hardware

Flusskontrolle vs. Überlastungsüberwachung

- *Flusskontrolle:* Stellt sicher, dass ein schneller Sender einen langsamen Empfänger nicht mit Daten überschwemmt.

Flusskontrolle vs. Überlastungsüberwachung

- *Flusskontrolle:* Stellt sicher, dass ein schneller Sender einen langsamen Empfänger nicht mit Daten überschwemmt.
- *Überlastungsüberwachung:* Stellt sicher, dass ein Teilnetz den anfallenden Datenverkehr bewältigen kann.

Massnahmen gegen Überlastung

- Virtuelle Verbindungen statt Datagramme
- Warteschlangen für Pakete
- gute Routing-Algorithmen
- Verwerfen von Paketen
- Verwaltung der Paket-Lebensdauer

Überlastungsüberwachung bei virtuellen Verbindungen

Normalerweise treten in Teilnetzen aus virtuellen Verbindungen weniger Überlastungen als bei Datagramm-Teilnetzen auf, da beim Verbindungsaufbau vor der Übertragung Leistungskapazität reserviert wird.

Überlastungsüberwachung bei virtuellen Verbindungen

Normalerweise treten in Teilnetzen aus virtuellen Verbindungen weniger Überlastungen als bei Datagramm-Teilnetzen auf, da beim Verbindungsaufbau vor der Übertragung Leistungskapazität reserviert wird.

Sollte dennoch eine Überlastung auftreten, werden einfach keine weiteren virtuellen Verbindungen mehr aufgebaut.

Überlastungsüberwachung bei virtuellen Verbindungen

- Ein einigen Netzen schreiben die Router bei Überlastung ein *Warnbit* in die Datenpakete. Sendet der Empfänger eine Bestätigung an den Sender zurück, wird das Warnbit in die Bestätigung kopiert und der Sender weiss, dass er seine Datenmenge drosseln sollte.
- Satt der obigen indirekten Methode, kann ein Router auch direkt eine Mitteilung an den Sender schicken. Ein solche Nachricht heisst *Choke-Paket* (*Drosselpaket*).

Lastabwurf

Kann die Überlastung nicht durch eine der oben beschriebenen Methoden behoben werden, bleibt einem Router nichts anderes übrig, als Pakete zu verwerfen.

Was ist besser?

- Milch-Regel: *Lieber neu als alt*
- Wein-Regel: *Lieber alt als neu*

Jitter

Bei Audio- und Video-Streaming spielt es eine Rolle, in welchen Abständen die Datenpakete ankommen.

Die Schwankung bei den Paketankunftzeiten wird *Jitter* (Zittern) genannt.

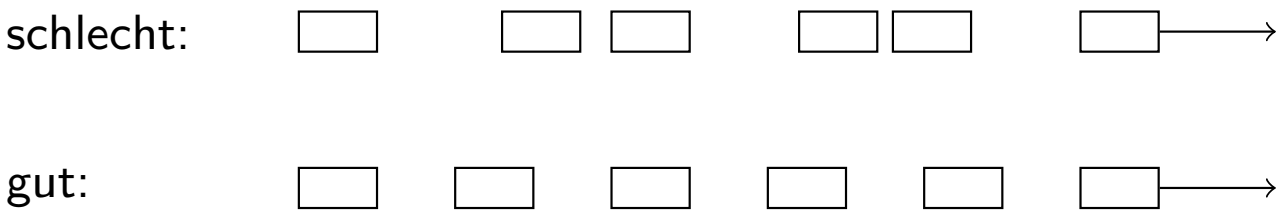


Abbildung 28: Paketfolge mit und ohne Jitter

Gewünscht wird ein möglichst niedriger Jitter.

Dienstgüte

Besteht aus:

- Zuverlässigkeit
- Übertragungsverzögerung
- Jitter
- Bandbreite

Der Leaky Bucket-Algorithmus

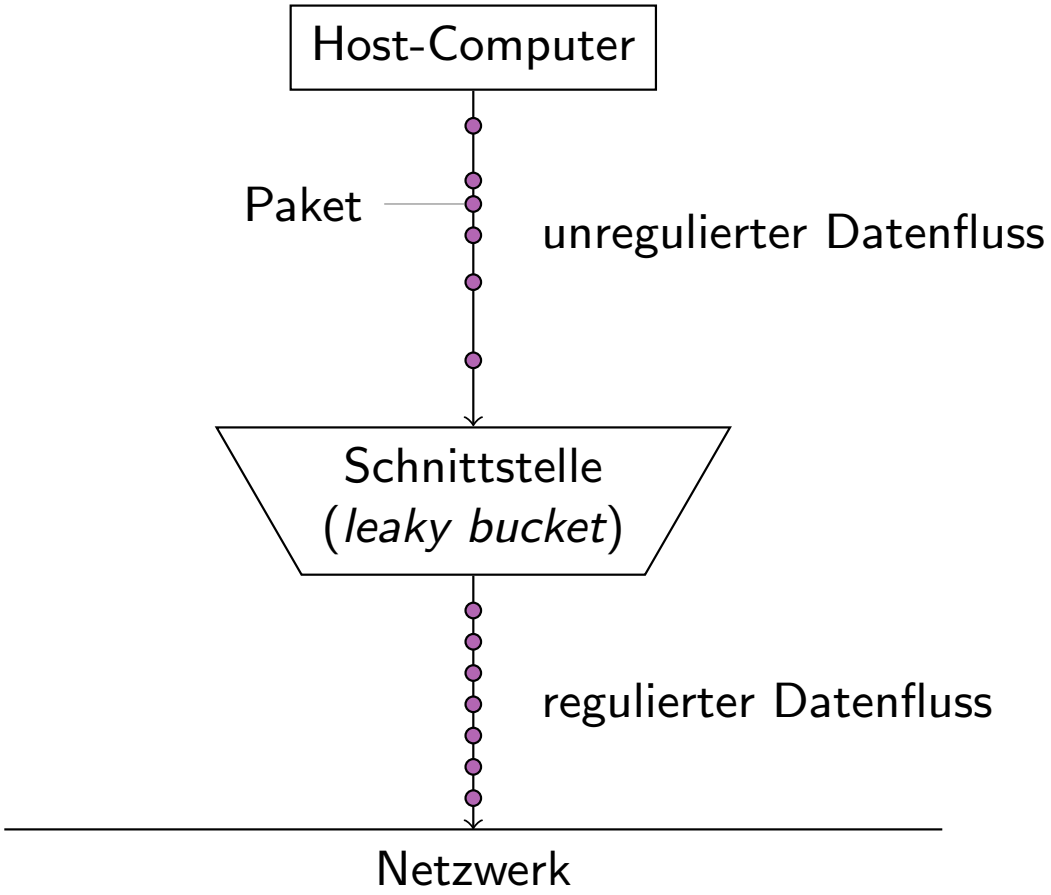


Abbildung 29: Leaky Bucket

Typen von Fragmentierung

- transparente Fragmentierung: nach jeder Teilstrecke werden fragmentierte Pakete wider defragmentiert.
- nicht transparente Fragmentierung: Fragmente bleiben über mehrere Teilstrecken hinweg fragmentiert. [Internet]

Beide Verfahre haben Vor- und Nachteile.

IPv4-Datagramme

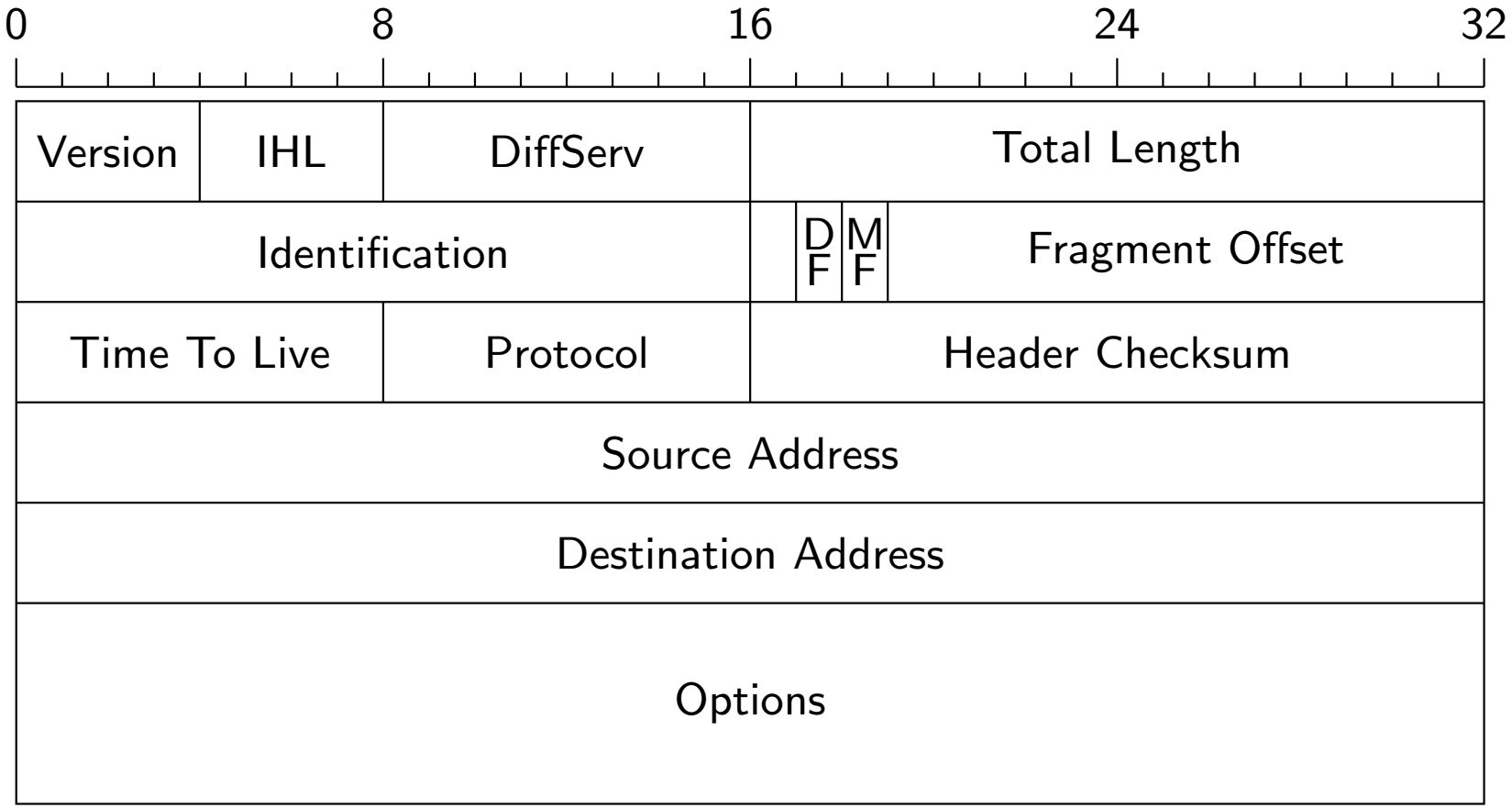


Abbildung 30: Ein IPv4-Datagramm

Version

Dieses Feld bezeichnet die Version des Protokolls, zu dem das Datagramm gehört. Derzeit dominiert noch Version IPv4. Es soll früher oder später durch IPv6 abgelöst werden.

IHL – IP Header Length

Da die Länge des Headers nicht konstant ist (siehe Options), gibt dieses Feld die Länge in 32-Bit-Wörtern an. Die minimale Wert beträgt 5, der maximale Wert 15.

Differentiated Services

Die Bedeutung dieses Feldes hat sich im Laufe der Zeit verändert. Es wurde ursprünglich zur Unterscheidung der Dienstklasse (*Type of Service*) mit verschiedenen Kombinationen aus Zuverlässigkeit und Geschwindigkeit eingerichtet. Neu wird es für Differenzierte Dienste verwendet:

- Expedited Forwarding (Express-Weiterleitung)
- Assured Forwarding (gesicherte Weiterleitung)

Die restlichen 2 Bits sind für die Überlastungsbenachrichtigung (ECN: Explicit Congestion Notification) reserviert.

Total Length

Anzahl der Bytes des gesamten Datagramms, d.h. von Header und Nutzdaten.

Identification

Eine Zahl, die dem Ziel-Host mitteilt, zu welchem Paket das Fragment gehört. Dies ist nötig, damit der Empfänger zusammengehörige Fragmente wieder zusammensetzen kann.

Bit 49

Dieses Bit wird nicht verwendet! Sollte aber auf 0 gesetzt sein.

DF

DF steht für „Don’t Fragment“ (nicht unterteilen)

MF

MF steht für „More Fragments“. Es bedeutet, dass der Empfänger noch mit weiteren Fragmenten rechnen muss, die zum selben Paket gehören.

Fragment Offset

Die Byte-Position im Paket, an die das Fragment beim Zusammensetzen gehört. Ausser beim letzten Fragment muss diese Zahl ein Vielfaches von 8 sein.

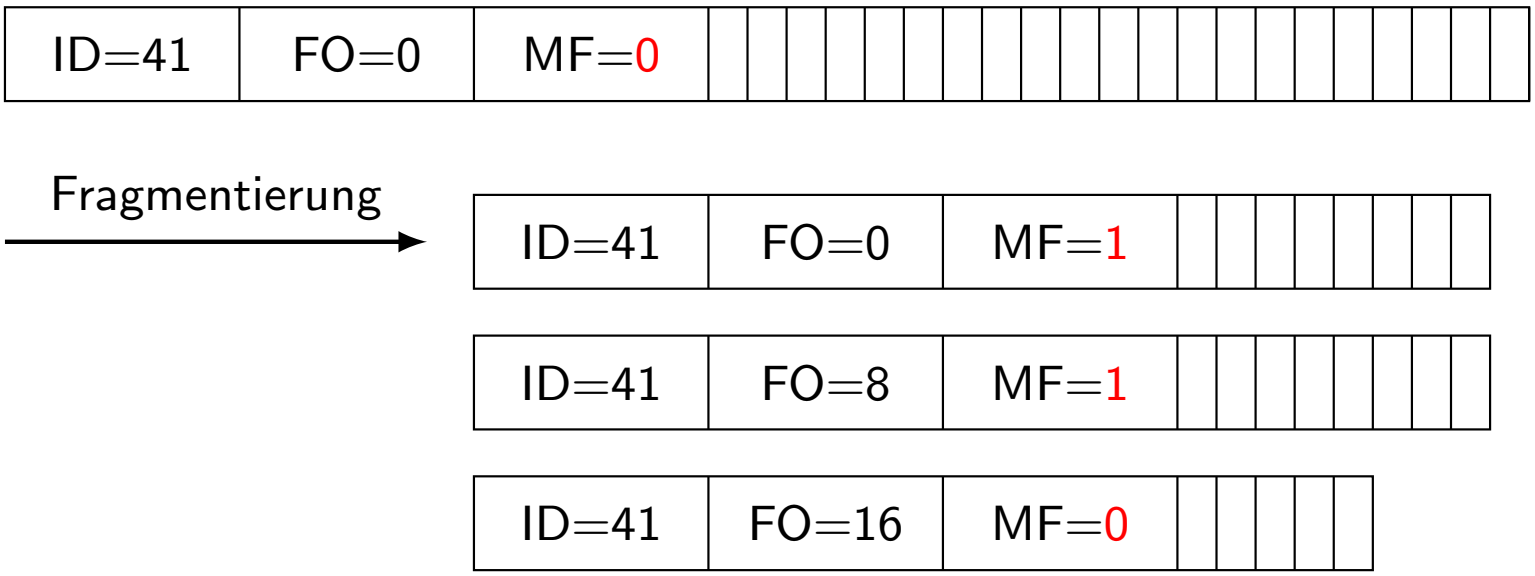


Abbildung 31: Fragmentierung

TTL

Das Feld *Time To Live* enthält einen Zähler, mit dem die Lebensdauer von Paketen begrenzt werden kann. Ursprünglich war damit die Lebensdauer in Sekunden gemeint. Mittlerweile werden damit nur noch die Teilstrecken (*Hops*) gezählt. Erreicht ein Zähler den Wert null, wird das Paket verworfen und ein Warnpaket an den Quell-Host geschickt.

Protocol

Gibt Auskunft, zu welchem Protokoll die im IPv4-Feld transportierten Daten gehören. Zum Beispiel:

- 6: TCP
- 17: UDP

http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml

Header Checksum (Berechnung)

Setze alle Bits im Prüfsummenfeld auf 0. Addiere alle 16-Bit-Felder des Headers nach den Regeln des Einerkomplements (siehe unten). Schreibe diese Summe ins Prüfsummenfeld.

```
  1111100011000011
+ 1010000001110100
------------------
 1100110010110111 (Zwischenresultat)
  1001100100110111 (den Übertrag 1 abschneiden ...)
+                1 (... und addieren)
 ------------------
  1001100100111000 (Bits kippen)
  0110011011000111 (Summe im Einerkomplement)
```

Header Checksum (Kontrolle)

Addiere alle 16-Bit-Felder des Headers nach den Regeln des Einerkomplements. Haben alle Bits im Resultat den Wert 1, wurden die Daten wahrscheinlich korrekt übertragen:

```
  1111100011000011
+ 1010000001110100
+ 0110011011000111 (Prüfsumme von oben)
------------------
 1111111111111110
  1111111111111110 (den Übertrag abschneiden ...)
+                1 (... und addieren)
 ----------------
  1111111111111111 (=> Paket ist wahrscheinlich ok)
```

Source Address, Destination Address

IP-Adresse von Quelle und Ziel. Man beachte, das diese Werte durch einen NAT-Prozess verändert werden kann.

Options

Zusätzlicher Platz, der Daten aufnehmen soll, die im ursprünglichen Entwurf nicht vorgesehen waren.

Form

- IPv4-Adressen sind 32-Bit-Addressen.
- Jede IP-Adresse bezieht sich auf eine Netzwerkschnittstelle.
- Router haben mehrere Schnittstellen und daher auch mehrere Netzwerkadressen
- Dezimalpunktschreibweise (*dotted decimal notation*): jedes der 4 Bytes wird als Dezimalzahl dargestellt.
  Beispiel: 128.208.2.151

Präfix

IP-Adressen sind hierarchisch aufgebaut.

Jeder IPv4-Adresse besteht aus einem *Netzwerkteil* variabler Länge und einem *Hostteil* (z. B. einem LAN). Beispiel: 128.208.0.089/24

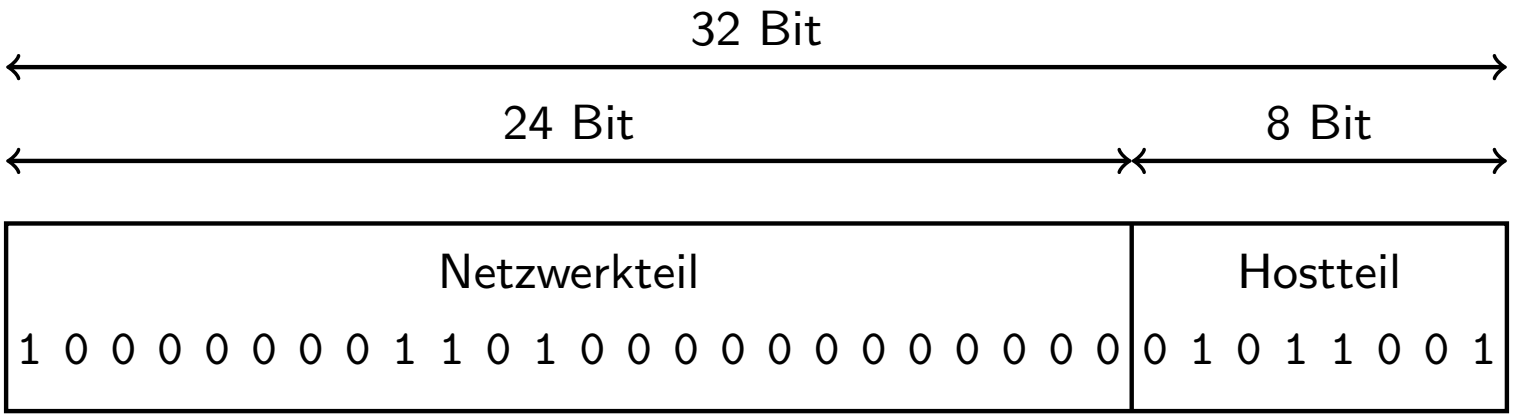


Abbildung 32: Aufbau von IPv4-Adressen

Netzmaske

Eine (Sub-)Netzmaske ist in IPv4 ebenfalls eine 32-Bit-Zahl, welche eine IP-Adresse in Netzwerkteil und Geräteteil trennt.

Durch UND-Verknüpfung der IP mit der Subnetzmaske erhält man den Netzwerkteil. Durch UND-Verknüpfung mit der invertierten Subnetzmaske erhält man den Geräteteil. Die kleinste Adresse des Netzwerks beschreibt das Netzwerk selbst, die grösste Adresse ist für den Broadcast reserviert.

Beispiel: 192.145.96.201/22

```
IP-Adresse
192.145.096.201   11000000.10010001.01100000.11001001
Subnetzmaske:
255.255.252.000   11111111.11111111.11111100.00000000
Netzwerkteil:
192.145.096.000   11000000.10010001.01100000.00000000
Hostteil:
000.000.000.201   00000000.00000000.00000000.11001001
Broadcast:
192.145.099.255   11000000.10010001.01100011.11111111
```

Für Hosts stehen $2^{10} - 2$ Adressen zur Verfügung. Warum?

ICANN

Die Internet Corporation for Assigned Names and Numbers (ICANN) verwaltet die IP-Nummern, um Konflikte zu vermeiden. Teile des Adressraums werden wiederum an regionale Behörden übergeben.

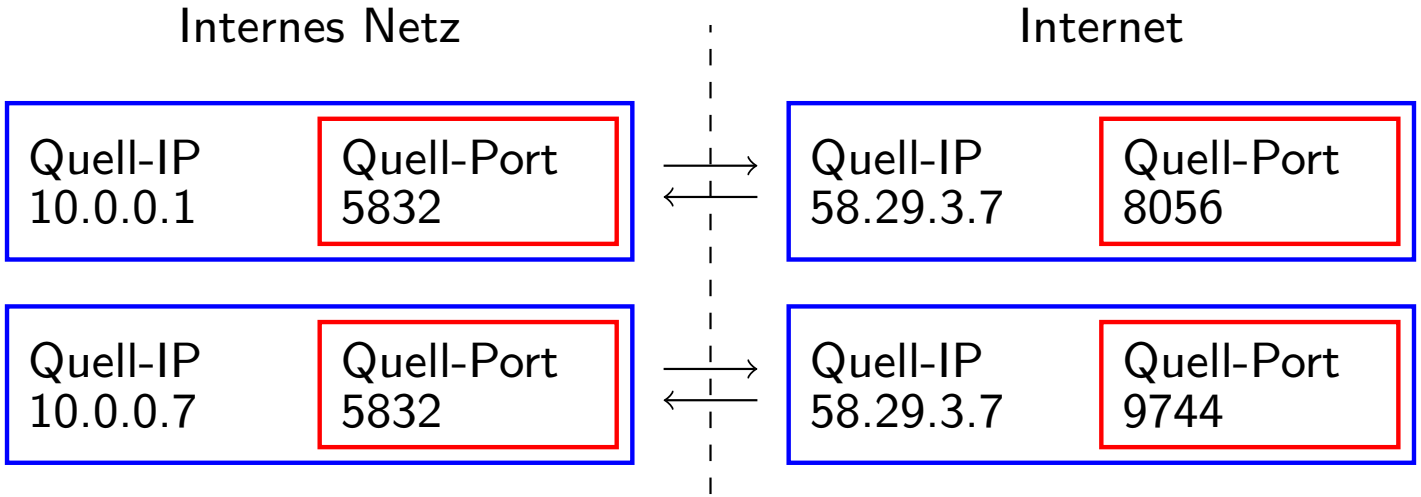
Reservierte IPv4-Nummern

Die folgenden drei Adress-Bereiche sind für den Gebrauch in „internen“ Netzen reserviert. Sie dürfen nicht im Internet verwendet werden.

```
10.0.0.0 bis 10.255.255.255/8
172.16.0.0 bis 172.31.255.255/12
192.168.0.0 bis 192.168.255.255/16
```

Network Address Translation (NAT)

Das Problem: Die etwa 4 Milliarden möglichen IPv4-Addressen sind bereits in Gebrauch.



Index	IP	Port
8056	10.0.0.1	5832
9744	10.0.0.7	5832

Abbildung 33: NAT

IPv6

- langfristiges Ziel: Ablösung von IPv4 durch IPv6
- IPv6 ist seit 1998 ein Internet-Standard! Der Wechsel findet aber nur schleppend statt.
- Auch Smartphones, Fernseher und vielleicht auch bald Kühlschränke nutzen das Internet und benötigen eine IP-Adresse.
- Wichtigstes Merkmal: 128-Bit-Adressen.

IPv6-Datagramme

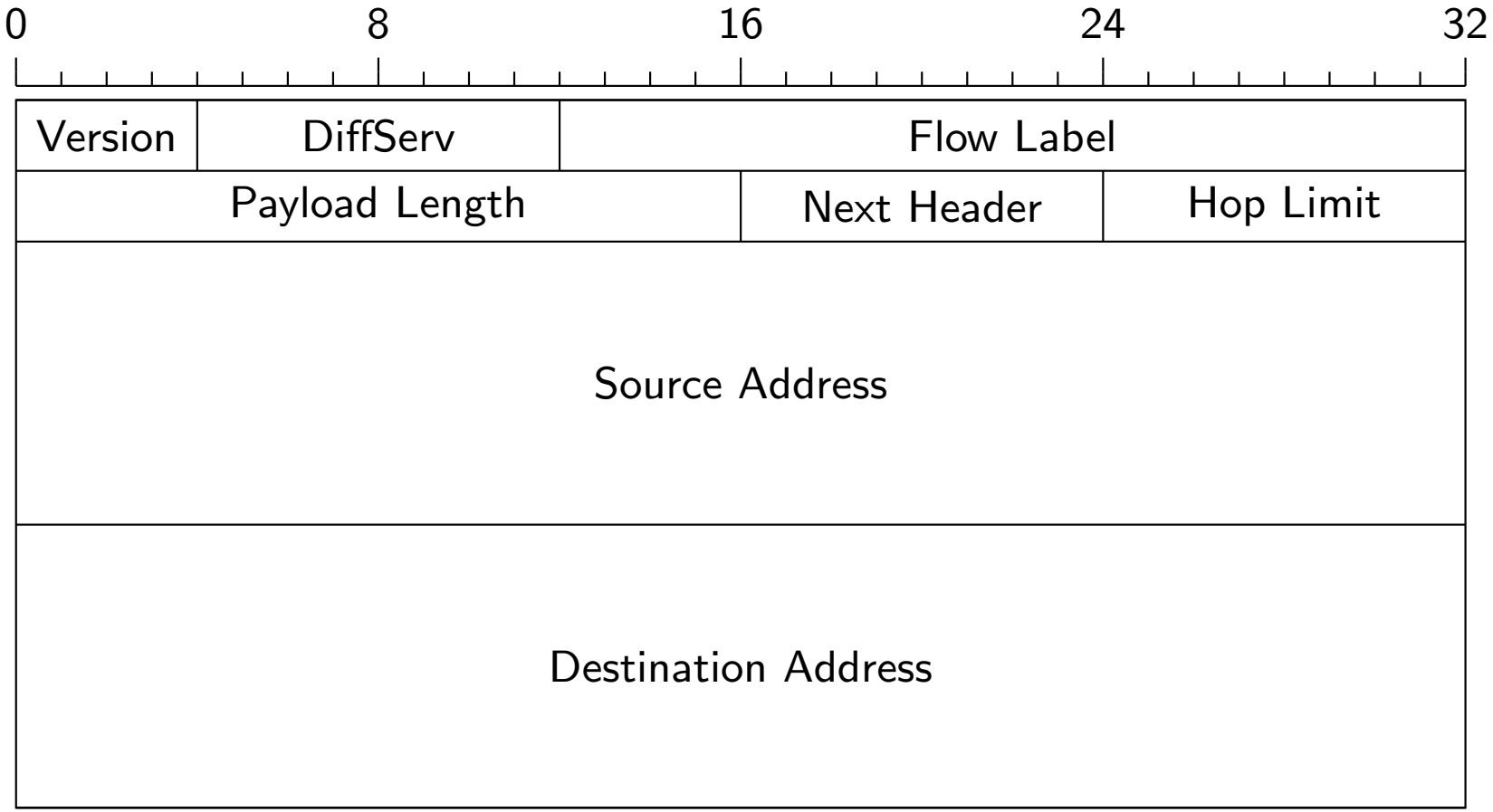


Abbildung 34: Ein IPv6-Datagramm

IPv6-Datagramme (Fortsetzung)

- *Version:* IPv6
- *Differentiated Services:* Analog zum gleichnamigen IPv4-Feld
- *Flow Label:* Ein Index für Gruppen von Paketen mit gleichen Anforderungen.
- *Payload:* Anzahl der Bytes, die auf den Header folgen.
- *Next Header:* Verweist auf einen Erweiterungs-Header
- *Hop Limit:* Maximale Teilstreckenzahl

Darstellung von IPv6-Adressnummern

Durch Doppelpunkte getrennte Gruppen von je 4 Hex-Ziffern

Beispiel: `43C2:0000:0000:0000:1AD9:0000:0037:FF08`

Vereinfachungen:

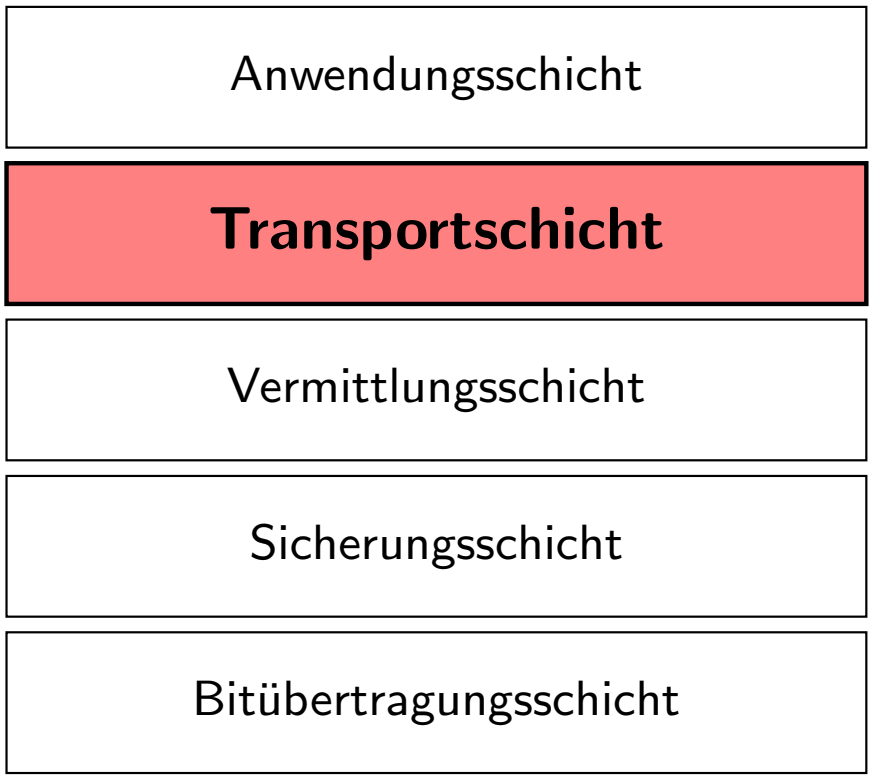
- Weglassen führender Nullen in jeder 4er-Gruppe
- Ersetzen der grössten Nullgruppe durch `::`

Also: `43C2::1AD9:0:37:FF08`

- IPv4-Adressen können integriert werden, indem man ihnen zwei Doppelpunkten voranstellt.

Beispiel: `::123.45.67.89`

Standortbestimmung



Was leistet die Transportschicht?

Die Transportschicht soll der Anwendungsschicht einen zuverlässigen, gut funktionierenden Dienst zur Verfügung stellen.

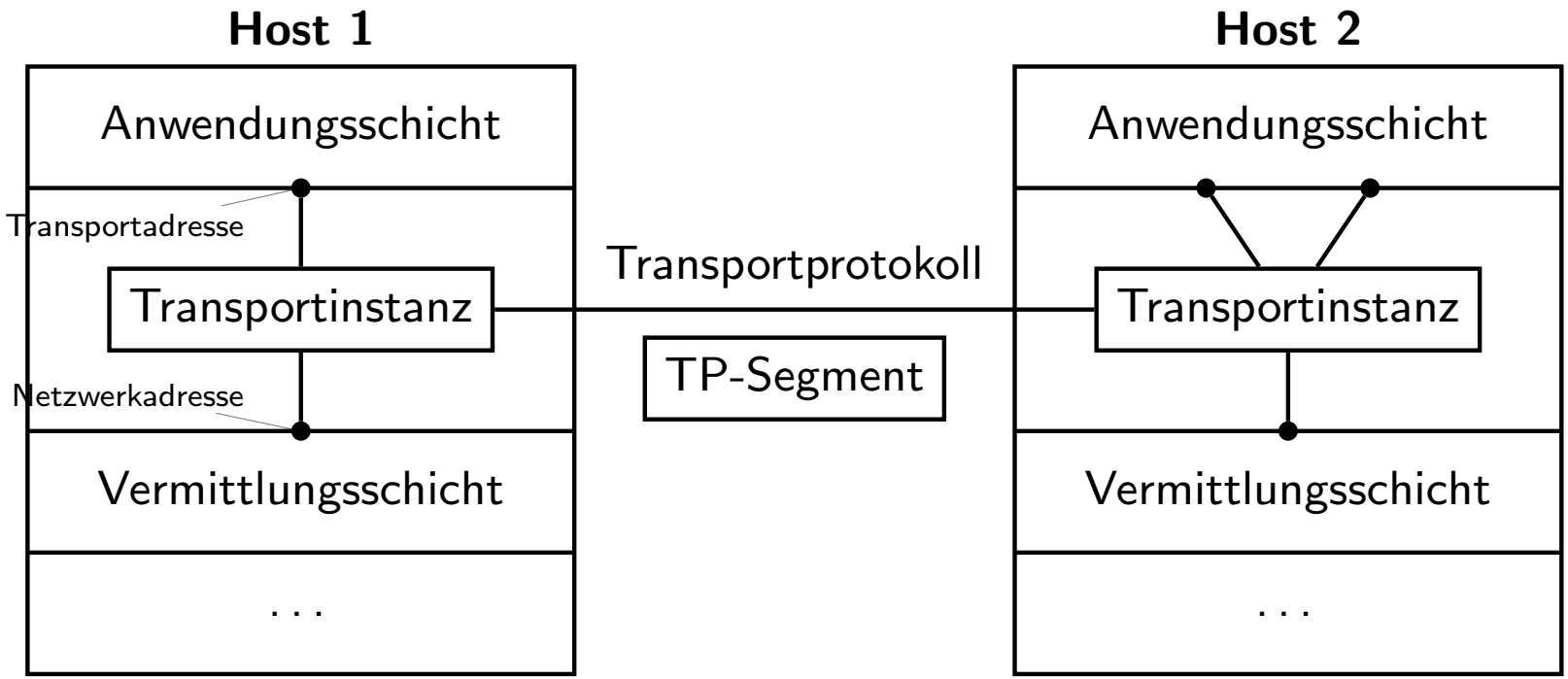


Abbildung 35: Beziehung der obersten drei Schichten

Kapselung der Dateneinheiten

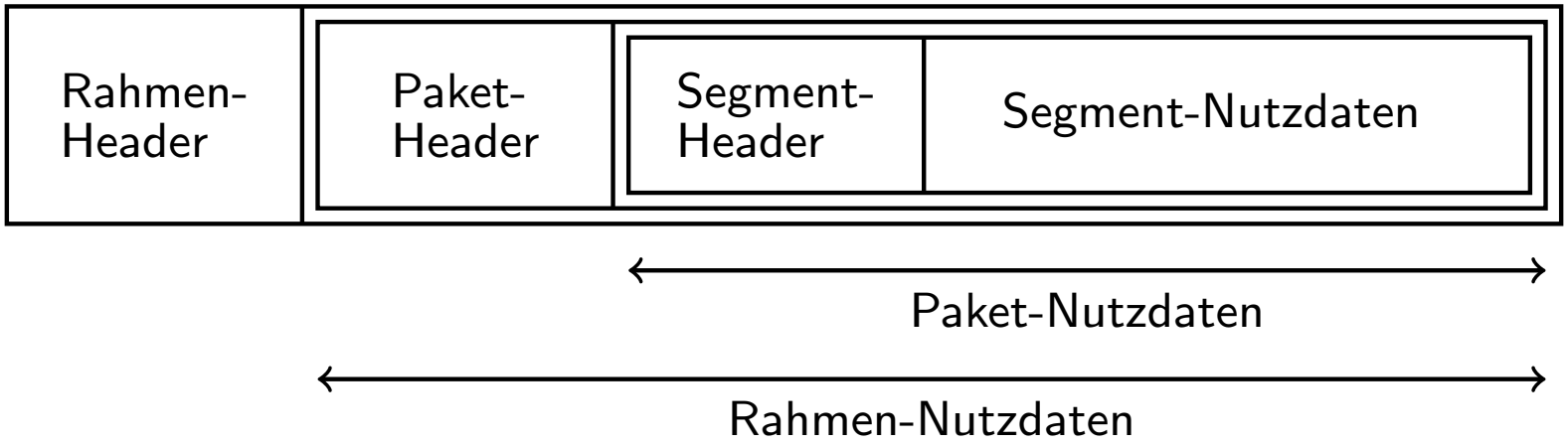


Abbildung 36: Kapselung von Segmenten und Paketen

In der Sicherungsschicht heissen die Dateneinheiten Rahmen.

In der Vermittlungsschicht heissen die Dateneinheiten Pakete.

In der Transportschicht heissen die Dateneinheiten Segmente.

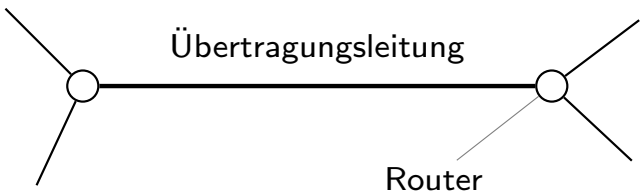
Dienst-Primitive der Transportschicht

Primitive	**Segment**	**Bedeutung**
LISTEN	keines	Blockiert, bis ein Prozess eine Verbindung aufbauen möchte
CONNECT	CONNECTION REQUEST	Versucht, aktiv eine Verbindung aufzubauen
SEND	DATA	Sendet Informationen
RECEIVE	keines	Blockiert, bis ein Paket ankommt
DISCONNECT	DISCONNECTION REQUEST	Die betreffende Seite will die Verbindung freigeben.

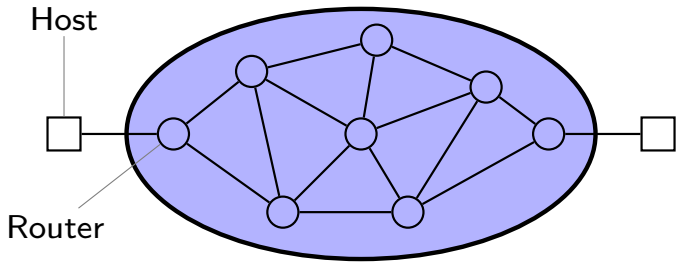
Tabelle 1: Dienst-Primitive der Transportschicht

Vergleich von Sicherungs- und Transportschicht

Sicherungsschicht



Transportschicht



Gemeinsamkeiten:

- Fehlerüberwachung
- Steuerung der Reihenfolge
- Flusskontrolle

Unterschiede:

- Verbindungsaufnahme
- Speicherkapazität
- Anzahl der Verbindungen

Adressierung

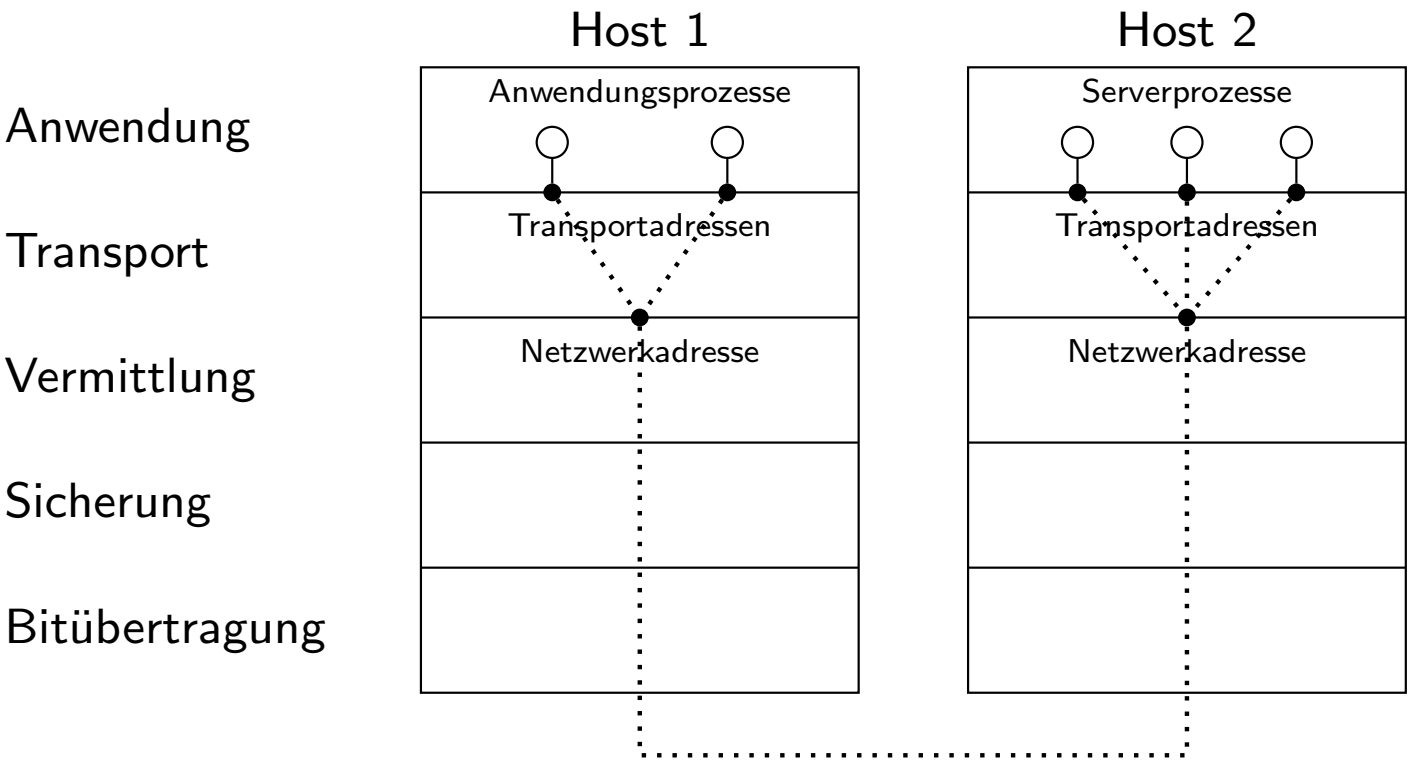


Abbildung 37: TSAPs und NSAPs

TSAP: Transport Service Access Point oder kurz *Port*

NSAP: Network Service Access Point

Zugang zu Netzwerkprozessen

- feste TSAP-Adressen (Well Known Ports)
- Portmapper
- Prozess-Server

Naive Herangehensweise

Die Transportinstanz der Datenquelle sendet eine `CONNECTION REQUEST`-Dateneinheit an die Transportinstanz des Ziels und wartet darauf, dass diese zur Bestätigung eine `CONNECTION ACCEPTED`-Dateneinheit zurückschickt.

Die zentralen Probleme

- Reihenfolge der Segmente
- Duplikate aufgrund von Datenstau und Retransmission

Lösungsansatz

1. Durchnummerieren der Segmente
2. Verhindern, dass sich zwei Segmente mit derselben Nummer im Netz befinden.

Initial Sequence Numbers (INR)

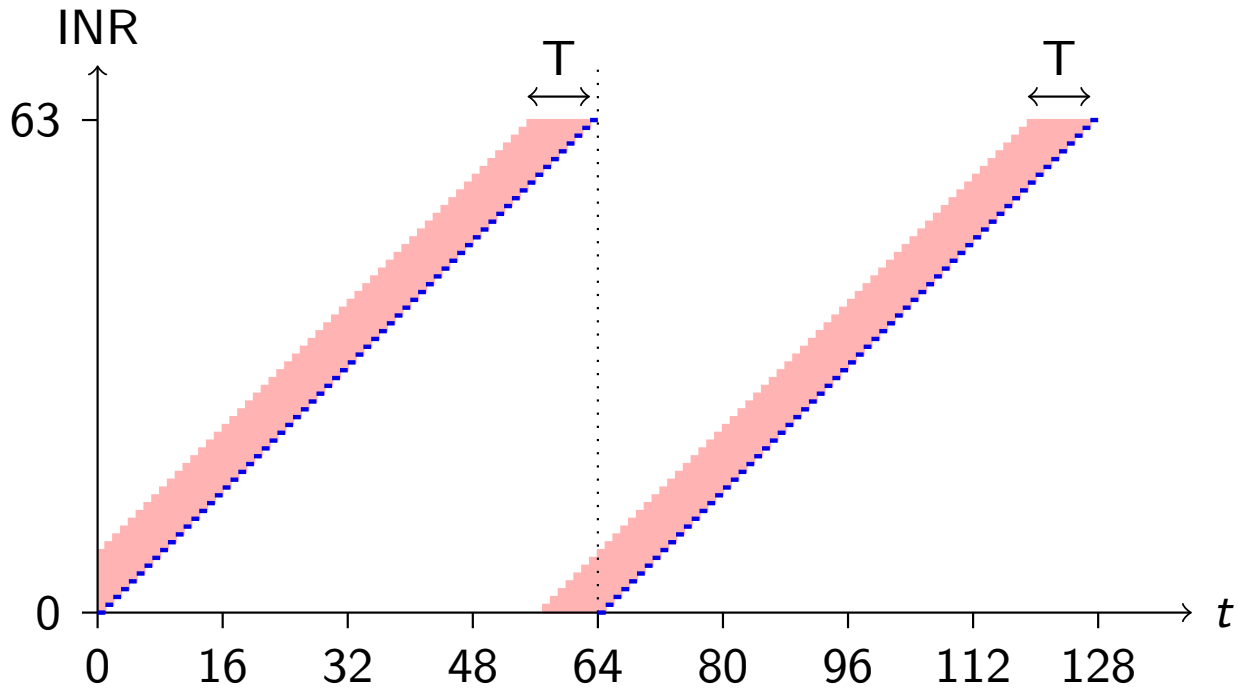


Abbildung 38: Anfangsfolgenummern

INR: Die letzten k Bits eines Digitalzählers (hier: $k = 6$)

T: Zeitspanne vom Versand eines Segments bis zum Ende der

Dreiwege-Handshake (R. Tomlinson, 1975)

```
 A                                                    B
-->  <SEQ 17><SYN><>                                 -->
     A synchronisiert mit ISN 17.

<--  <SEQ 23><SYN, ACK=18><>                         <--
     B synchronisiert mit ISN 23
     und bestätigt 17+1 von A.

-->  <SEQ 18><ACK 24><10 Datenbytes>                 -->
     A bestätigt 23+1 von A und sendet Daten.

<--  <SEQ 24><ACK 28><>                              <--
     B bestätigt A's Daten. ...
```

Verbindungsabbau

- symmetrisch: Die Verbindung wird in jede Richtung einzeln abgebaut.
- asymmetrisch: Einer der Kommunikationspartner beendet die Verbindung (Gefahr von Datenverlust).

Problem: Pakete können verloren gehen.

Das Zwei-Armeen-Problem

- Die blaue Armee ist der grauen Armee überlegen, wenn beide Teile gleichzeitig angreifen.
- Die Abstimmung des Angriffs ist nur über Boten möglich, der unbemerkt das Tal mit der grauen Armee durchqueren muss.

Abbildung 39: Das Zwei-Armeen-Problem

Können sich die beiden Teile der blauen Armee zuverlässig absprechen?

Verbindungssabbau: Beispiel 1

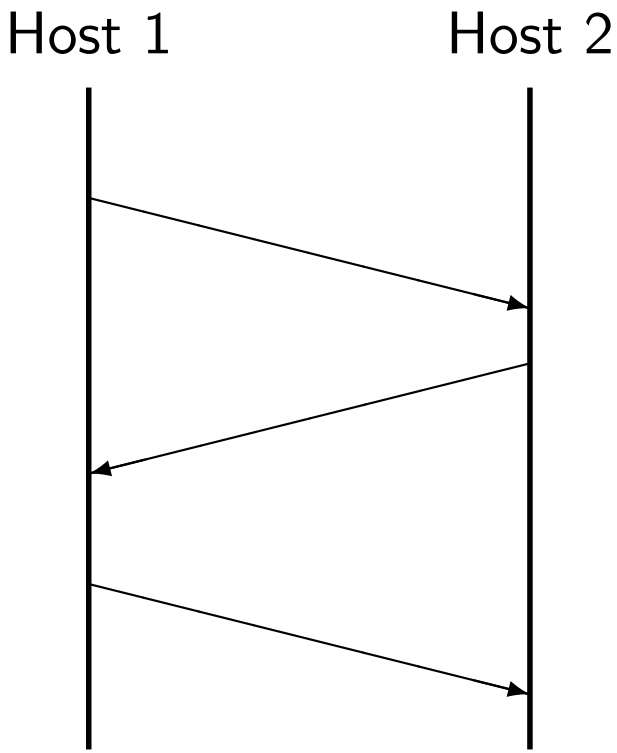


Abbildung 40: Verbindungssabbau ohne Paketverlust

Verbindungssabbau: Beispiel 1

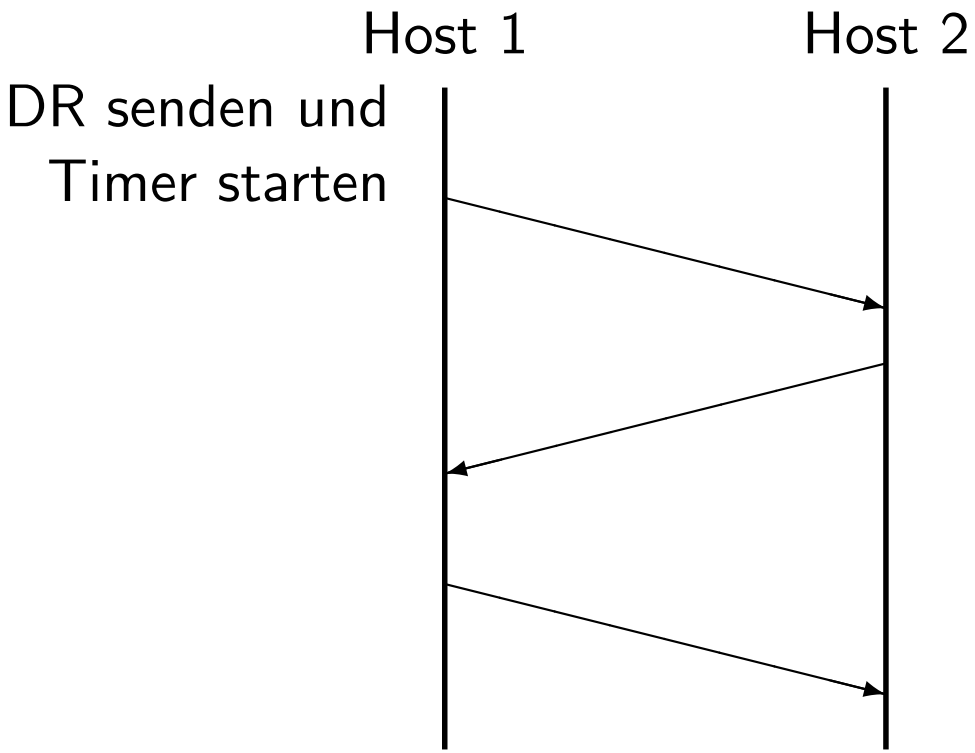


Abbildung 40: Verbindungssabbau ohne Paketverlust

Verbindungssabbau: Beispiel 1

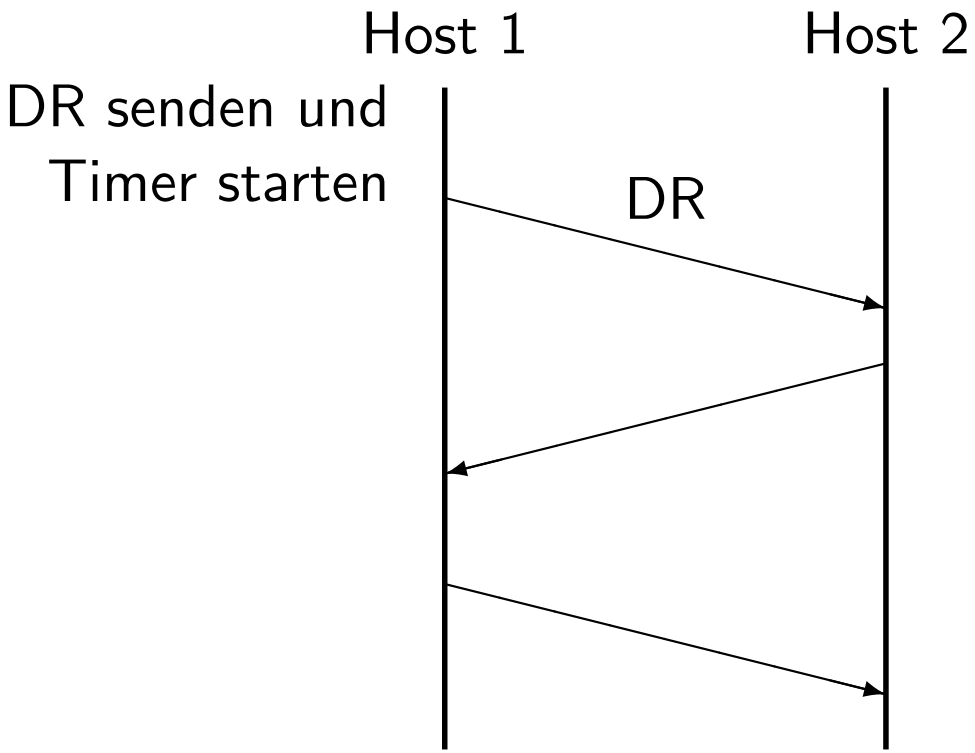


Abbildung 40: Verbindungssabbau ohne Paketverlust

Verbindungssabbau: Beispiel 1

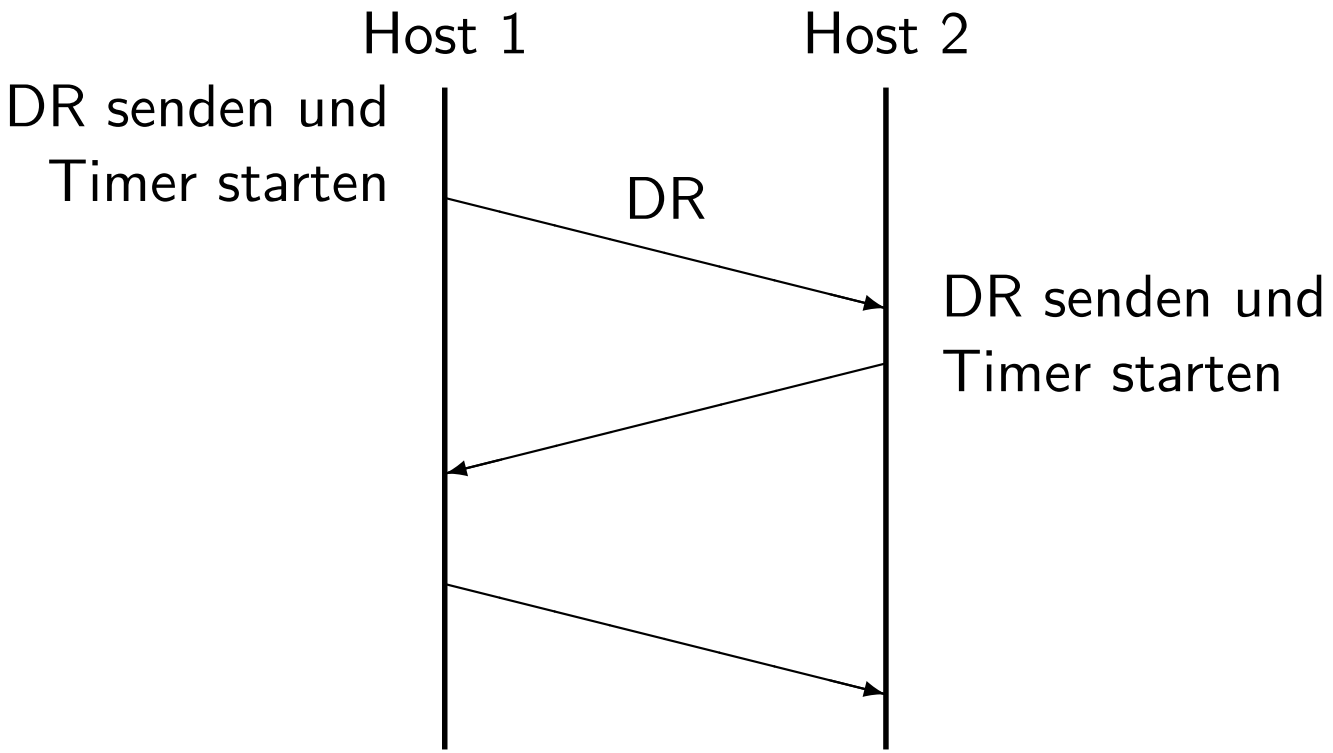


Abbildung 40: Verbindungssabbau ohne Paketverlust

Verbindungssabbau: Beispiel 1

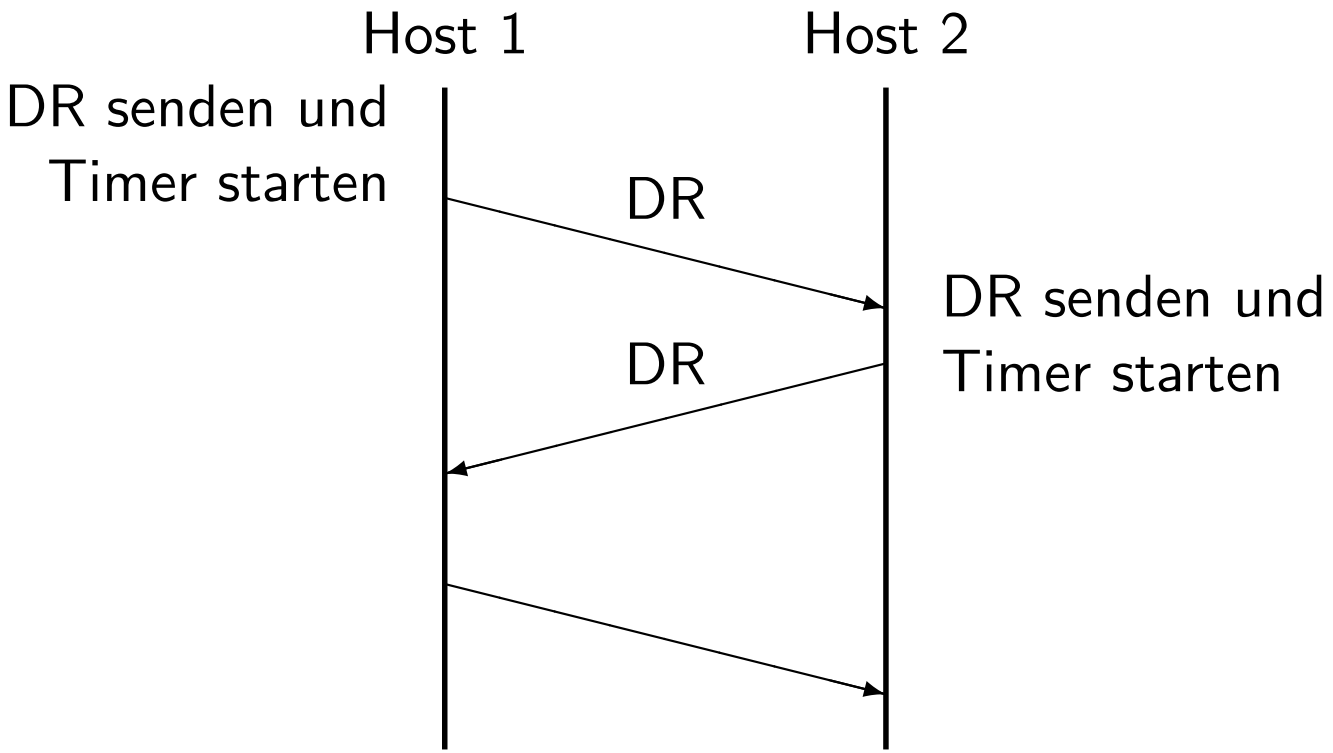


Abbildung 40: Verbindungssabbau ohne Paketverlust

Verbindungssabbau: Beispiel 1

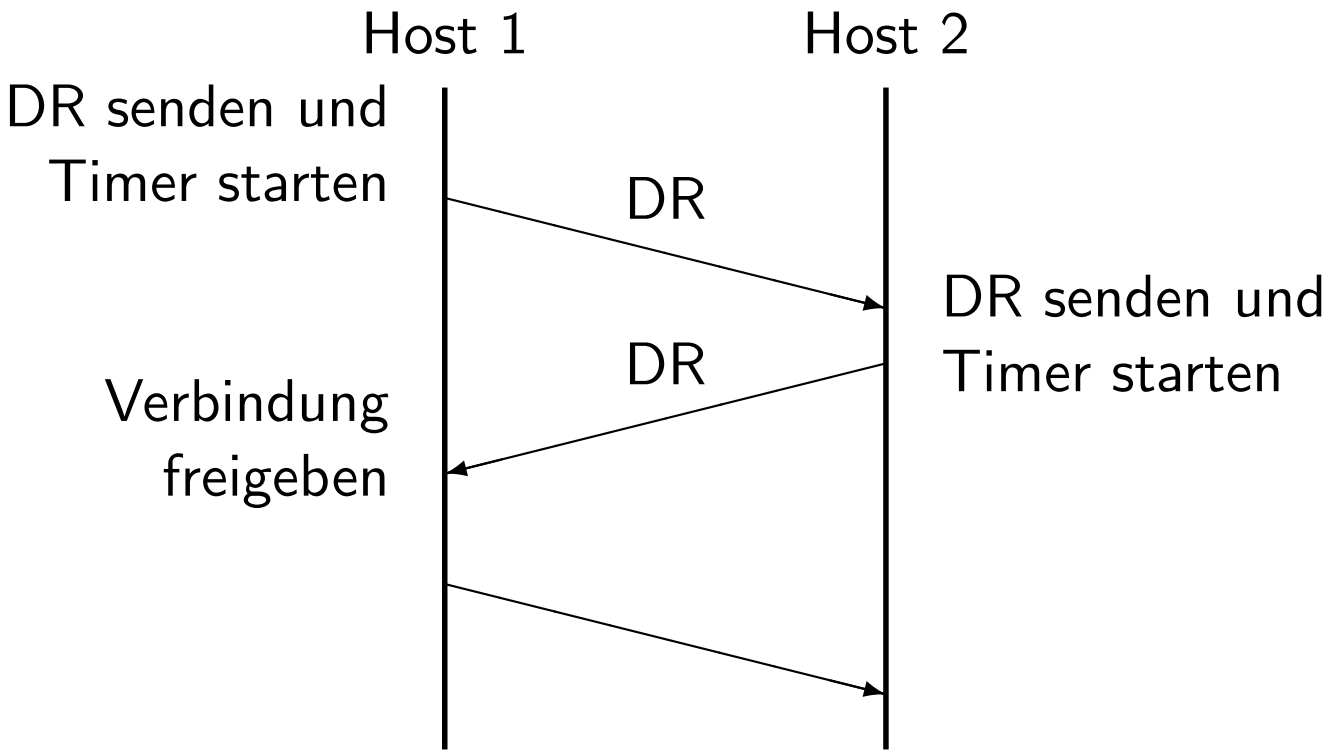


Abbildung 40: Verbindungssabbau ohne Paketverlust

Verbindungssabbau: Beispiel 1

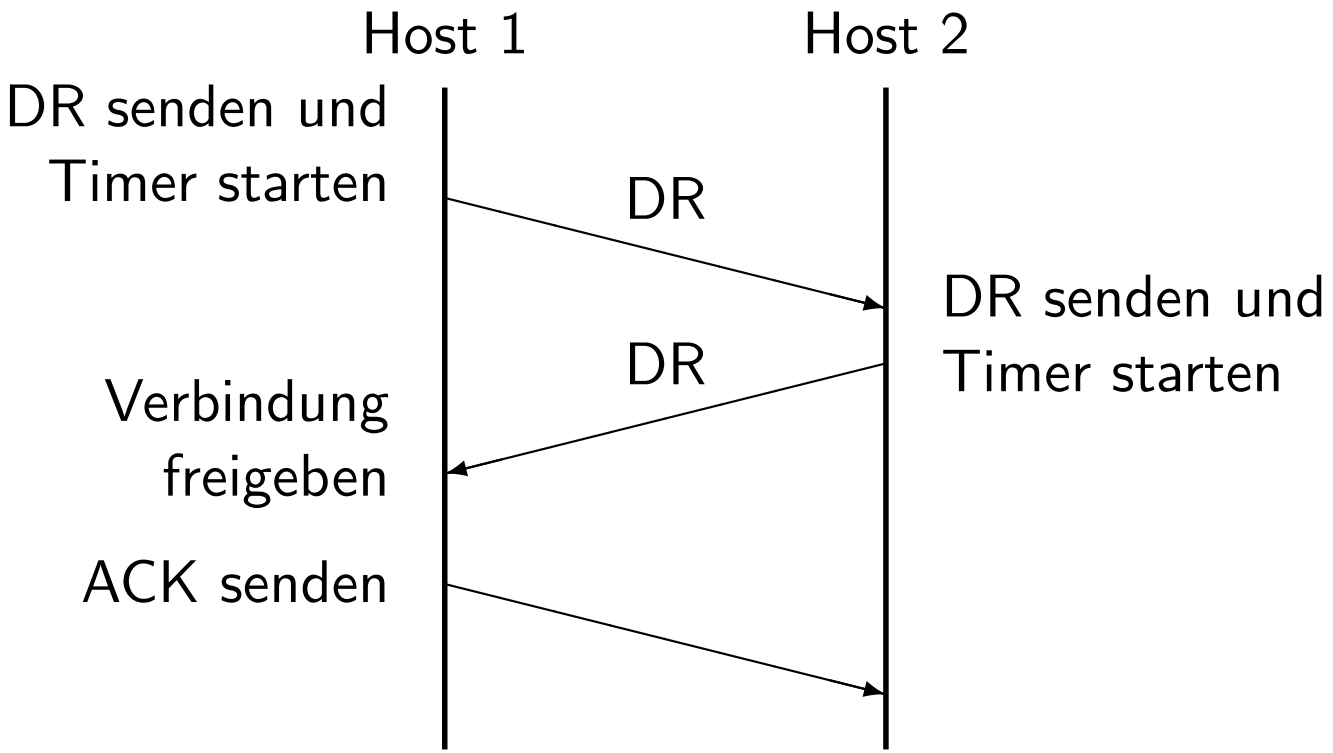


Abbildung 40: Verbindungssabbau ohne Paketverlust

Verbindungssabbau: Beispiel 1

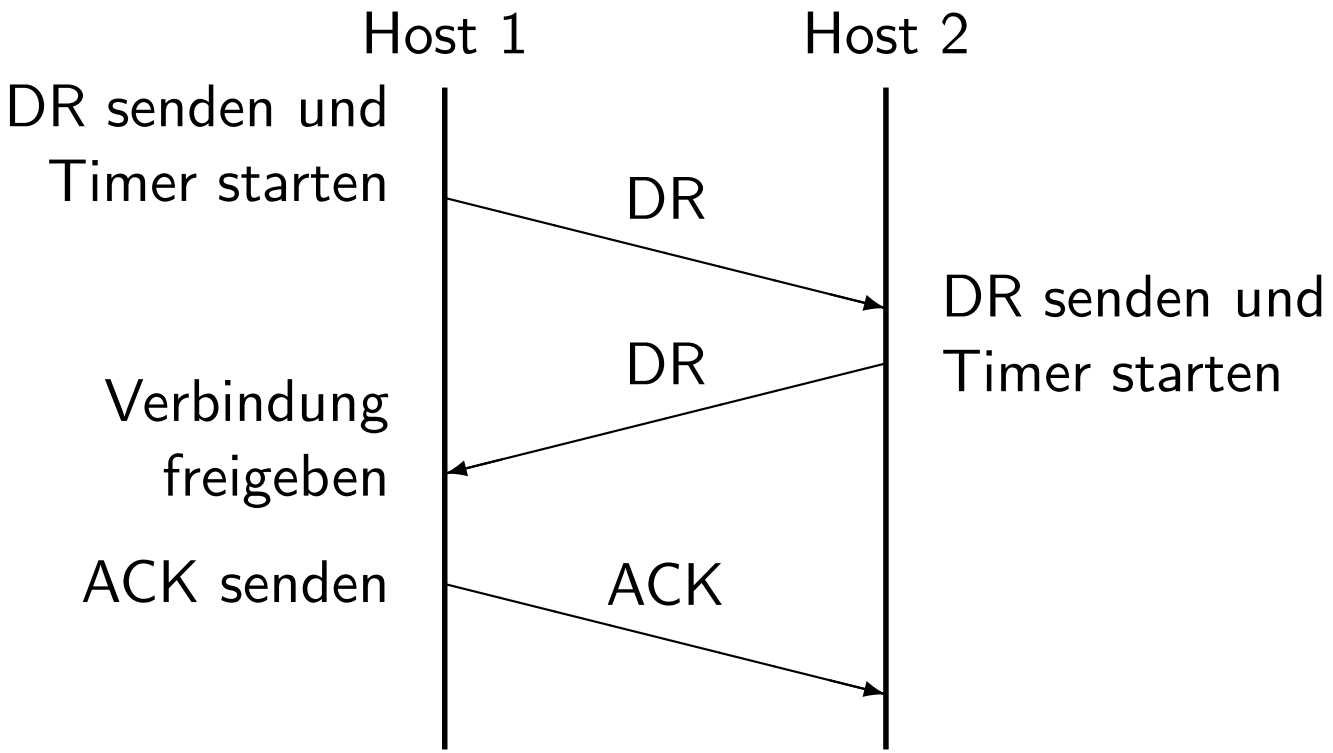


Abbildung 40: Verbindungssabbau ohne Paketverlust

Verbindungssabbau: Beispiel 1

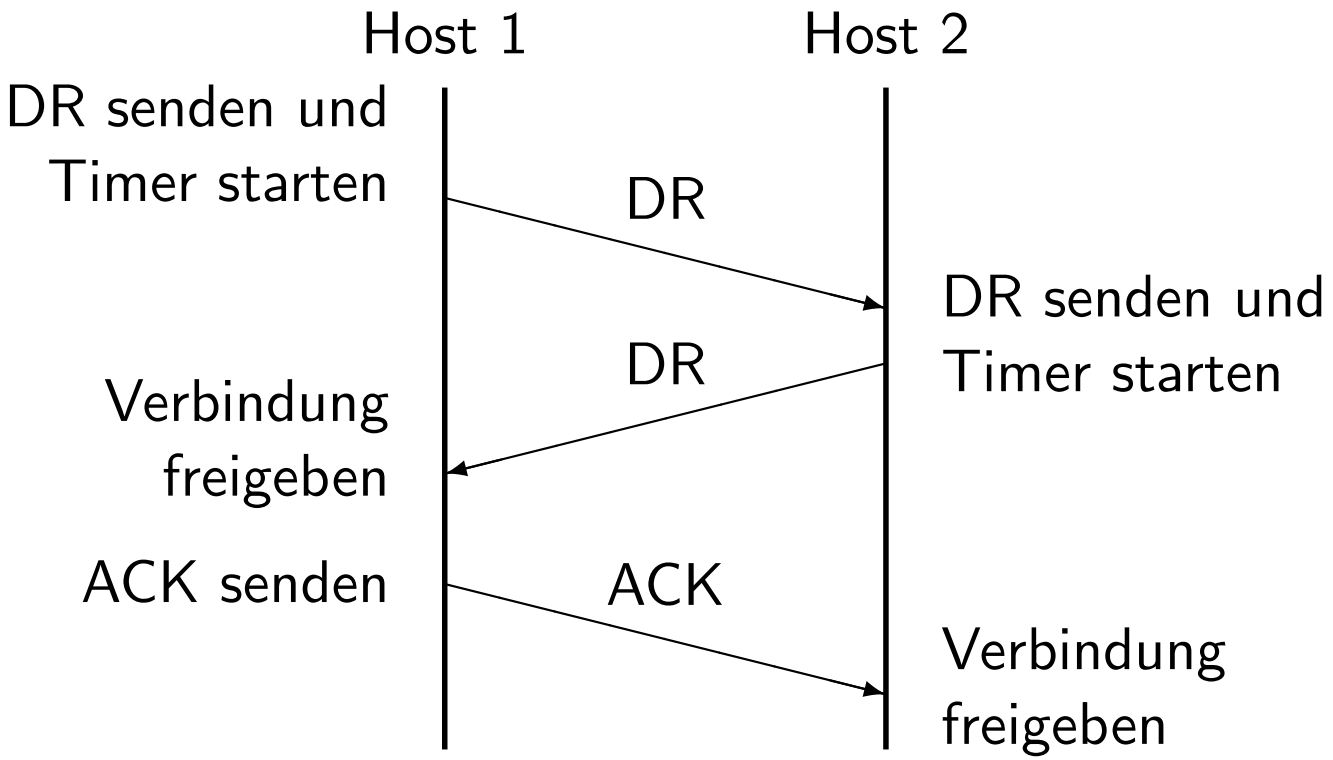


Abbildung 40: Verbindungssabbau ohne Paketverlust

Verbindungssabbau: Beispiel 1

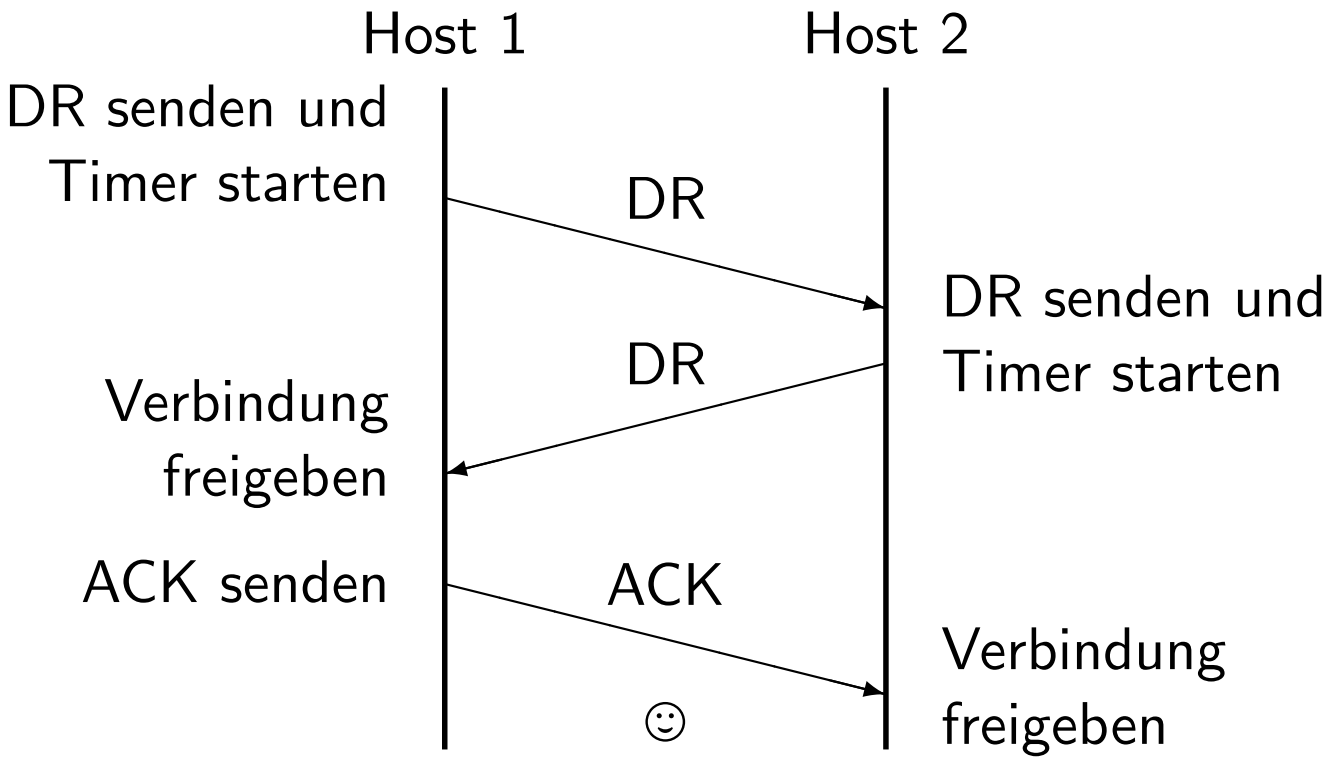


Abbildung 40: Verbindungssabbau ohne Paketverlust

Verbindungssabbau: Beispiel 2

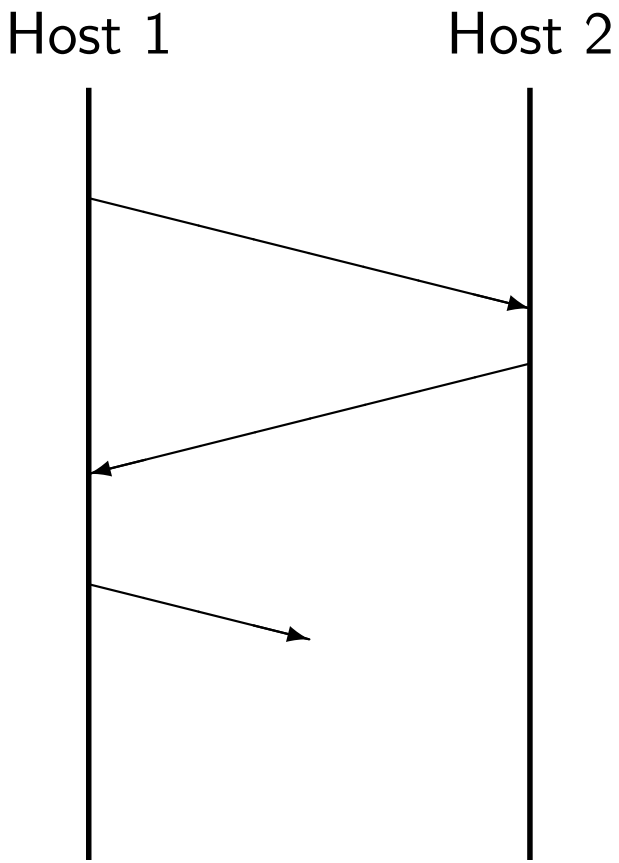


Abbildung 41: Verbindungssabbau bei Paketverlust

Verbindungssabbau: Beispiel 2

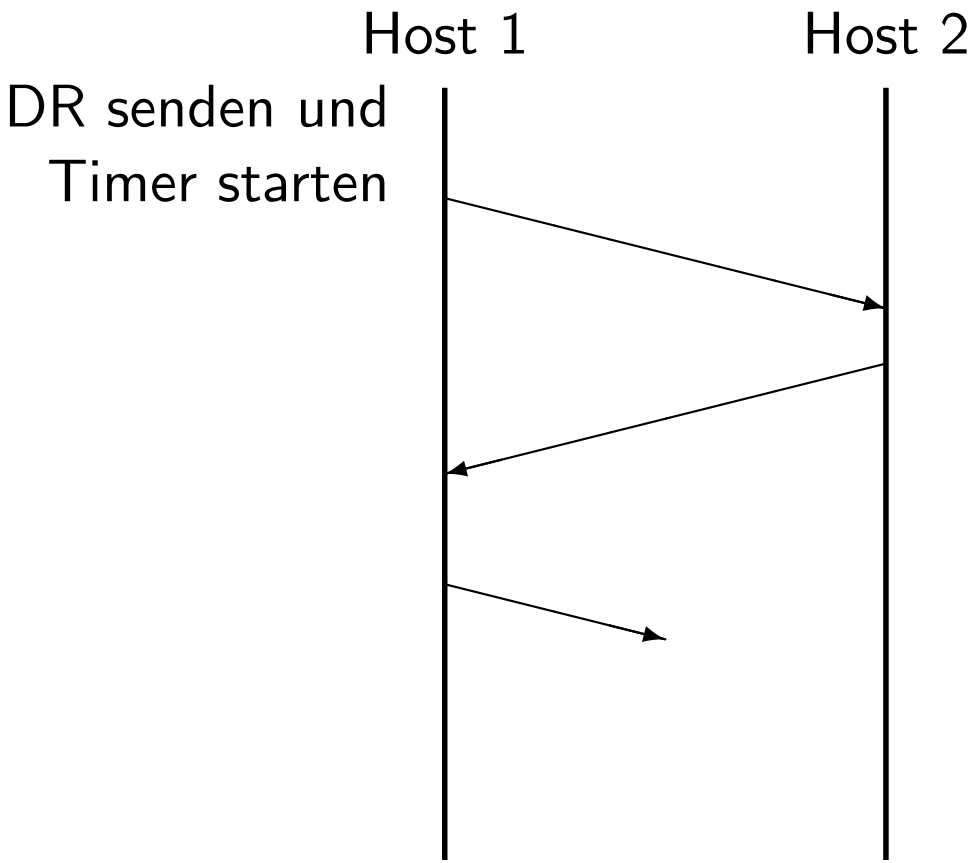


Abbildung 41: Verbindungssabbau bei Paketverlust

Verbindungssabbau: Beispiel 2

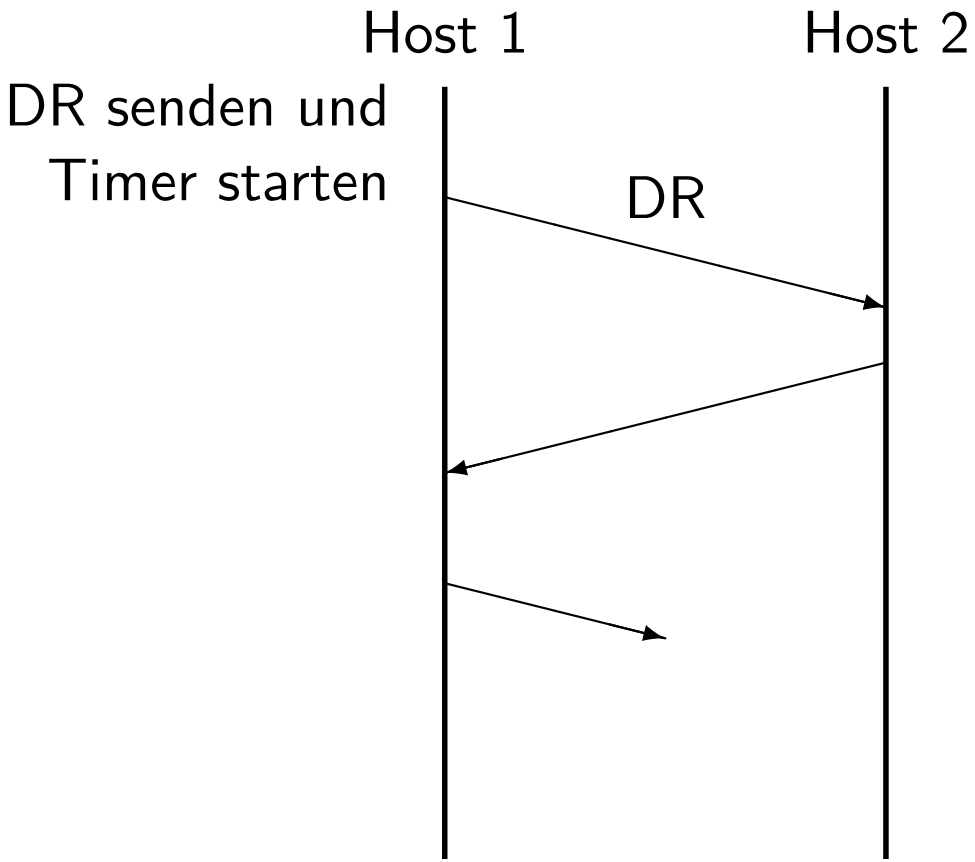


Abbildung 41: Verbindungssabbau bei Paketverlust

Verbindungssabbau: Beispiel 2

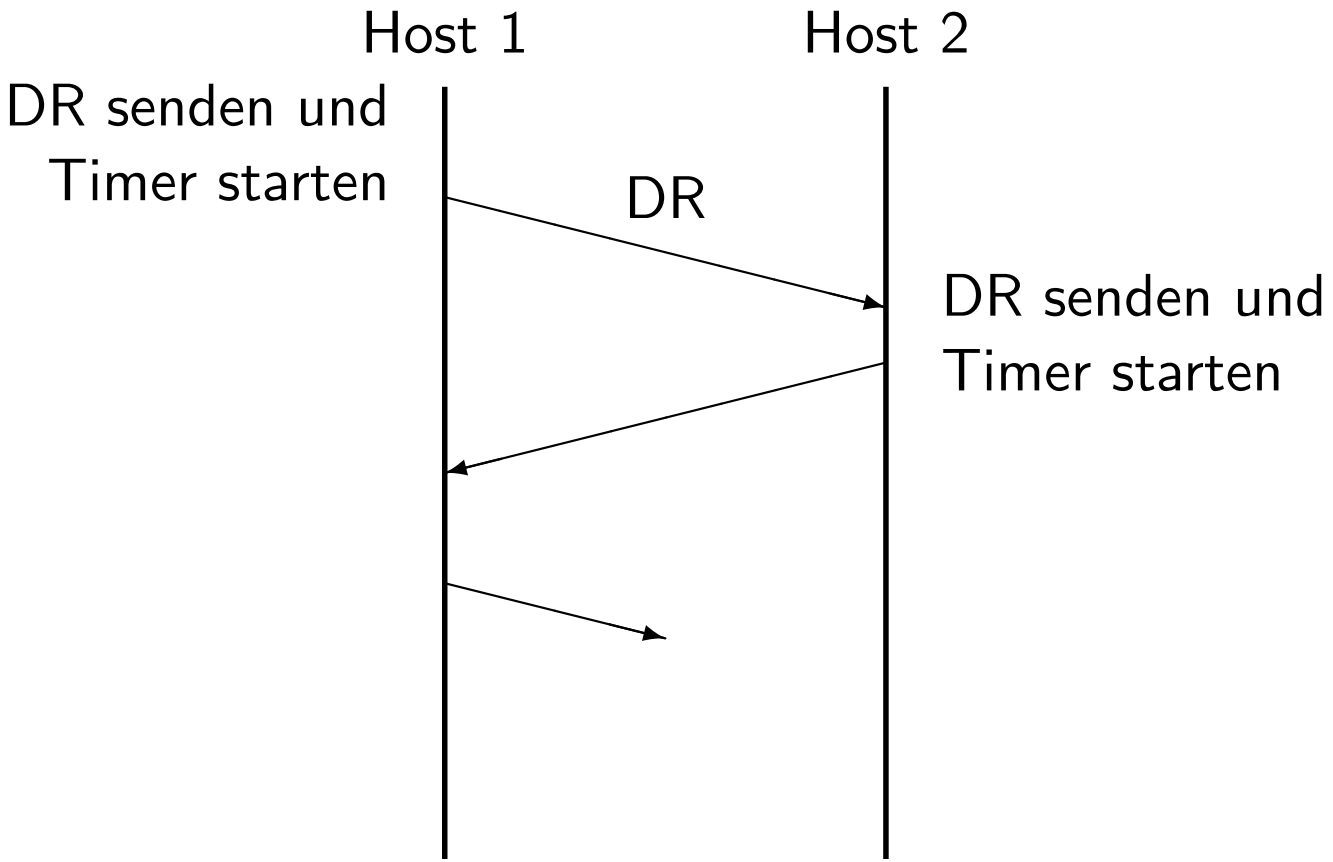


Abbildung 41: Verbindungssabbau bei Paketverlust

Verbindungssabbau: Beispiel 2

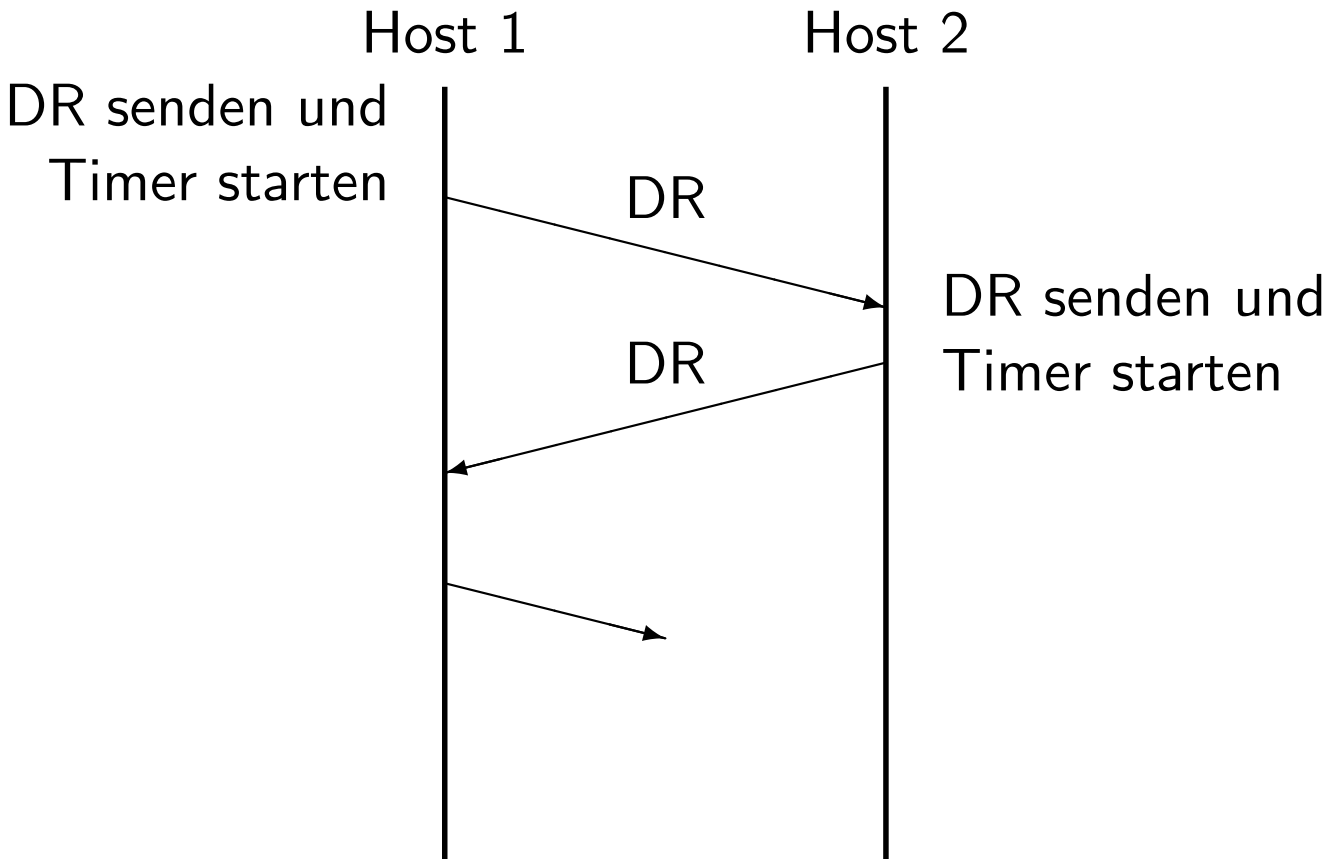


Abbildung 41: Verbindungssabbau bei Paketverlust

Verbindungssabbau: Beispiel 2

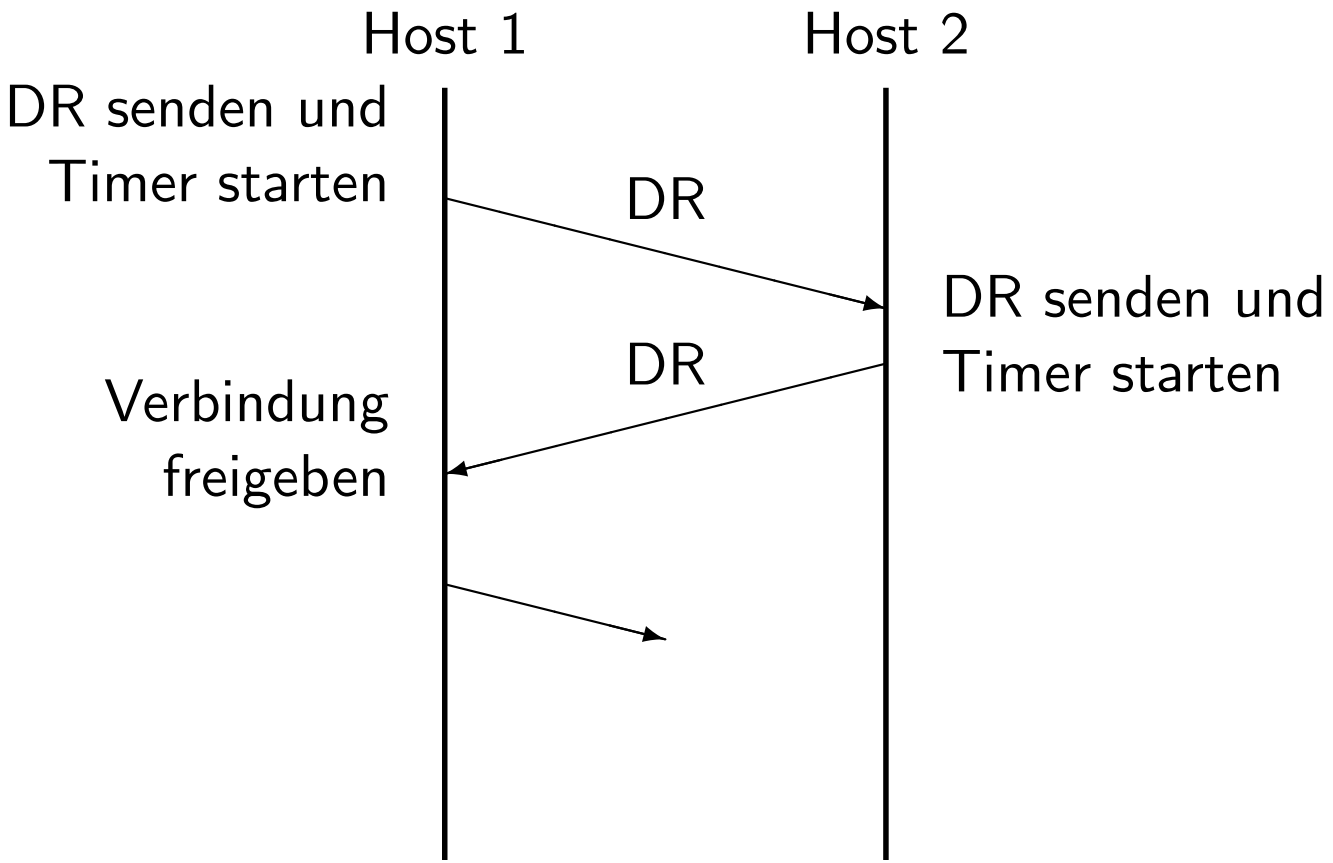


Abbildung 41: Verbindungssabbau bei Paketverlust

Verbindungssabbau: Beispiel 2

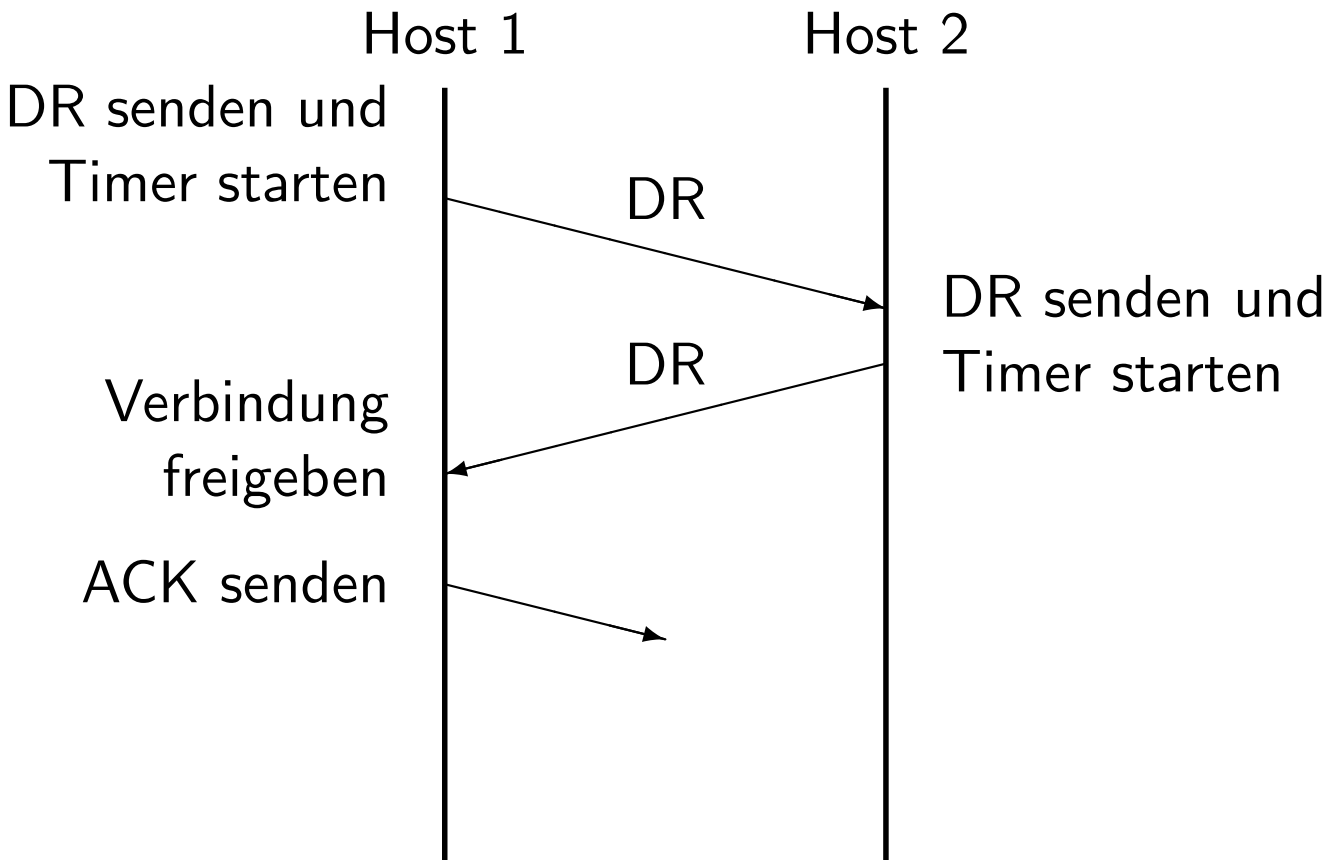


Abbildung 41: Verbindungssabbau bei Paketverlust

Verbindungssabbau: Beispiel 2

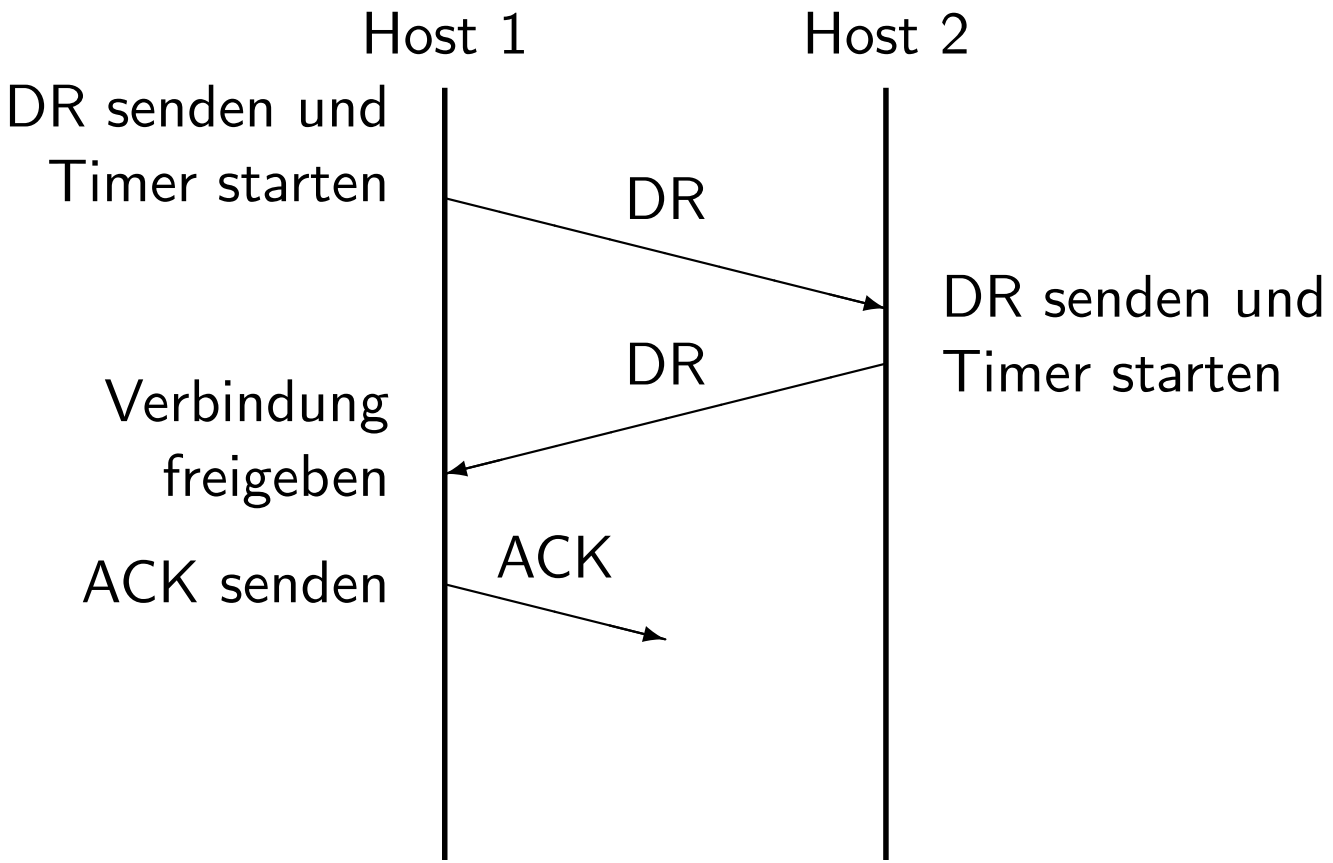


Abbildung 41: Verbindungssabbau bei Paketverlust

Verbindungssabbau: Beispiel 2

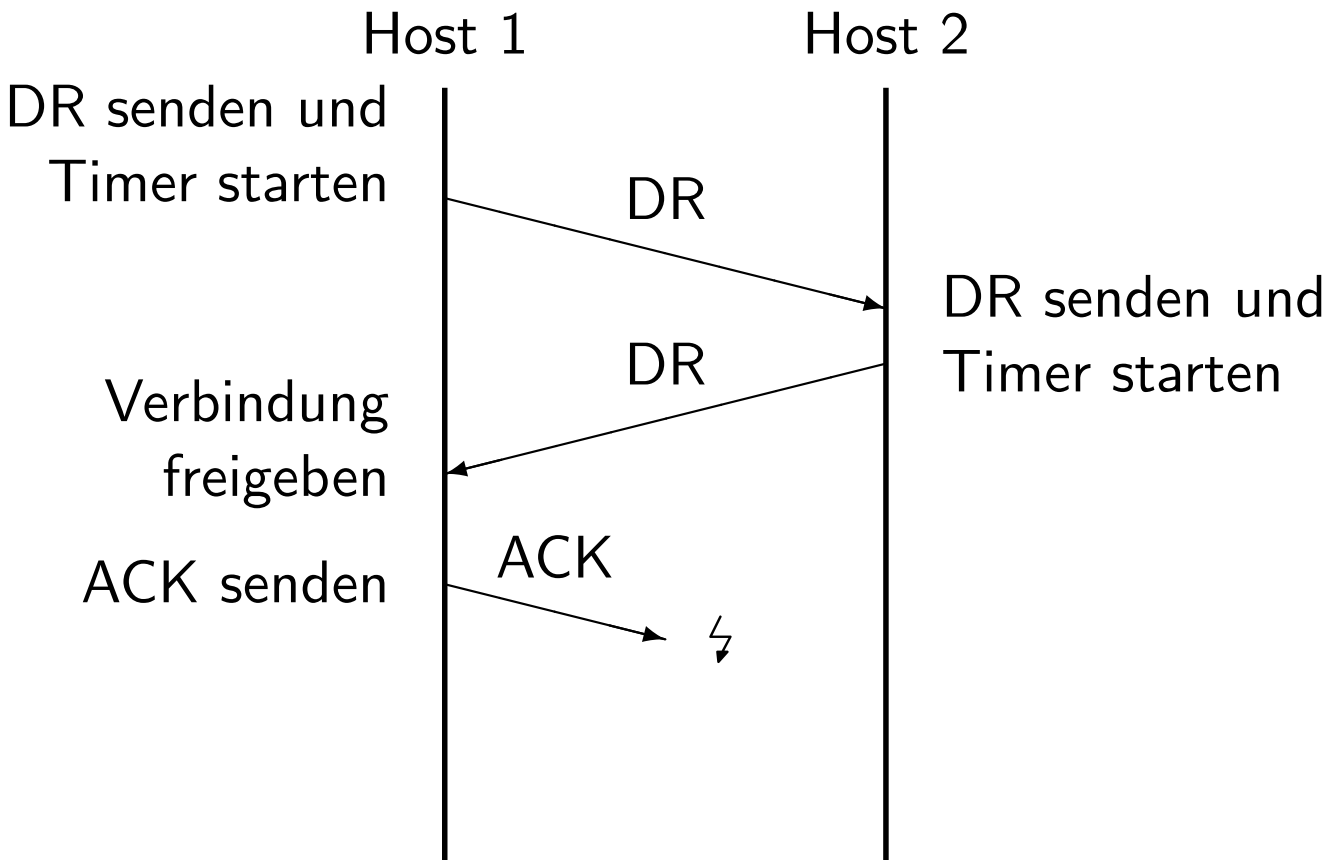


Abbildung 41: Verbindungssabbau bei Paketverlust

Verbindungssabbau: Beispiel 2

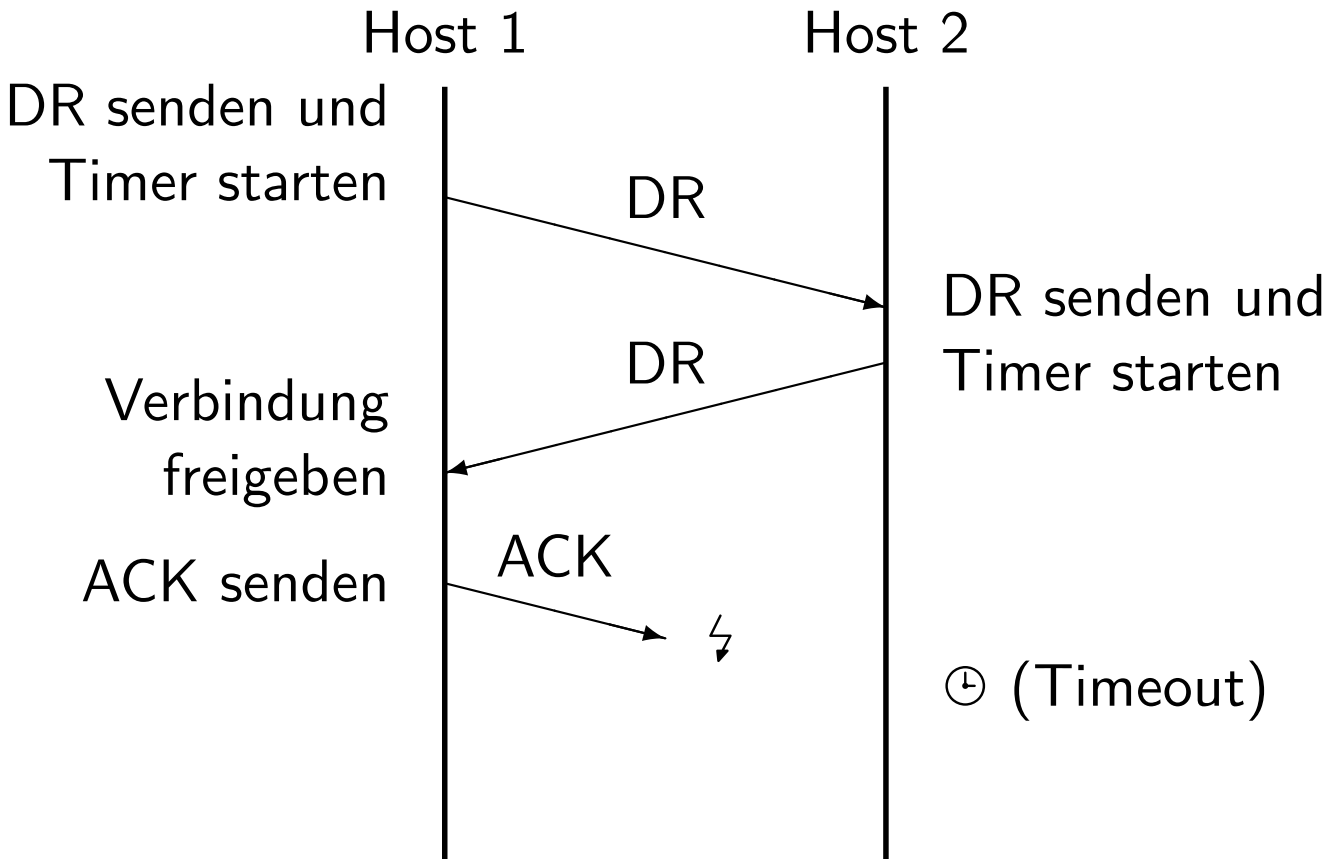


Abbildung 41: Verbindungssabbau bei Paketverlust

Verbindungssabbau: Beispiel 2

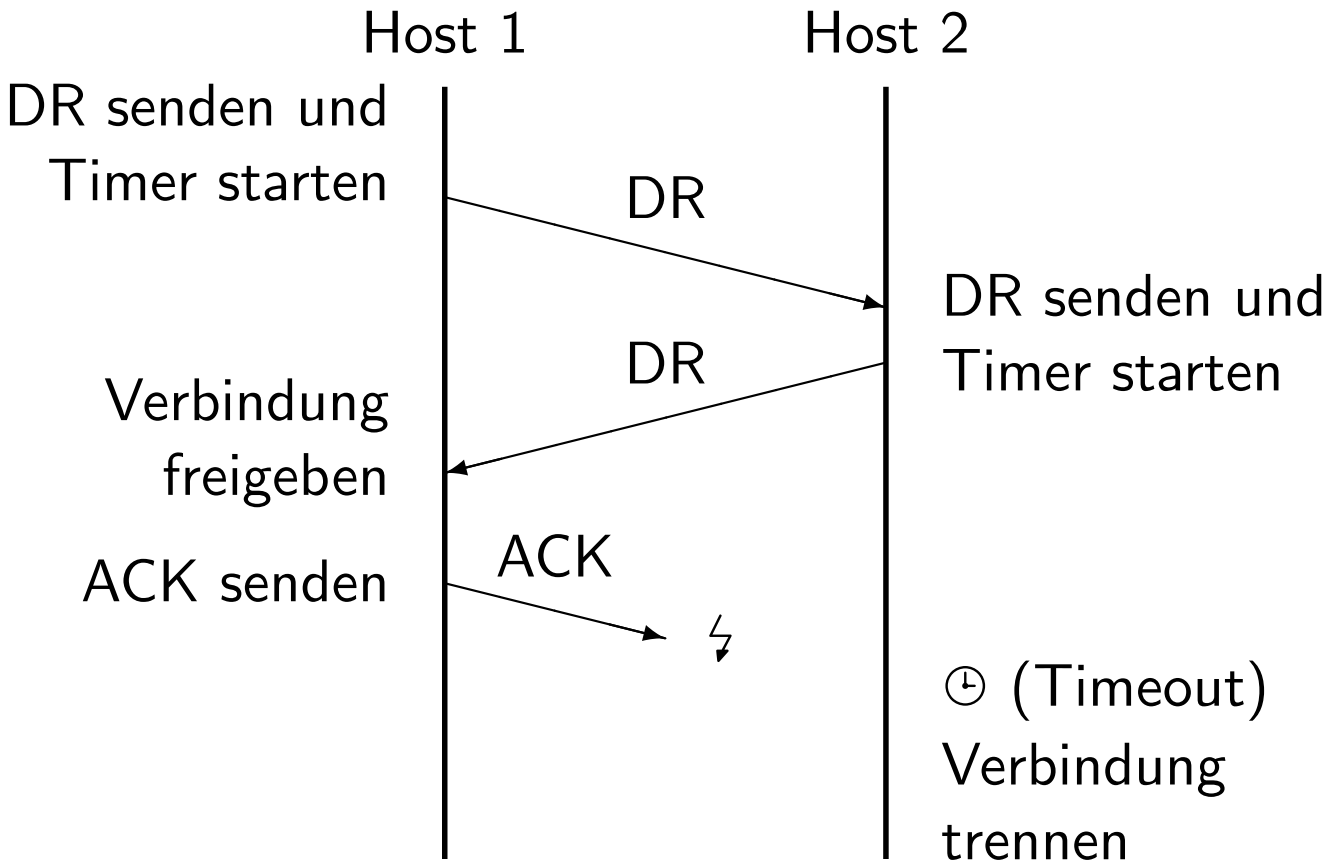


Abbildung 41: Verbindungssabbau bei Paketverlust

Verbindungssabbau: Beispiel 2

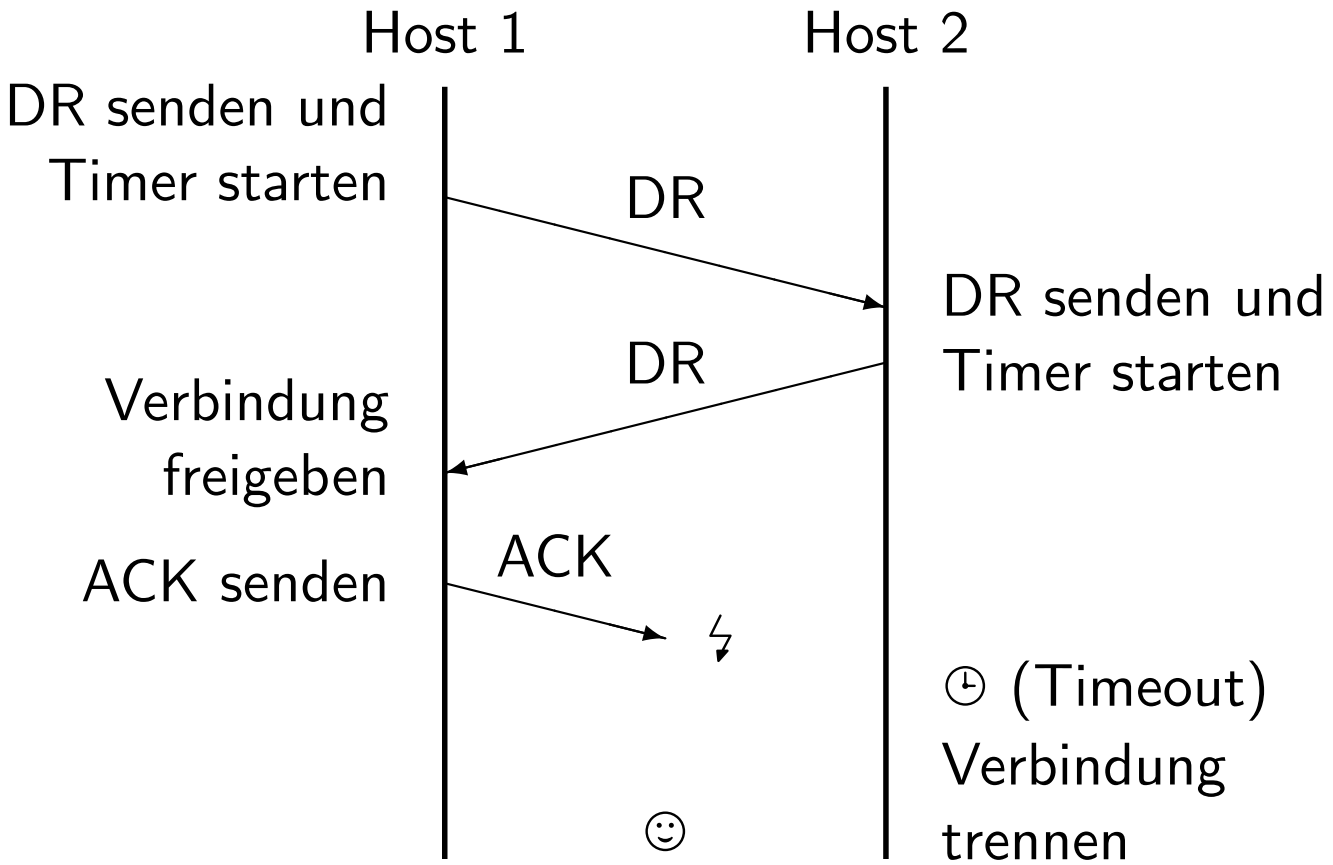


Abbildung 41: Verbindungssabbau bei Paketverlust

Fehlerüberwachung

Wie auf der Sicherungsschicht werden Fehlererkennungscodes verwendet (z. B. einen CRC-Code).

Sequenznummern

Wie auf der Sicherungsschicht werden die Segmente mit Folgenummern versehen und vom Sender noch einmal übertragen, bis er vom vom Empfänger eine Bestätigung über den erfolgreichen Empfang erhält. (Automatic Repeat Request, ARQ)

Schiebefensterprotokoll

Wie auf der Sicherungsschicht kann der Sender bloss eine maximale Anzahl Segmente nacheinander losschicken. Danach dürfen erst dann weitere Segmente gesendet werden, wenn entsprechende Bestätigungen eingegangen sind.

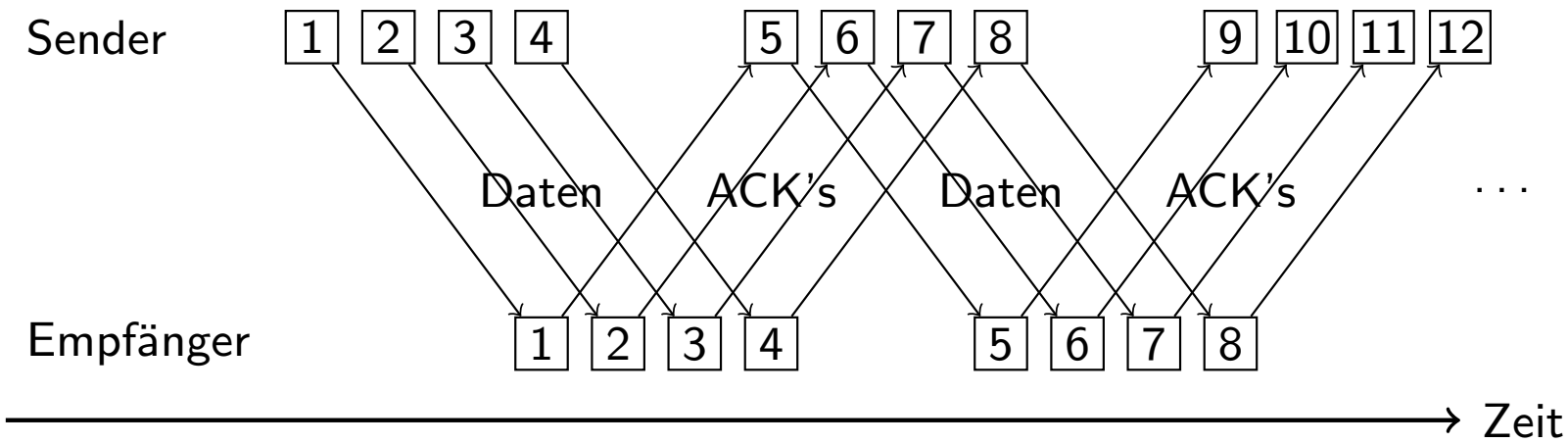


Abbildung 42: Schiebefensterprotokoll mit Fenstergrösse 4

Warum die Doppelspurigkeiten?

- Da die Prozesse der Transportschicht auf den Hosts laufen, können die Hosts die Qualität ihrer Netzwerkverbindungen unabhängig von der Sicherungs- und Vermittlungsschicht verbessern.
- Fehler in Routern werden von der Sicherungsschicht nicht erfasst. (Ende-zu-Ende-Prüfung)
- Die Prüfung auf der Sicherungsschicht entlastet die Transportschicht.

Ziel

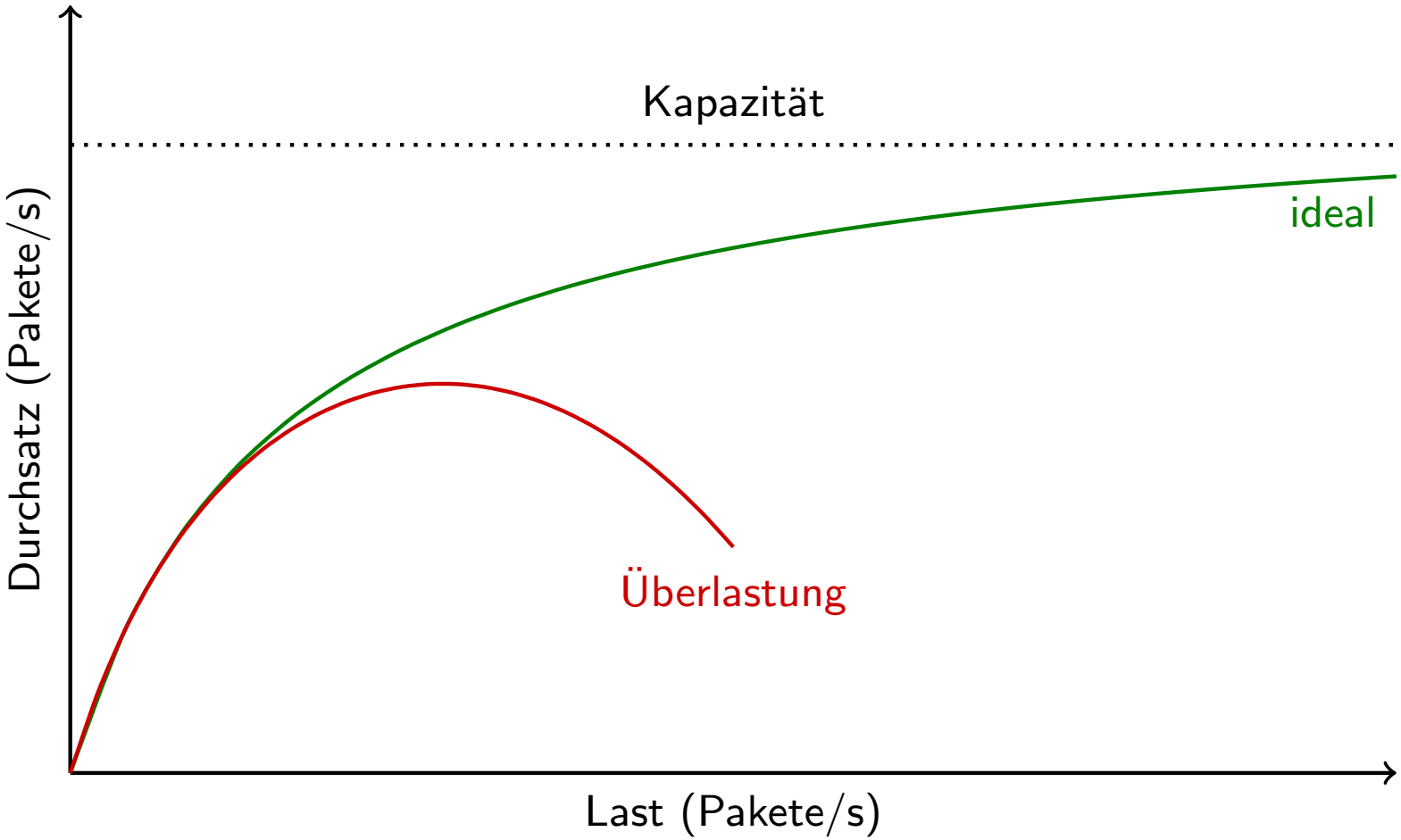


Abbildung 43: Überlastung in Netzwerken

Max-Min-Fairness

„Gleichverteilung“ funktioniert nur bedingt:

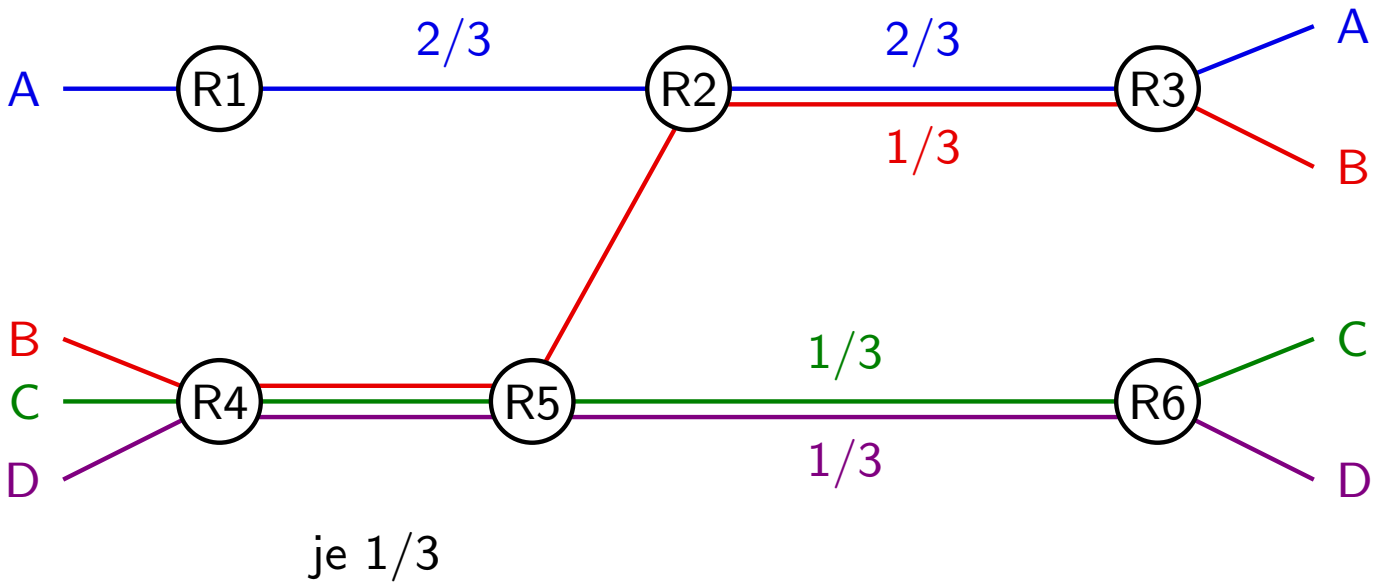


Abbildung 44: Max-Min-Zuordnung

Max-Min-Fairness: Jeder Versuch, einer der Verbindungen mehr Bandbreite zu geben führt dazu, dass eine andere Verbindung weniger Bandbreite erhält

Überlastung

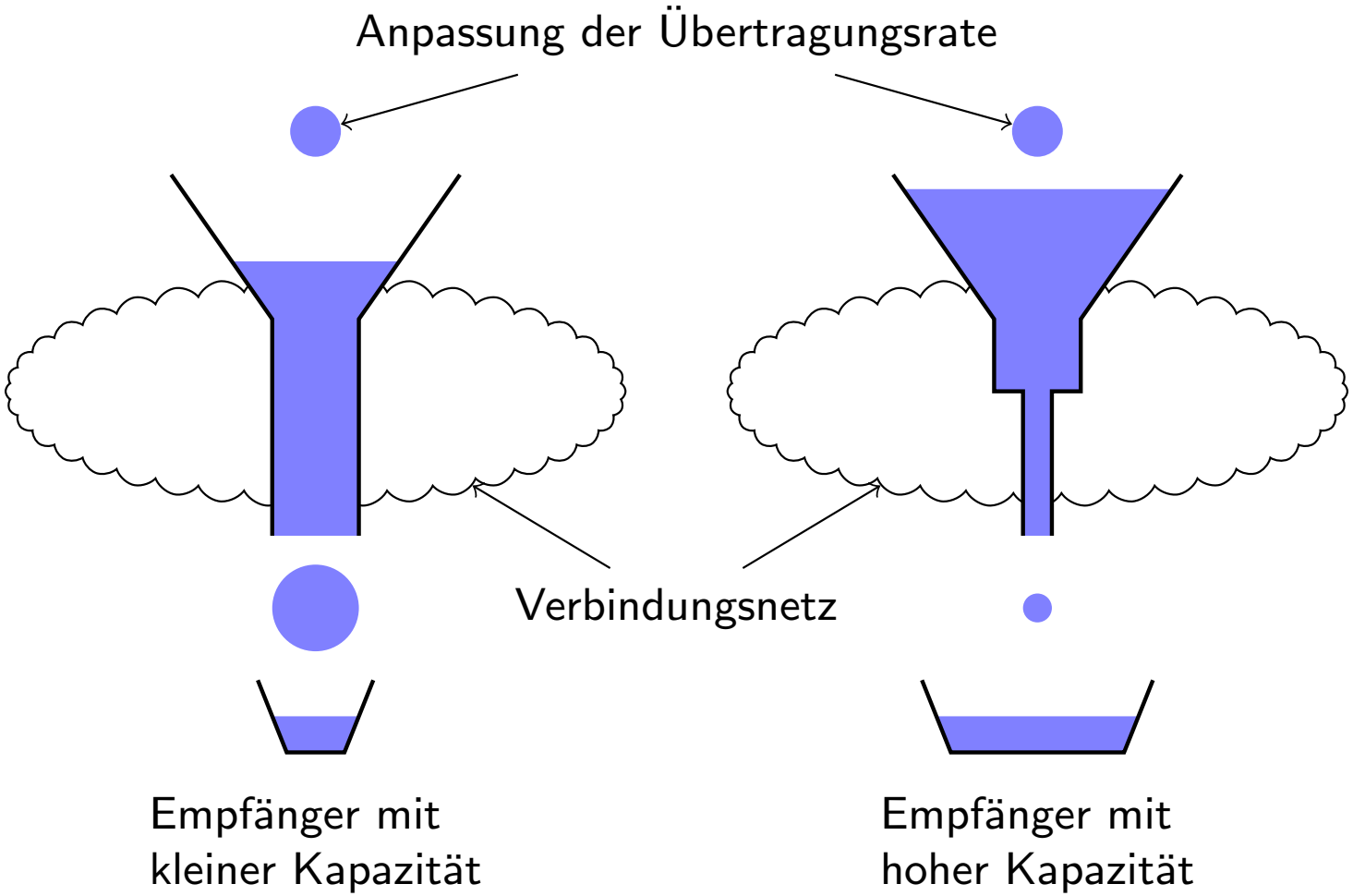


Abbildung 45: Fluss- und Überlastungskontrolle

Was ist UDP?

Das User Datagram Protocol (UDP) ist ein verbindungsloses Transportprotokoll, das beispielsweise in folgenden Situationen eingesetzte wird:

- Client-Server-Situationen
- DNS (Domain Name System)
- Multimedia-/Echtzeitanwendungen (z. B. RTP)

UDP-Header

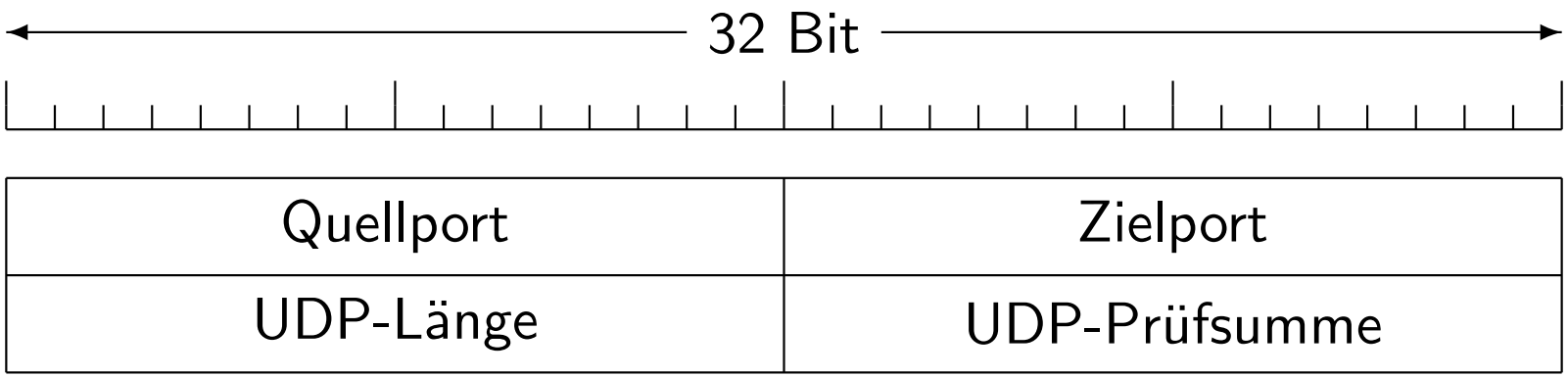


Abbildung 46: UDP-Header

UDP-Länge: 8-Byte-Header + Daten

Prüfsumme: optional

Was UDP nicht kann

- Flusskontrolle
- Fehlerkontrolle
- erneute Übertragung bei Fehlern

Was ist TCP?

TCP steht für Transmission Control Protocol

- zuverlässiger, verbindungsorientierter Dienst
- Host-zu-Host-Bytestrom (Einhaltung der Datenreihenfolge)
- Datenpufferung beim Sender und Empfänger
- Daten werden erst gesendet, wenn ein Segment gefüllt ist oder die Daten „überfällig“ sind. Mit dem PSH-Flag kann ein sofortiger Versand erzwungen werden.
- Jedes gesendete Byte wird mit einer Sequenznummer gezählt.
- Die gesendeten Bytes werden quittiert und die Quittungen enthalten die Sequenznummer des Bytes, das als nächstes erwartet wird.

TCP-Header

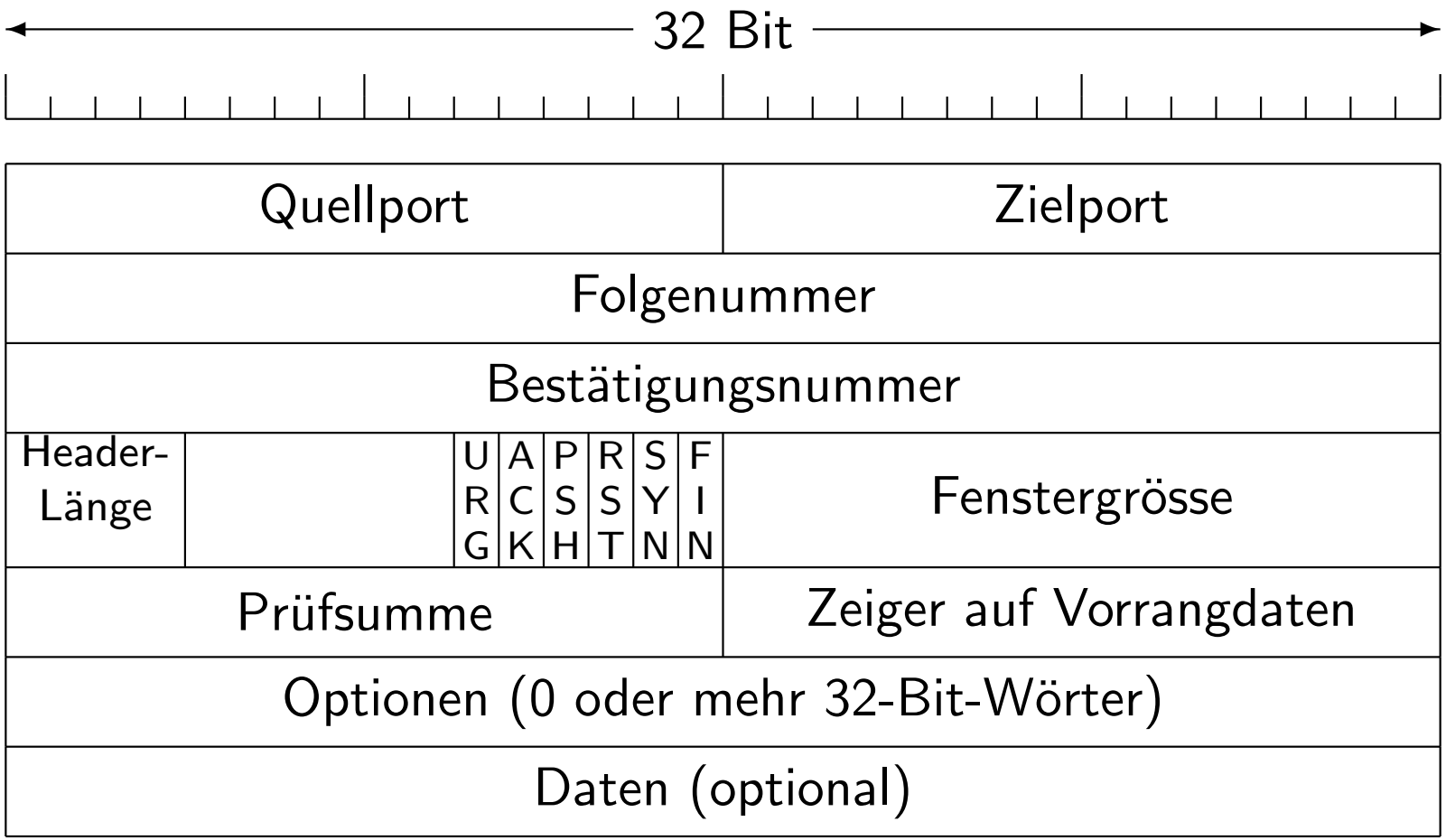


Abbildung 47: TCP-Header

Felder im TCP-Header (1)

- Quell- und Zielport: 16-Bit Prozess-ID
- Folgenummer: Folgenummer des ersten Bytes in den Daten. Wird mit dem SYN-Flag zurückgesetzt.
- Bestätigungsnummer: Nummer des nächsten erwarteten Bytes. (Nur mit ACK-Flag gültig.)
- Header-Länge: Anzahl der 32-Bit Worte im Header (Beginn der Daten)
- URG: Urgent-Flag
- PSH: Push-Flag (Versand erzwingen)

Felder im TCP-Header (Fortsetzung)

- RST: Reset-Flag (aktuelle Verbindung zurücksetzen)
- SYN: Flag für den Verbindungsaufbau und die Synchronisation der Sequenznummer
- ACK: Flag für den Verbindungsaufbau und die Anzeige, dass die Bestätigungsnummer gültig ist.
- FIN: Zeigt an, dass alle Daten gesendet wurden.
- Window-Size: Fenstergrösse fürs Schiebefensterprotokoll
- Prüfsumme: optional (in IPv6 obligatorisch)
- Zeiger auf Vorrangdaten: Falls das URG-Flag gesetzt ist, wird die Position im Datenfeld angegeben, wo Vorrangdaten stehen.

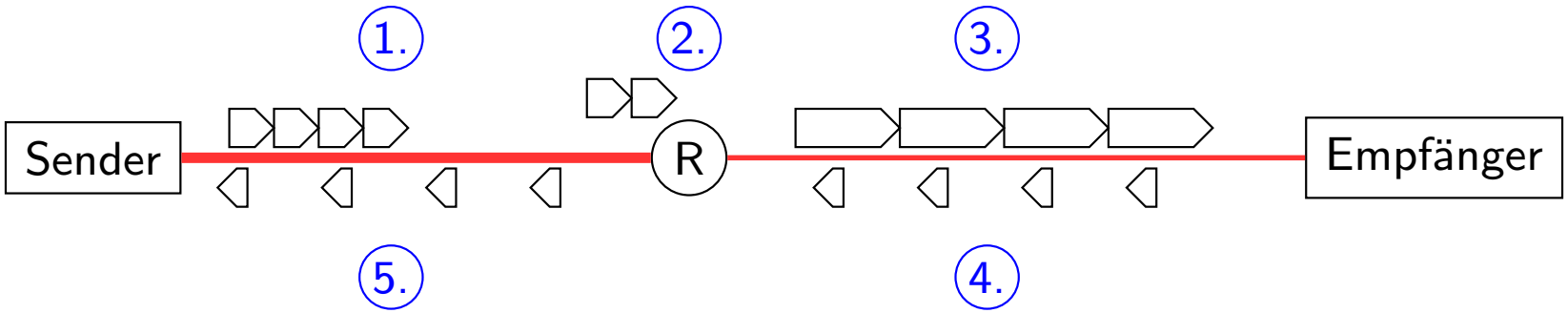
Das Überlastungsfenster

Das TCP-Protokoll verwaltet ein *Überlastungsfenster* (*congestion window*. Die Grösse des Überlastungsfensters ist die Anzahl der Bytes, die der Sender im Netz haben kann.

Zur Erinnerung: Der Sender verwaltet auch ein *Flusskontrollfenster*, das die Anzahl der Bytes angibt, die der Empfänger puffern kann.

Bei der Entscheidung, wie viele Daten ins Netz gesendet werden können, wählt TCP das jeweils kleinere Fenster.

ACK-Taktung

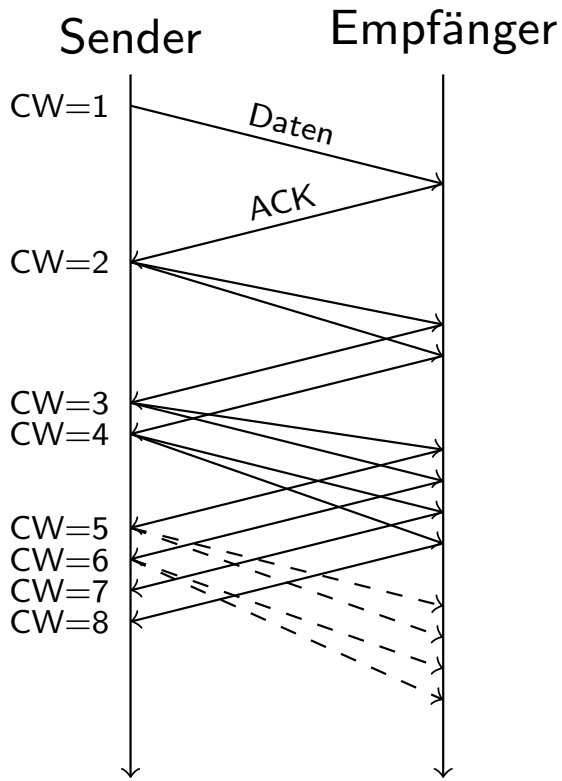


ACK-Taktung

1. Paketschübe auf der schnellen Verbindung
2. Stau beim Router vor der langsamen Verbindung
3. Pakete werden auf der langsamen Verbindung „gedehnt“.
4. ACK’s in gleichen Abständen wie beim Empfang
5. ACK’s bewahren die Abstände auf der schnellen Verbindung.

Die Abstände der ACK-Segmente erlauben Rückschlüsse auf die langsamste Verbindung des Datenpfads.

Slow-Start-Algorithmus



Zu Beginn wird eine hohe Schwelle für das Congestion Window (CW) gewählt.

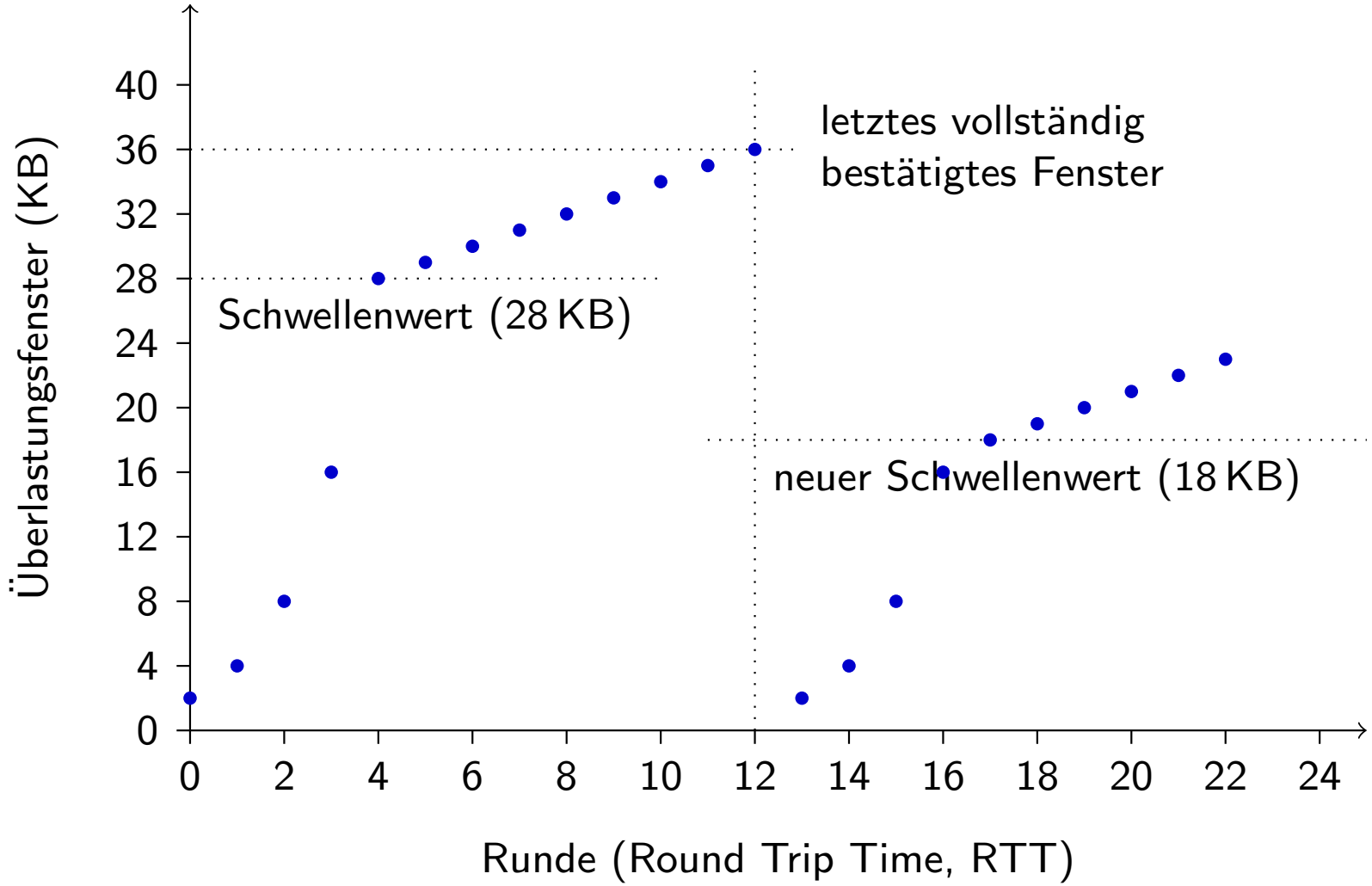
Für jedes bestätigte Segment dürfen zwei weitere Segmente gesendet werden.

Kommen Bestätigungen nicht mehr rechtzeitig an, wird die Schwelle halbiert und Slow-Start beginnt neu.

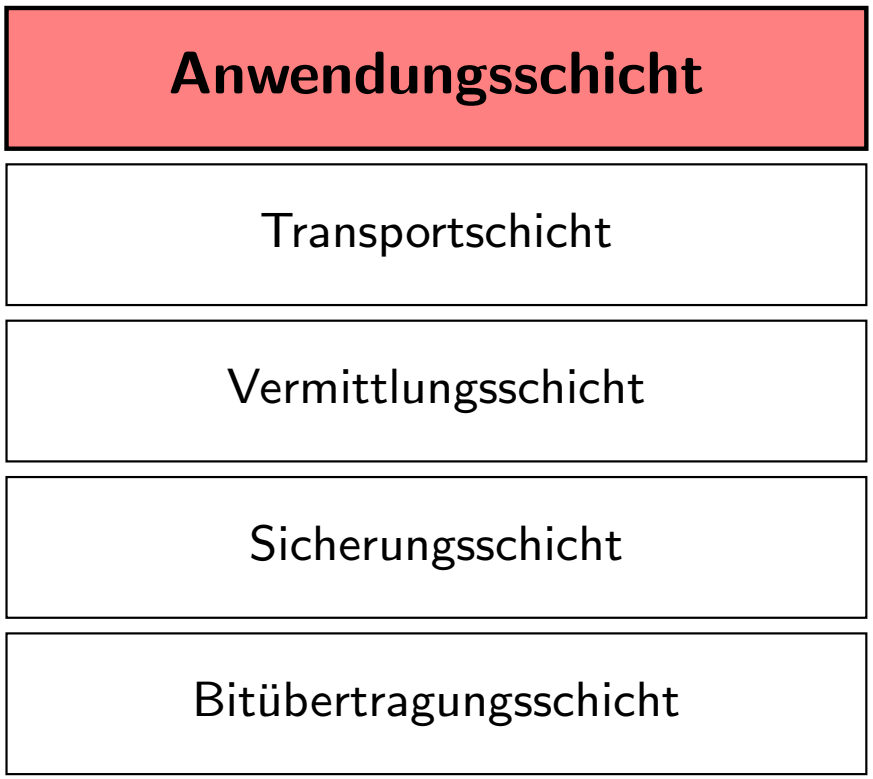
Slow-Start ist überhaupt nicht langsam!

Abbildung 48: Der Slow-Start-Algorithmus

TCP Tahoe


Überlastungsfenster (KB)
0
4
8
12
16
20
24
28
32
36
40
0
2
4
6
8
10
12
14
16
18
20
22
24
Runde (Round Trip Time, RTT)
letztes vollständig
bestätigtes Fenster
Schwellenwert (28 KB)
neuer Schwellenwert (18 KB)

Standortbestimmung



Die Technologien der bisher behandelten Schichten ermöglichen es, Daten in einem Computernetzwerk – insbesondere im Internet – von einem Host zu einem anderen zu befördern.

In diesem Kapitel besprechen wir zwei Klassen von Anwendungen, die auf diesen Schichten aufsetzen. Zuvor müssen wir uns mit aber dem Domain Name Service (DNS) befassen, das diese Anwendungen unterstützt.

Das Ausgangsproblem

Die Adressierung im Internet erfolgt über IPv4- oder IPv6-Nummern.

Das Ausgangsproblem

Die Adressierung im Internet erfolgt über IPv4- oder IPv6-Nummern.

Menschen können (und wollen) sich keine langen Ziffernfolgen merken.

Das Ausgangsproblem

Die Adressierung im Internet erfolgt über IPv4- oder IPv6-Nummern.

Menschen können (und wollen) sich keine langen Ziffernfolgen merken.

Anstelle der IP-Adressen werden symbolische Namen wie `www.kollegistans.ch` verwendet.

Vorteile der symbolischen Namen

- Rechner können ihre IP-Adressen wechseln, bleiben aber unter demselben Domänennamen erreichbar.
- Ein Domänenname kann über mehrere IP-Adressen erreicht werden. („Lastverteilung“)
- Mehrere Domänennamen können sich eine IP-Adresse „teilen“.

Historisches

In der Frühzeit des Internets gab es auf jedem Host eine Datei, mit der Namensinformation jedes anderen Rechners im Internet.

Historisches

In der Frühzeit des Internets gab es auf jedem Host eine Datei, mit der Namensinformation jedes anderen Rechners im Internet.

Als das Internet immer grösser wurde, musste man nach einer neuen Lösung suchen. Warum?

Historisches

In der Frühzeit des Internets gab es auf jedem Host eine Datei, mit der Namensinformation jedes anderen Rechners im Internet.

Als das Internet immer grösser wurde, musste man nach einer neuen Lösung suchen. Warum?

Ab 1983 wurde mit der Entwicklung des DNS begonnen.

Was ist das DNS?

- Ein hierarchisches, auf sogenannten Domänen basierendes Benennungschema.
- Ein verteiltes Datenbanksystem zur Implementierung dieses Benennungsschemas.

DNS-Namensraum

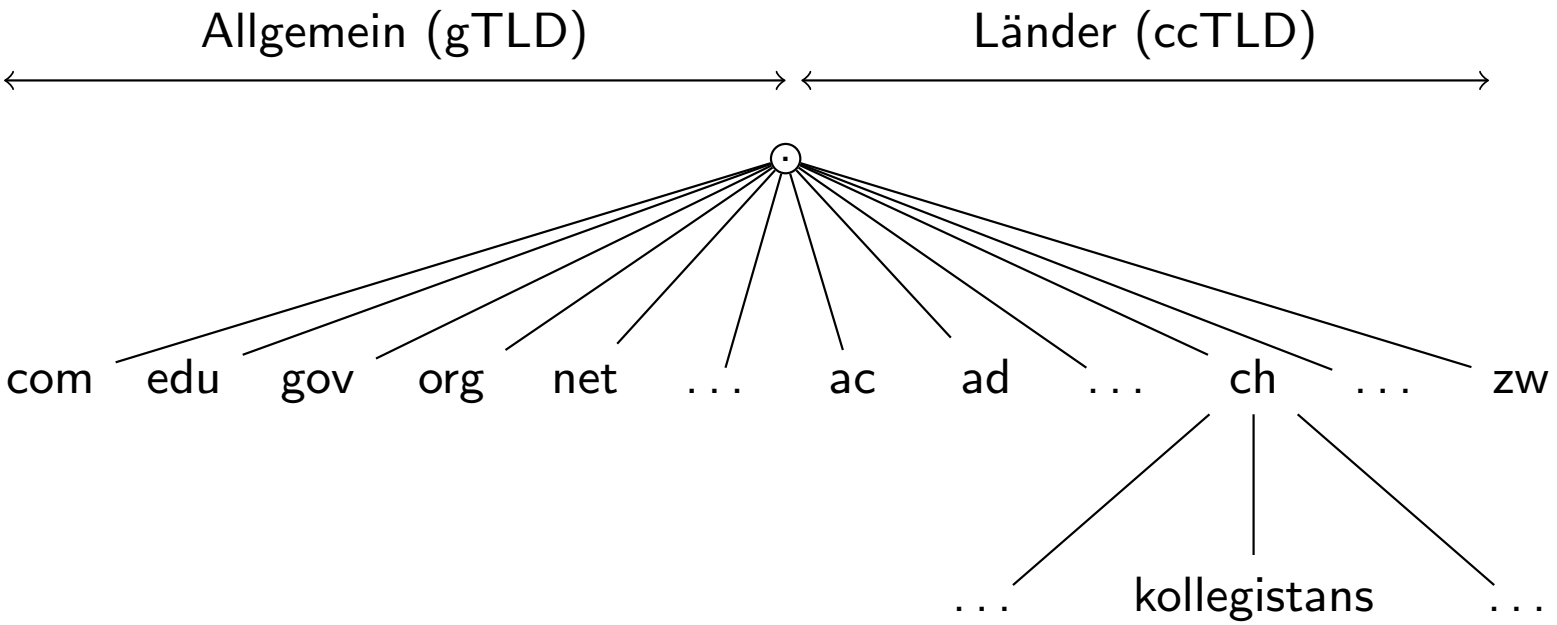


Abbildung 50: TLDs

Zuoberst befindet sich der Wurzelknoten. Er wird durch einen Punkt dargestellt. An diesem Punkt sind die Top Level Domains (TLDs) angehängt. Daran können weitere Subdomänen angehängt

Domänennamen

- aufwärts bis zur Wurzel, durch Punkte getrennt: `kollegistans.ch.`, `inf.ethz.ch.`
- Gross- und Kleinschreibung wird nicht beachtet.
- Jede Komponente kann bis zu 63 Zeichen haben.
- Der vollständige Pfad kann bis zu 255 Zeichen haben.

Ressourcendatensätze

Mit jedem Domänennamen können mehrere Ressourcendatensätze verbunden werden. Ein solcher Datensatz besteht aus 5 Einträgen

- `Domain_name`: Domäne, für die der Datensatz gilt.
- `Time_to_live`: Dauerhaftigkeit der Information in Sekunden
- `Class`: Für Internetinformation ist dies immer `IN`
- `Type`: Art des Datensatzes (siehe unten)
- `Value`: Je nach Typ eine Zahl, ein Domänenname oder ein ASCII-String

Typen von Ressourcendatensätze

Hier eine Auswahl:

Typ	**Bedeutung**	**Wert**
SOA	Start of Authority	Zonenparameter
A	IPv4-Adresse eines Hosts	32-Bit-Ganzzahl
AAAA	IPv6-Adresse eines Hosts	128-Bit-Ganzzahl
MX	Mail Exchange	Mailserver
NS	Nameserver	Servername der Domäne
CNAME	Kanonischer Name	ein Alias für den Wert

Tabelle 2: Ressourcendatensatz-Typen (Auswahl)

Nameserver

Der DNS-Namensraum ist in nicht überlappende Zonen aufgeteilt.

Jede Zone ist mit einem oder mehreren Nameservern verbunden. (Host mit der Datenbank für die Zone)

Üblicherweise gibt es einen primären Nameserver und einen oder mehrere sekundäre Namesever, die ihre Informationen vom primären Nameserver erhalten.

Die Nameserver der Top Level Domänen (Root Server) sind redundant vernetzt.

Namensauflösung

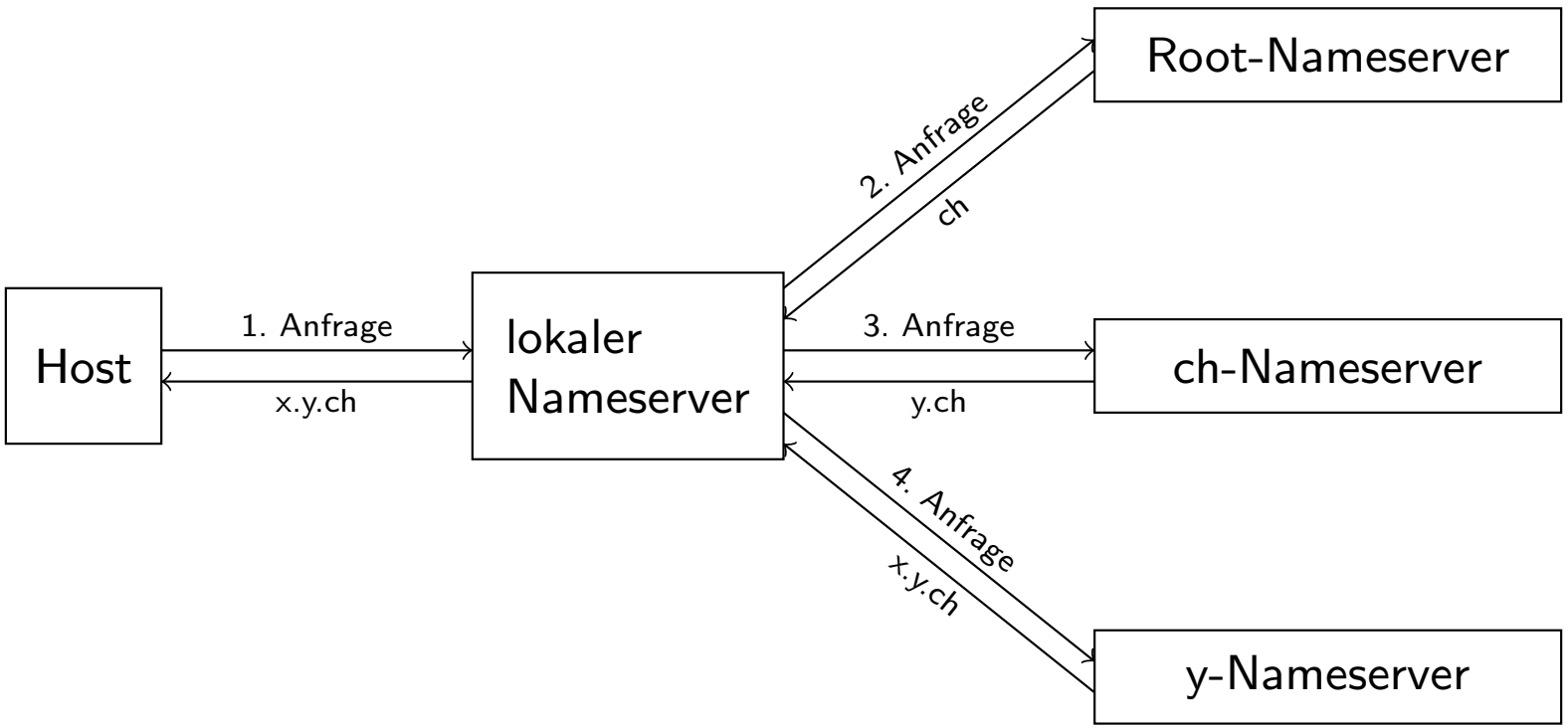


Abbildung 51: Namensauflösung

Als Übertragungsprotokoll wird UDP verwendet.

Abfragetyp

- Iterative Abfrage: Der lokale Nameserver fragt sich einzeln durch die Hierarchie der Nameserver, bis ihm die gesuchte Adresse zurückgeliefert wird.

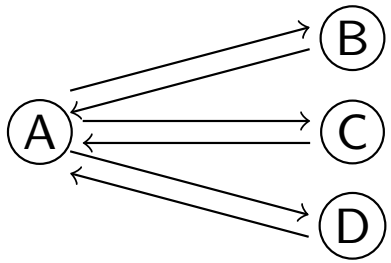


Abbildung 52: Iterative Abfrage

- Rekursive Abfrage: Ein Host fragt einen Server, der wiederum einen Server fragt usw. Sobald die Adresse gefunden wurde, wird sie auf dem umgekehrten Weg zurückgereicht.

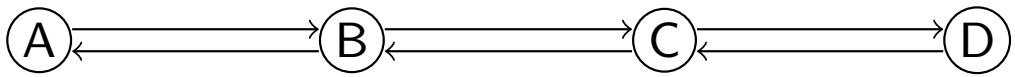


Abbildung 53: Rekursive Abfrage

Beispiel

```
user@pc:\~$ dig +trace +additional -t kollegistans.ch
```

Architektur

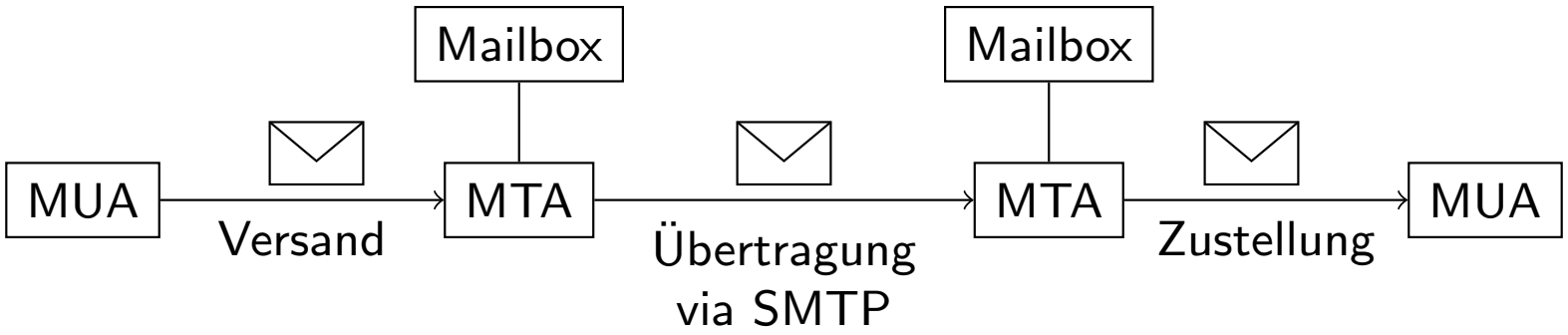


Abbildung 54: Architektur des E-Mail-Systems

- MUA (Mail User Agent): Ein Programm (meist mit graphischer Schnittstelle), um Nachrichten zu erstellen, anzusehen oder zu verwalten.
- MTA (Message Transfer Agent): Üblicherweise ein Prozess, der auf einem Mailserver läuft.
- SMTP (Simple Mail Transfer Protocol): Regelt den Austausch elektronischer Nachrichten im Internet. (RFC 5321)

Das Nachrichtenformat (IMF, RFC 5322)

Ein Mail-Objekt besteht aus einem Umschlag (*envelope*), der die für den Versand nötigen Informationen enthält (Empfänger- und Absenderadresse) sowie dem Inhalt (*content*).

Der Inhalt besteht aus dem Kopf (*header*) und dem Rumpf (*body*). Der Rumpf stellt die eigentlichen Nachricht dar. Der Header enthält unter anderem folgende Angaben:

`From`	Absender
`To`	Empfänger
`Cc`	weitere Empfänger
`Bcc`	weitere Empfänger (ohne Anzeige)
`Subject`	Betreff
`Date`	Zeitpunkt der Erstellung

MIME-Erweiterungen

SMTP war ursprünglich nur für ASCII-Textnachrichten ausgelegt.

Durch die weltweite Nutzung des Internets entstand das Bedürfnis, auch Binärdaten (Bilder, Videos, Programme) zu versenden.

MIME (Multipurpose Internet Mail Extensions) definiert fünf zusätzliche Nachrichten-Header:

MIME-Version:	1.0
Content-Description:	als menschenlesbare Zeichenkette
Content-Id:	eindeutige Identifikationsnummer
Content-Transfer-Encoding:	Codierung des Inhalts
Content-Type:	Nachrichtentyp

MIME-Typen

Typ	**Untertyp**	**Beschreibung**
text	plain, html, xml, ...	Textformate
image	jpeg, png, tiff, ...	Bilder
audio	mpeg, mp4, ...	Audiodateien
audio	mpeg, quicktime, ...	Videodateien
application	pdf, zip, ...	Daten von Anwendungen
message	http, rfc822	eingeschlossene Nachricht
multipart	mixed, alternative, ...	Kombination

Tabelle 3: MIME-Typen (Auswahl)

Base 64-Codierung

Nicht immer sind Mailserver in der Lage, binäre Nachrichten (Programme, Bilder, . . .) zu versenden.

Die Base 64-Codierung zerlegt drei aufeinanderfolgende Bytes in vier 6-Bit-Zahlen, die jeweils durch folgende ASCII-Zeichen codiert werden:

Nummer	0	. . .	25	26	. . .	51	52	. . .	61	62	63
Zeichen	A	. . .	Z	a	. . .	z	0	. . .	9	+	/

Tabelle 4: Base64-Codierung

Ist die Bytezahl kein Vielfaches von Drei, muss sie entweder mit einem oder zwei Nullbytes aufgefüllt werden. Dies wird durch die ASCII-Zeichen = bzw. == signalisiert.

Der Nachrichtentransport via SMTP

1. Der Sender-MTA ermittelt die Adresse Empfänger-MTA über das DNS ermittelt.
2. Der Sender-MTA baut eine TCP-Verbindung zum Empfänger-MTA auf (oft Port 25).
3. Der Empfänger nimmt die Nachricht nach entsprechender Prüfung entgegen.
4. Ist eine Nachricht unzustellbar, wird sie zusammen mit einem Fehlerbericht an den Absender zurückgeschickt.

SMTP ist ein ASCII-Protokoll; es kann z. B. manuell über eine Telnet-Verbindung mit einem geeigneten Mailserver ausgeführt werden.

Schwachstellen von SMTP

- keine Zugangskontrolle
- keine Prüfung des Absenders (Spam!)
- keine Integritätskontrolle
- keine Verschlüsselung

Durch das Protokoll ESMTP (Extended SMTP) lässt sich SMTP modular um entsprechende Funktionalitäten erweitern.

Mail Submission

In den Frühzeiten des Internets liefen die MUAs und MTAs auf den gleichen Rechnern, so dass eine Mail direkt vom Sender-MUA an den Sender MTA zum Versand weitergereicht werden konnte.

Heute laufen die MTAs auf Computern, die oft von jedem Ort des Internets aus erreichbar sein sollen, damit entfernte Benutzer Mails versenden und empfangen können.

Damit nicht jeder Netzteilnehmer Mails von einem beliebigen Mailserver (mit einer beliebigen Identität) versenden kann, wird heute der Zugang zu Mailservern mit der AUTH-Erweiterung eingeschränkt (Port 587).

Die Endzustellung

Mit SMTP können Mails aufgegeben und transportiert werden.

Für das Abrufen von Mails oder das Verwalten der Mailbox gibt es separate Protokolle.

- IMAP (Internet Message Access Protokoll)
  Auf dem Mailserver läuft zusätzlich ein IMAP-Server, der auf dem Port 143 lauscht. Auf der Gegenseite führt der MUA einen IMAP-Client aus, der IMAP-Befehle an den Server sendet (LOGIN, SELECT, DELETE, ...).
- POP3 (Post Office Protocol, Version 3)
  Das Vorgängerprotokoll von IMAP. POP3 lädt z. B. alle Nachrichten vom Mailserver auf den Client herunter und besitzt einen kleinerem Befehlsumfang als IMAP.

WWW

Das **W**orld **W**ide **W**eb wurde 1989 am Europäischen Kernforschungszentrum (CERN) in Genf entwickelt und ist ein Konzept, bei dem *Webserver* Informationen bereitstellen, die von *Clients* abgerufen werden können.

WWW

Das **W**orld **W**ide **W**eb wurde 1989 am Europäischen Kernforschungszentrum (CERN) in Genf entwickelt und ist ein Konzept, bei dem *Webserver* Informationen bereitstellen, die von *Clients* abgerufen werden können.

Das WWW verwendet das *Internet* als Kommunikationsstruktur.

WWW

Das **W**orld **W**ide **W**eb wurde 1989 am Europäischen Kernforschungszentrum (CERN) in Genf entwickelt und ist ein Konzept, bei dem *Webserver* Informationen bereitstellen, die von *Clients* abgerufen werden können.

Das WWW verwendet das *Internet* als Kommunikationsstruktur.

Das WWW basiert auf drei Grundkomponenten:

- HTTP (ein Kommunikationsprotokoll)
- URI (ein Adressierungsschema)
- HTML (eine Beschreibungssprache)

HTTP

HTTP = **H**yper**T**ext **T**ransfer **P**rotocol

HTTP

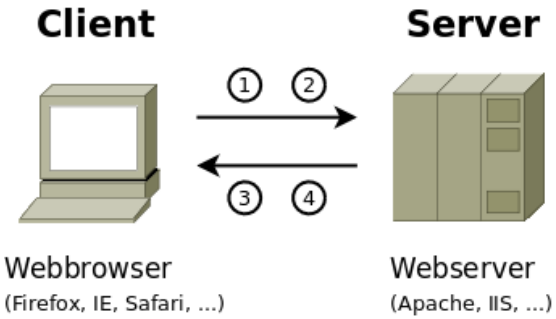
HTTP = **H**yper**T**ext **T**ransfer **P**rotocol

Grobe Beschreibung des HTTP:

HTTP

HTTP = **H**yper**T**ext **T**ransfer **P**rotocol

Grobe Beschreibung des HTTP:

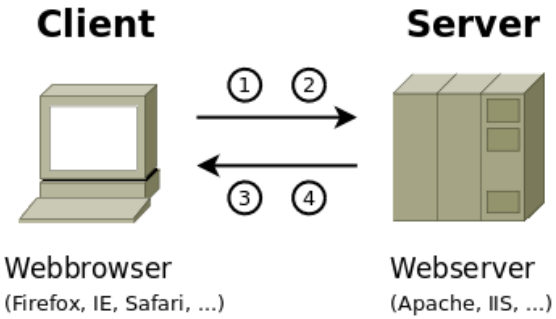


1. Verbindungsaufbau

HTTP

HTTP = **H**yper**T**ext **T**ransfer **P**rotocol

Grobe Beschreibung des HTTP:

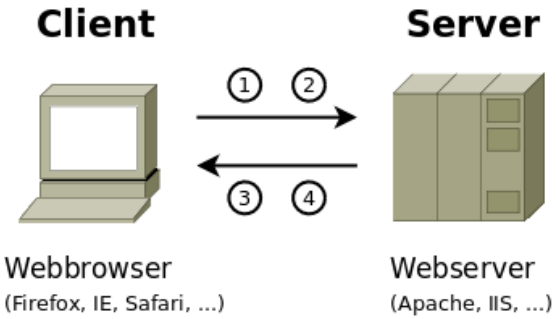


1. Verbindungsaufbau
2. Anforderung der Information

HTTP

HTTP = **H**yper**T**ext **T**ransfer **P**rotocol

Grobe Beschreibung des HTTP:

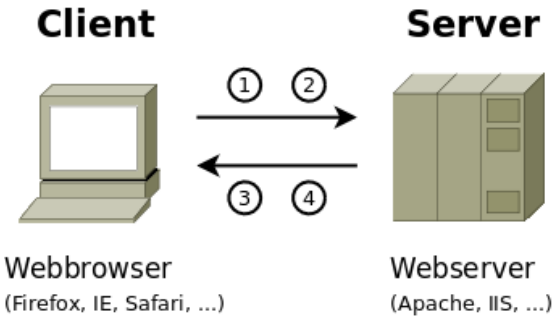


1. Verbindungsaufbau
2. Anforderung der Information
3. Übermitteln der Information

HTTP

HTTP = **H**yper**T**ext **T**ransfer **P**rotocol

Grobe Beschreibung des HTTP:

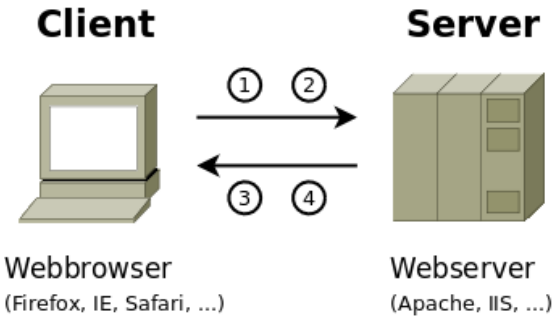


1. Verbindungsaufbau
2. Anforderung der Information
3. Übermitteln der Information
4. Verbindungsabbruch

HTTP

HTTP = **H**yper**T**ext **T**ransfer **P**rotocol

Grobe Beschreibung des HTTP:



1. Verbindungsaufbau
2. Anforderung der Information
3. Übermitteln der Information
4. Verbindungsabbruch

HTTP ist ein *zustandsloses* Protokoll.

Der Server „vergisst“, welchem Client er welche Informationen gesendet hat.

URI

URI = **U**niform **R**esource **I**dentifier

Eine URI ist eine Zeichenfolge, die eine physikalische oder abstrakte Datenquelle im WWW kennzeichnet.

URI

URI = **U**niform **R**esource **I**dentifier

Eine URI ist eine Zeichenfolge, die eine physikalische oder abstrakte Datenquelle im WWW kennzeichnet.

Für uns von Bedeutung ist eine spezielle Form einer URI:

URI

URI = **U**niform **R**esource **I**dentifier

Eine URI ist eine Zeichenfolge, die eine physikalische oder abstrakte Datenquelle im WWW kennzeichnet.

Für uns von Bedeutung ist eine spezielle Form einer URI:

URL =

URI

URI = **U**niform **R**esource **I**dentifier

Eine URI ist eine Zeichenfolge, die eine physikalische oder abstrakte Datenquelle im WWW kennzeichnet.

Für uns von Bedeutung ist eine spezielle Form einer URI:

URL = **U**niform

URI

URI = **U**niform **R**esource **I**dentifier

Eine URI ist eine Zeichenfolge, die eine physikalische oder abstrakte Datenquelle im WWW kennzeichnet.

Für uns von Bedeutung ist eine spezielle Form einer URI:

URL = **U**niform **R**esource

URI

URI = **U**niform **R**esource **I**dentifier

Eine URI ist eine Zeichenfolge, die eine physikalische oder abstrakte Datenquelle im WWW kennzeichnet.

Für uns von Bedeutung ist eine spezielle Form einer URI:

URL = **U**niform **R**esource **L**ocator

URI

URI = **U**niform **R**esource **I**dentifier

Eine URI ist eine Zeichenfolge, die eine physikalische oder abstrakte Datenquelle im WWW kennzeichnet.

Für uns von Bedeutung ist eine spezielle Form einer URI:

URL = **U**niform **R**esource **L**ocator

Diese Zeichenfolge gibt die Zugriffsmethode (*Schema*) und den Ort der Ressource im WWW an.

URI

URI = **U**niform **R**esource **I**dentifier

Eine URI ist eine Zeichenfolge, die eine physikalische oder abstrakte Datenquelle im WWW kennzeichnet.

Für uns von Bedeutung ist eine spezielle Form einer URI:

URL = **U**niform **R**esource **L**ocator

Diese Zeichenfolge gibt die Zugriffsmethode (*Schema*) und den Ort der Ressource im WWW an.

- `http://www.kollegistans.ch` (Webseite ohne Pfad)
- `http://de.wikipedia.org/wiki/Uniform_Resource_Locator` (mit Pfad)
- `file:///C:/meindatei.txt` (Pfad zu einer lokalen Datei)

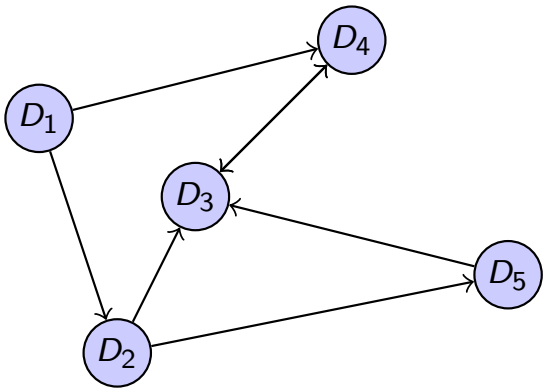
Worum geht es?

HTML: **H**yper**t**ext **M**arkup **L**anguage

Worum geht es?

HTML: **H**yper**t**ext **M**arkup **L**anguage

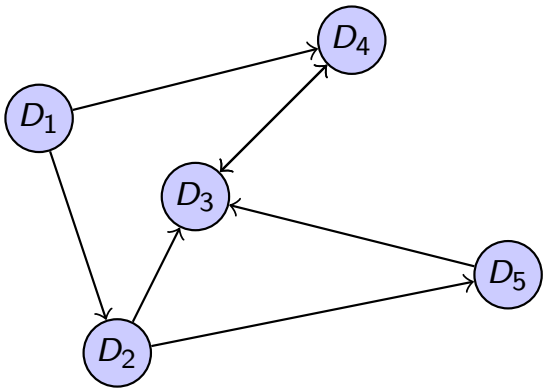
Hypertext: Dokumente sind durch Links netzwerkartig verknüpft.



Worum geht es?

HTML: **H**yper**t**ext **M**arkup **L**anguage

Hypertext: Dokumente sind durch Links netzwerkartig verknüpft.



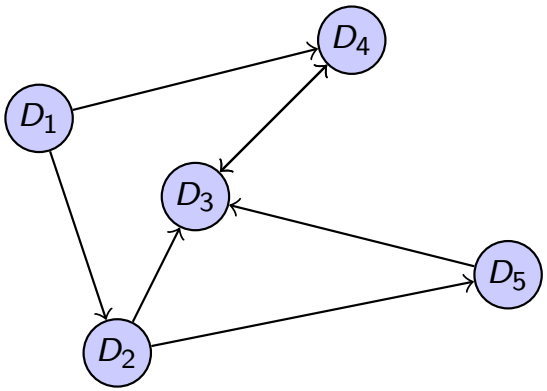
Markup: Auszeichnung, Beschreibung

```
<h1>Das Liebesleben der Waldameise</h1>
```

Worum geht es?

HTML: **H**yper**t**ext **M**arkup **L**anguage

Hypertext: Dokumente sind durch Links netzwerkartig verknüpft.



Markup: Auszeichnung, Beschreibung

```
<h1>Das Liebesleben der Waldameise</h1>
```

Language: „Sprache“ aber keine Programmiersprache, denn HTML kennt weder Variablen noch Kontrollstrukturen.

HTML – Grundstruktur

Grundstruktur eines HTML-Dokuments:

```
<!DOCTYPE html>

<html>

  <head>
    <!-- Dokumentkopf -->
  </head>

  <body>
    <!-- Dokumentrumpf ("Webseite") -->
  </body>

</html>
```

HTML – die DTD

Die HTML-*Document Type Definition* (DTD) legt fest, nach welchen Regeln das vorliegende HTML-Dokument aufgebaut ist. Die DTD für die aktuelle HTML-Version 5 lautet:

`<!DOCTYPE html>`.

Eine Dokumenttyp-Deklaration wird zwischen zwei spitze Klammern eingeschlossen und beginnt mit einem Ausrufezeichen.

HTML-Elemente (Tags)

Die Auszeichnung (das Markup) des Textes erfolgt meistens durch ein Paar von *Tags*, die das auszuzeichnende Objekt einschliessen. Beim Start- und Endtag wird der Tagname von einem Paar spitzer Klammern eingeschlossen wobei der Endtag zusätzlich einen Schrägstrich nach der öffnenden spitzen Klammer hat. Beispiel:

```
<p>Das ist <i>wichtiger</i> Text.</p>
```

- Der `p`-Tag zeigt an, dass der umschlossene Text ein eigenständiger Absatz ist und so dargestellt werden soll.
- Das `i`-Tag verlangt vom Browser, dass er das Wort `wichtiger` kursiv darstellt.

HTML-Tags, die paarweise auftreten, werden *Container-Tags* (*Container-Elemente*) genannt.

Daneben gibt es auch sogenannte *Standalone-Tags*, die leer sein

HTML – Attribute

Das erste Wort in einem Start-Tag ist der Tag-Name. Alle weiteren Wörter in einem Starttag werden *Attribute* genannt und haben den Zweck, die Funktion eines Tags zu verändern oder sie genauer zu definieren. Ein Attribut besteht meistens aus einem Namen und einem Wert (in Anführungs und Schlusszeichen), die durch ein Gleichheitszeichen verbunden sind. Beispiele:

- `<input type="text", size="15", disabled>`
- `<input type="button" value="Start">`

Das Standalone-Tag `<input>` erzeugt im ersten Beispiel ein Eingabefeld für maximal 15 Zeichen, das jedoch deaktiviert ist.

Im zweiten Beispiel wird eine Schaltfläche mit der Aufschrift *Start* erzeugt.

Attribute stehen immer im Start- und nie im Endtag. Wie die Beispielen zeigen, können auch Standalone-Tags Attribute haben.

HTML – Schriftgestaltung

Tag	Beschreibung
`<b>...</b>`	Fettdruck (bold)
`<i>...</i>`	Kursivschrift (italic)
`<tt>...</tt>`	Nichtproportionalschrift (teletype)
`<em>...</em>`	Hervorheben (emphasize)
`<sub>...</sub>`	Tieferstellen (subscript)
`<sup>...</sup>`	Höherstellen (superscript)
`<pre>...</pre>`	Formatierung übernehmen (preformatted)

Es gibt noch weitere Tags zur Schriftgestaltung, deren Verwendung aber vom W3-Konsortium „missbilligt“ werden, da HTML hauptsächlich die logische Struktur und nicht die Darstellung eines Dokuments auszeichnen soll. Für das Aussehen des Textes sollte in Zukunft CSS verwendet werden.

HTML – Dokumentgliederung

Tag	Beschreibung
`<p>...</p>`	Absatz (paragraph)
`<h1>...</h1>`	Überschrift 1. Ordnung (heading)
...	...
`<h6>...</h6>`	Überschrift 6. Ordnung (heading)
` `	Zeilenumbruch (break)
`<hr>`	horizontale Trennlinie (horizontal rule)

HTML — Verweise (Links)

Hyperlinks werden mit `<a>...</a>` definiert. (anchor = „Anker“)

```
<a href="http://www.kollegistans.ch">zum Kollegi</a>
```

- Das Attribut `href` (hypertext-reference) muss als Wert eine gültige URL haben. Bei einer absoluten URL das Schema nicht vergessen!
- Anstelle einer *absoluten* URL kann auch eine zum aktuellen Dokument *relative* Adresse angegeben werden. Dabei steht ein Punkt für das aktuelle Verzeichnis und zwei Punkte für das darüber liegende Verzeichnis.

  ```
  <a
  href="./../form/order.html">Bestellformular</a>
  ```

  (Gehe eine Verzeichnisebene hinauf, wechsle von da aus in den Unterordner `form` und öffne dort das Dokument `order.html`.)

HTML – Bilder

Bilder werden mit dem Standalone-Tag `<img>` gekennzeichnet. Das Attribut `src` (source) mit dem Pfad zur Bilddatei sollte natürlich vorhanden sein. Beispiele:

- `<img src="./tulpe.jpg" width="100px">`
  Lade das Bild `tulpe.jpg` aus dem aktuellen Verzeichnis und skaliere es auf eine Breite von 100 Pixeln.
- `<img src="./cars/srt8.png" height="50%">`
  Lade das Bild `srt8.png` aus dem Unterverzeichnis `cars` und verkleinere es Bild auf 50% der ursprünglichen Höhe.

HTML – Listen

nummerierte Aufzählungen ... (ordered list)	unummerierte Aufzählungen ... (unordered list)
  - erster Eintrag - zweiter Eintrag - dritter Eintrag 	 - erster Eintrag - zweiter Eintrag - dritter Eintrag
1. erster Eintrag 2. zweiter Eintrag 3. dritter Eintrag	▶ erster Eintrag ▶ zweiter Eintrag ▶ dritter Eintrag

Listen können auch verschachtelt werden

HTML – Tabellen

- `<table>...</table>` Tabellenstruktur
- `<tr>...</tr>` Tabellenzeile (table row)
- `<td>...</td>` Tabellenzeile (table data)
- CSS für Formatierungen (Rahmen, Zellenausrichtung) verwenden.

```
<table border="1">
  <tr> <td>abc</td><td>xyz</td> </tr>
  <tr> <td>123</td><td>456</td> </tr>
</table>
```

ergibt etwa:

abc	xyz
123	456