

Medien

Ausbreitungsgeschwindigkeit

Lichtgeschwindigkeit im Vakuum: $c_0 = 299'792'458 \frac{m}{s}$
Faustregel in Medien: $200'000 \frac{km}{s} = 20 \frac{cm}{ns}$

Signaldämpfung

Angegeben in Dezibel; auch: Insertion Loss, Attenuation

Dämpfung $A = 10 * \log(P_1/P_2) = 20 * \log(U_1/U_2)$

Höhere Frequenz → mehr Dämpfung

Halbierung der Leistung entspricht ca. 3dB

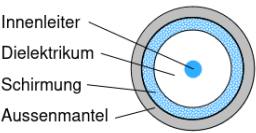
Signal-to-Noise-Ratio SNR

$SNR = 10 * \log(P_{Signal}/P_{Noise}) dB$

Kabel

Koaxial

- + besser als twisted pair für hohe Frequenzen
- + relativ unempfindlich gegen elektromagn. Störungen
- mechanisch heikel (knicken/quetschen)



Parasymmetrisch (Twisted Pair)

- + bereits lange im Einsatz
- + bei guter Qualität auch für Breitband geeignet
- mit oder ohne Schild

Shielded Twisted Pair (STP)

Bezeichnet nach ISO 11801: xx/yTP

xx steht für die Gesamtschirmung

- U ungeschirmt
- F Folienschirm
- S Geflechschirm
- SF Folien- & Geflechschirm

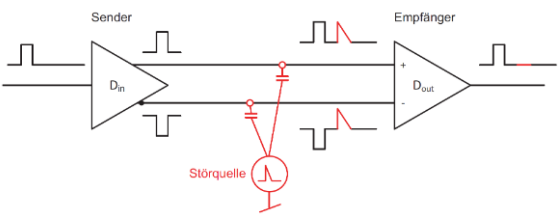
y steht für die Aderpaarschirmung

- U ungeschirmt
- F Folienschirm
- S Geflechschirm

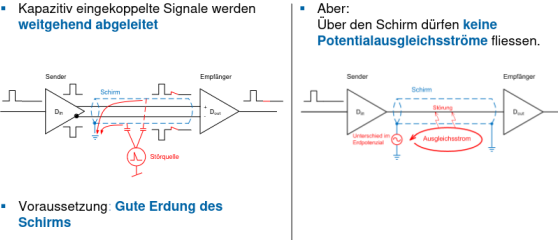
Störungen bei TP

Kapazitive/Induktive Störungen treten bei TP öfter als bei Koax oder Glasfaser auf.

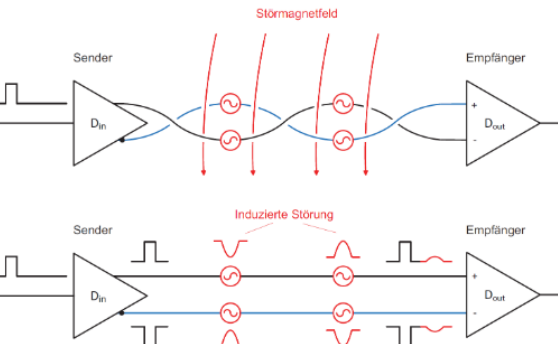
Kapazitive Störungen von benachbarten Leitungen heissen Crosstalk(über-/nebensprechen). Diese Störungen können durch ein invertiertes komplementäres Signal weitgehen aufgehoben werden. Der Empfänger subtrahiert die beiden Signale und eliminiert dadurch Störungen.



Alternativ können kapazitive Störungen mit einem leitenden Schirm abgefangen werden.



Induktive Störungen (durch ein Magnetfeld) können nicht durch ein komplementäres Signal alleine gelöst werden, da die die Störung auf beiden Signalen entgegengesetzt ist. Dies kann über verdrehen der Aderpaare gelöst werden, benachbarte Schleifen heben sich so immer auf.



Kategorien

- Cat 1..4 Billigkabel für analoge Sprachübertragung (< 1Mb/s)
- Cat 5 bis 100 MHz, z.B. 100Mb/s oder 1 Gb/s Ethernet bis 100m
- Cat 6 250 MHz, 1 Gb/s Ethernet und 10 Gb/s Ethernet bis 55m
- Cat 7 600 MHz, z.B. für 10 Gb/s Ethernet bis 100m

Lichtwellenleiter

- + hohe Bandbreite → hohe Datenrate
- + geringe Dämpfung → lange Übertragungsstrecken
- + resistent gegen elektromagnetische Störungen

Zentrum aus Kernglas mit hoher optischer Dichte (Brechungsindex $n_{kern} = 1.5$) umschlossen von Mantelglas mit geringer optischer Dichte (Brechungsindex $n_{mantel} = 1.48$). Dadurch werden Lichtstrahlen im Kern totalreflektiert($\beta > 90^\circ$) und keine Energie durch Absorption verloren

Multimode

- + dicker Kern (mehrere Wege/Modes für das Licht)
- hohe Dispersion(Signalverschmierung auf langen Wegen)
 - Kann reduziert werden durch Einsatz von Gradientenfaser (übergang zwischen Kern und Mantel)

Singlemode

- + keine Dispersion
- + hohe Datenraten auf hohe Distanzen
- dünner Kern, nur eine Grundmode

Physical Layer

Begriffe

(Leitungs-)Symbol

Zu einem gewissen Zeitpunkt übertragenes physikalisches Signal das mit einer bestimmten Symbolrate seinen Wert verändert.

nicht wie in INCO "eine von N möglichen Nachrichten"

Informationsgehalt/Bit

Informationsgehalt (von Symbol/Nachricht) $N_{Bit} = ld(\text{Anzahl Möglichkeiten})$

Zeichen

Einheit der übertragenen Daten, z.B. ein ASCII Zeichen

Baudrate

Schrittgeschwindigkeit = Leitungs-Symbole pro Sekunde

Maximale Baudrate f_s ist doppelte Bandbreite B (Hz) gemäss Nyquist:

$f_s = 2B$

Durchsatz/Bit-/Datenübertragungsrate

übertragung von Information pro Zeit

Maximale Bitrate (Hartley's Gesetz)

$R \leq 2B * ld(\# \text{unterscheidbare Signalzustände})$

Potenzen (in der Kommunikation werden NICHT zweierpotenzen verwendet):

- kBit = 10^3 Bit → kbps = 10^3 bps
- MBit = 10^6 Bit → Mbps = 10^6 bps
- GBit = 10^9 Bit → Gbps = 10^9 bps

Kanalkapazität

Berücksichtigt neben der Bitrate auch Störungen

$C_s[Bit/s] = B * ld(1 + \frac{S}{N})$

Flowcontrol

Erlaubt Empfänger den Sender temporär zu stoppen Verwendung bei Speichermangel/langsamer Verarbeitung oder Überlast im Netzwerk

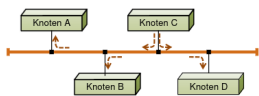
Kann auch implizit vorkommen, in dem der Sender nach jeder Nachricht auf eine Quittung wartet bevor er die nächste schickt

Ethernet

Begriffe

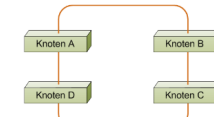
Topologien

Bus



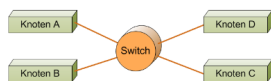
- Alle Stationen
 - sind passiv angeschlossen
 - horchen Leitung permanent ab
 - werden aktiv, wenn sie etwas senden wollen
- Keine festgelegte Ausbreitungsrichtung
- Empfänger erkennt anhand einer Adresse, ob die Daten für ihn relevant sind

Ring



- Benötigt Verfahren zur Verhinderung von "endlosem Kreisverkehr"
- Gewisse Redundanz: beim Ausfall einer Station kann immer noch jede Station erreicht werden

Stern



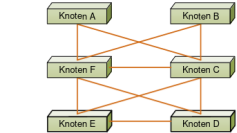
- Jede Station an zentralen Verteiler (Switch/Bridge) angeschlossen
- Verteiler entkoppelt Knoten elektrisch und macht LAN weniger störungsanfällig
- Verteiler sendet Daten, die er von einer Station erhält, an die anderen Knoten weiter

Linie



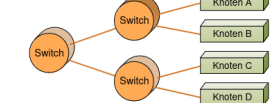
- Punkt-zu-Punkt Verbindungen zwischen benachbarten Knoten
- Alle Stationen müssen
 - Daten empfangen
 - Daten regenerieren
 - falls nötig weiterleiten
- Der Ausfall einer Station führt zur Segmentierung des LAN in zwei Teile

Vermascht (teilweise oder komplett)



- Weitere Erhöhung der Redundanz:
 - Ausfall einer oder eventuell auch mehrerer Stationen oder Verbindungen kann toleriert werden
 - Zusätzliche Kosten und Aufwand, um mehrfache Lieferung von Daten zu verhindern

Baum



- Hierarchische Erweiterung der Sterntopologie
- Intelligenten Switches ermöglichen einen Grossteil der Kommunikation „lokal“
 - zwischen A und B bzw. C und D
- Dadurch Verringerung der Last für die einzelnen Switches

Übertragungsarten

Immer nur ein Sender! Anzahl empfänger unterscheidet sich

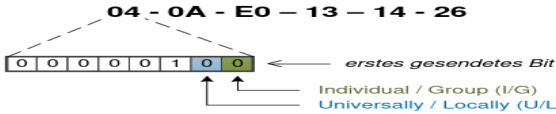
Unicast Genau 1 klar definierter Empfänger, Frame trägt dessen Adresse

Multicast Gruppe von Empfängern, Frame trägt Adresse dieser Gruppe

Broadcast An alle Knoten im LAN, Frame trägt Broadcast-Adresse

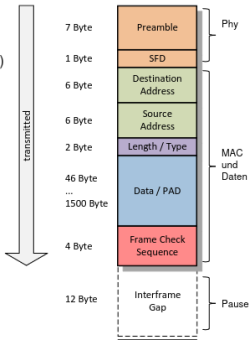
MAC-Adresse

Erste drei Byte Hersteller + Individual/Group & Universally/Locally bits, die letzten drei Bytes Laufnummer durch Hersteller



Ethernet Frame

- Bit Synchronisation durch Präamble, Bytes & Frames durch Start Frame Delimiter (SFD = 1010'1011)
- Frame-Länge 64 ... 1518 Byte (ohne Präable und SFD)
- Bytes nacheinander übertragen, pro Byte **LSB** zuerst
- Zahlenwerte (Length/Type/...) höchstwertiges Byte zuerst gesendet (Network Byte Order)
- Length/Type (2 Bytes):
 - Fall 1: Länge von DATA ohne PAD (≤ 1500)
 - Fall 2: Typ von DATA = Protokoll der nächsten Schicht (≥ 1536))
 - Beispiel: 0x0800 für IP
- Data / Padding (46 – 1500 Bytes):
 - Enthält die eigentlichen Datenbytes (Nutzinformation)
 - Bei weniger als 46 Bytes Nutzdaten wird mit Padding (PAD) Bytes aufgefüllt
- Frame Check Sequence, FCS (4 Bytes):
 - IEEE CRC-32 Algorithmus
- Interframe Gap, IFG (12 Bytes):
 - "Zwangspause" zwischen aufeinanderfolgenden Frames
 - Ist **NICHT** Teil des Ethernet Frames



Die FCS wird mit IEEE CRC-32 berechnet. Die Hamming-Distanz ist Abhängig von der Frame-Länge: 6 bis 226, danach 5 bis 2974 (Ethernet Limit = 1518)

Kennzahlen zu Ethernet

Overhead 18 Bytes aus DA,SA,L/T,FCS

Frame-Size 64 bis 1518 Bytes

Sendedauer $T_{frame} = \frac{N_{bit} = (FrameSize + 8) * 8}{Bitrate}$

Dauer der Leitungsbelegung $T_{leitung} = \frac{N_{bit} + 96}{Bitrate}$

Network Gear

Kreieren eine Broadcast Domain / LAN, Endgeräte merken davon nichts (transparent) sondern sprechen aus ihrer sicht direkt den Empfänger an.

Repeater/Hubs verstärkt Signal von einem Port und leitet sie weiter auf allen anderen
Bridge/Switch lernt Adressen, leitet Daten weiter an die richtigen ports (Filtering Database)

VLAN (802.1Q)

Repräsentiert eigene, virtuelle Broadcastdomain.

Auf einem Switch kann jedem Port nur eine VLAN-ID gegeben werden, da die Zuordnung erst am Switch selbst passiert (muss konfiguriert werden, "managed switch").

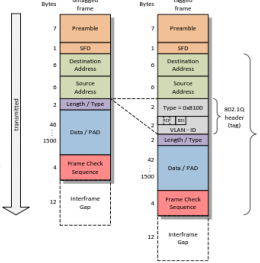
Bei VLAN gibt es 2 verschiedene Arten von Links (Verbindung zwischen Geräten).

Access Links nur einem VLAN zugehörig

Trunk Links oft zwischen Switches, gesendete Frames können mehreren VLANs angehören → Frames müssen vom Switch getagged werden

Erweiterung des Ethernet Headers durch einen VLAN-Tag

- Der Type 0x8100 bedeutet, dass das Frame «getagged» ist
 - Ein 12 Bit Identifier (VLAN-ID, VID) besagt, welchem VLAN das Frame angehört
 - Die 3 Bit des Priority Code Point (PCP) erlauben, das Frame mit einer Priorität zu versenden
 - Mit dem Drop Eligibility Indicator (DEI) werden Frames markiert, die bei Überlastsituationen zuerst verworfen werden sollen
- Die maximalen Nutzdatenlänge bleibt erhalten, der Ethernet Frame wird 4 Bytes länger
- VLANs können transparent eingesetzt werden



PCP: höhere Prio kann tiefere überholen in der Bridge

(Rapid) Spanning Tree

Ziel: Alle Segmente in einer loop-freien Topologie verbinden
Idee:

1. willkürliche aber eindeutige Root-Bridge wählen
2. von Root aus Baum aufbauen
3. redundante Pfade sperren

Algorithmus:

1. Initialisierung

- Alle Ports für Nutzdaten blockiert
- Annahme: "Ich bin Root"
- Austausch BPDUs mit Nachbarn

2. Aufbau des Spanning Tree

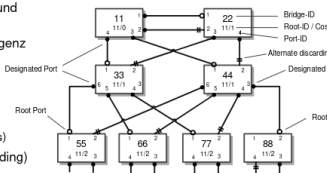
- Aufdatieren der Info zu Root (kleinste ID) und Pfadkosten zu dieser
- Austausch aufdatierter BPDUs bis Konvergenz

3. Setzen der Port Roles

- Freigeben für Nutzdaten von
 - Root-Ports (Empfang der «besten» BPDU)
 - Designated-Ports (Gegensstück zu Root-Ports)
- Alle anderen Ports bleiben blockiert (Discarding)

BPDUs (Bridge Protocol Data Units) beinhalten:

Root-ID (aus lokaler Sicht):	8 Byte
Root-Cost (aus lokaler Sicht):	2 Byte
Bridge-ID ("Ich"):	8 Byte
Port-ID (Sende-Port):	2 Byte



Das normale STP konvergiert & reagiert sehr langsam, die Rapid Variante RSTP ist vom Grundkonzept identisch aber reagiert schneller auf Topologieänderungen (~500ms)

Evolution von Ethernet

Bezeichnung(802.3): [Bitrate in Mbit/s] BASE/BROAD-[Art/Codierung] Bemerkung: früher statt Art/Codierung die max. Segmentlänge in 100m

Beispiele von relevanten in diesem Rahmen

- 10 Mb/s: 10Base-T
- 100 Mb/s: 100Base-TX
- 1000 Mb/s: 1000Base-T

Codierungsarten:

- T, TX, T1 Twisted Pair
- SR, DR, LR optisch
- C Twinax
- K Backplane

Autonegotiation

Für Rückwärtskompatibilität, damit Sender & Empfänger wissen was möglich ist. Umgesetzt mittels Fast Link Pulses FLP (seit 100BASE-TX) zwischen den regulären Normal Link Pulses NLP(10BASE-T).

100BASE-TX

Coderiung NRZI + umwandeln von 4 Bits in 5-Bit PCS (4B5B)
Start-of-Stream ähnlich zu 10BASE-T Preamble, bestimmte PCS Zeichen J/K
End-of-Stream markiert Ende des Frame, bestimmte PCS Zeichen T/R
IDLE NEU: die Leitung wird ununterbrochen mit IDLE gefüllt falls keine Nutzdaten, PCS Zeichen I

1000BASE-T

- 5-wertiger Leitungscode PAM-5
- Vollduplex: Alle 4 Aderpaare in beide Richtungen dank Gabelschaltung
- Next-Page bei FLP

10GBASE-T

- 16-wertiger Leitungscode PAM-16
- Effizientere Verteilung der Bits auf Aderpaare
- forward error correction
- neuer Stecker ab CAT7: GG45

IP (Network Layer)

Router/Gateway

Verbinden verschiedene Netze mit potentiell unterschiedlichen Technologien

Lösen 2 Aufgaben:

Routing Aufbau und Update von Routingtabellen
Forwarding Weiterleiten der Pakete anhand Routingtabellen

Routing-Tabelle

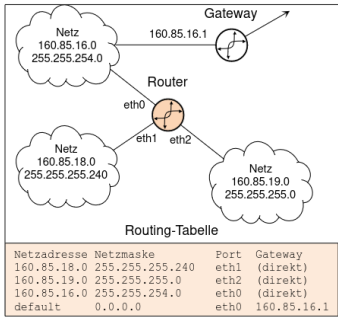
löst: wie Kann ich welches Netz erreichen?

Bei hierarchischem Routing (Router wissen welche Netze an anderen Router sind forwarden entsprechend) kommen oft auch aggregierte Routen vor.
Wenn so z.B. 130.0.0.0/25 und 130.0.128.0/25 beide via den selben Router erreichbar sind wird diese Route aggregiert und als 130.0.0.0/24 zusammengefasst.

Beispiel

- Die Routing-Tabelle sei nach der Länge der Netzmaske sortiert
- Sie wird von oben nach unten durchsucht
- Aus der Zieladresse und Netzmaske wird eine Zielnetzadresse bestimmt
- Verglichen werden **nur** die **Netzadressen**
- Der erste Eintrag, der „passt“, wird für die Weiterleitung verwendet⁽¹⁾
- Der default-Eintrag am Schluss (falls vorhanden) passt immer

⁽¹⁾ Existieren gleichwertige Alternativen, werden zusätzliche Kriterien (z.B. Pfadkosten) verwendet



Classful Routing

Ursprünglich wurden IP Adressen in 5 Routing Klassen eingeteilt (Classful Routing). Die ersten vier Adressbits erlauben eine Bestimmung der Klasse. Wird nicht mehr gemacht weil oftmals Adressraum verschwendet wird.

Klasse	Adressbereich	Anzahl Netze	Interfaces pro Netz
A	1.0.0.0 – 127.255.255.255	127	16'777'214
B	128.0.0.0 – 191.255.255.255	16'384	65'534
C	192.0.0.0 – 223.255.255.255	2'097'152	254
D	224.0.0.0 – 239.255.255.555	Multicast Adressen	
E	240.0.0.0 – 255.255.255.255	Reserviert für zukünftige Nutzung	

Subnetting

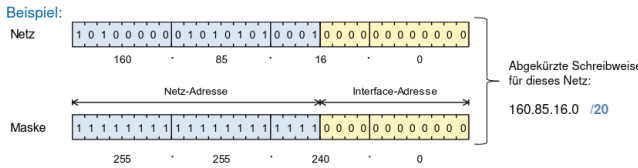
- das Netz in kleinere Subnetze teilen
- funktioniert simplifiziert durch beliebiges erweitern der Netzadresse
- hintereinanderliegende Netze können zusammengefügt werden

198.51.0110 0100	0000 0000	= C-Netz 198.51.100.0 /24
198.51.0110 0101	0000 0000	= C-Netz 198.51.101.0 /24
198.51.0110 0110	0000 0000	= C-Netz 198.51.102.0 /24
198.51.0110 0111	0000 0000	= C-Netz 198.51.103.0 /24
↓		
198.51.0110 01	00.0000 0000	= Subnetzmaske 255.255.252.0 oder /22

Adressierung / IPv4

Adresse eines Host = Netz-Adresse + Interface-Adresse

Subnetzmaske



Netzadresse

- reserviert, darf NICHT für interfaces verwendet werden
- tiefste Adresse im Subnet → alle interface bits 0
- berechnung durch $InterfaceAdresse \wedge Subnetzmaske$

Broadcast-Adresse

- reserviert, darf NICHT für interfaces verwendet werden
- höchste Adresse im Subnet → alle interface bits 1
- berechnung durch $InterfaceAdresse \vee invertierte Subnetzmaske$

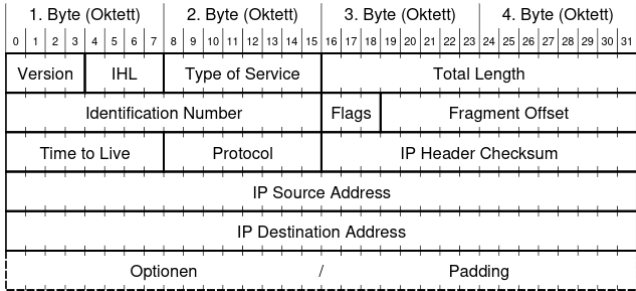
Private Adressen

Die 172.0.0.0/8 Adressen sind reserviert für Loopback und verlassen den Host nicht. Sie werden an ein emuliertes Loopback-Gerät geschickt dass direkt returned (kein Interface nötig).

Klasse	Netzadresse(n)	Anzahl Netze	Subnetzmaske
A	10.0.0.0	1	255.0.0.0
B	172.16.0.0 – 172.31.0.0	16	255.255.0.0
C	192.168.0.0 – 192.168.255.0	256	255.255.255.0

IPv4 Header

immer minimum 4Byte Blöcke



Version 4 oder 6
IHL Internet Header Length (/ 4 weil immer 4 Byte Blöcke)

- Header ohne optionen = 20 Bytes → IHL = 5
- Maximalwert 15 (4 Bits)

TOS Type of Service, erlaubt priorisierung

Feld	Position	Funktion	Definition
DSCP	0 – 5	Differentiated Services Codepoints	RFC 2474
ECN	6 – 7	Explicit Congestion Notification (IP Staukontrolle)	RFC 3168

Total Length inclusive Header & Nutzdaten
TTL Time To Live, In Anzahl Hops. Verhindern Loops, jeder Router dekrementiert, bei 0 → Paket verwerfen

Protocol Protokoll der Nutzdaten

- 1 ICMP Internet Control Message Protocol
- 6 TCP Transport Control Protocol
- 17 UDP User Datagramm Protocol

Header Checksum schützt NUR den Header, bei jedem Router neu berechnet(TTL)

Options/Padding variabel, heute selten, Padding für 32 bits

Identification Number identifikation, bleibt für fragmentierte Pakete gleich

Flags 3 Bits, steuert Fragmentierung über einzelne Bits (Aufzählung von links aus)

- 0. reserviert, immer 0
- 1. DF, 0/1 → May / Dont Fragment
- 2. MF, 0/1 → Last / More Fragments

Fragment Offset 13 Bits, steht für Anzahl 8 Byte Blöcke, bestimmt wo im gesamt paket die Daten hingehören

Fragmentierung

IP-Paket maximal 65535 Bytes, aber limitiert durch MTU des Netz.
Problem: spätere Netze haben eventuell tiefere MTU → Fragmentierung.

- jedes Fragment erhält seinen eigenen Header
- Identification Number bleibt gleich
- Total Length jeweils die Länge des Paket
- alle Pakete ausser letztes haben MF = 1 in den Flags
- alle Pakete haben die gleiche, vielfaches von 8, maximale Länge (ausser letztes)

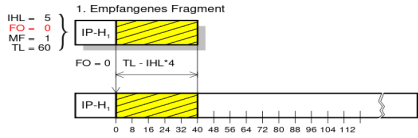
Paket mit 112 Bytes Nutzdaten, MTU = 60 Bytes → 3 Fragmente mit 40,40,32 Bytes

- 1. Fragment: TL = 60, Identification = 649, Fragment Offset = 0, MF = 1
- 2. Fragment: TL = 60, Identification = 649, Fragment Offset = 5, MF = 1
- 3. Fragment: TL = 52, Identification = 649, Fragment Offset = 10, MF = 0

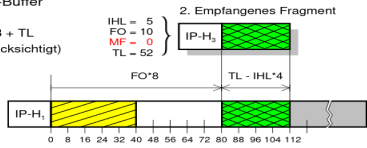
Reassembly

- Fragmente werden erst beim Zielhost zusammengesetzt
- Pakete mit FO = MF = 0 sind "komplett" → brauchen kein Reassembly
- für alle anderen (=Fragmente) wird folgende Datenstruktur alloziert
 - 1. Daten-Buffer für grösstmögliches Paket (64KB)
 - 2. Header-Buffer zur Rekonstruktion des original Headers
 - 3. Timer (ca. 15 Sekunden Timeout)

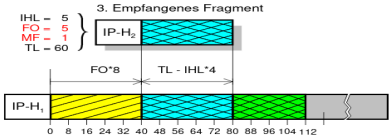
1. Fall: Das vorderste Fragment wird empfangen (FO = 0, MF = 1)
- Header wird in den Buffer kopiert
 - Nutzdaten werden am Anfang des Datenbuffers eingefügt
 - Länge der Daten in Bytes beträgt TL – IHL * 4
 - Entsprechenden Bits (1 Bit pro 8 Byte) in der Fragment-Block-Bit-Tabelle werden gesetzt



2. Fall: Das hinterste Fragment wird empfangen (MF = 0, FO > 0)
- Nutzdaten werden an Position FO * 8 in den Datenbuffer eingefügt
 - Länge der Daten in Bytes beträgt TL – IHL * 4
 - Entsprechenden Bits in der Fragment-Block-Bit-Tabelle werden gesetzt
 - Das Total-Length Feld im Header-Buffer wird gesetzt
 - Gesamtlänge in Bytes beträgt FO * 8 + TL
 - (Header Länge ist dann ein Mal berücksichtigt)
 - Nichtbenötigter Speicherplatz kann freigegeben werden, Gesamtlänge der Daten ist bekannt



3. Fall: Ein mittleres Fragment wird empfangen (MF = 1, FO > 0)
- Nutzdaten werden an Position FO * 8 in den Datenbuffer eingefügt
 - Länge der Daten in Bytes beträgt TL – IHL * 4

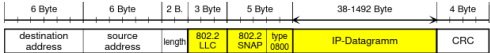


Kapselung

- Ethernet-Encapsulation: das IP-Paket wird als Data des Ethernet Frame übertragen → MTU = 1500
- Type im Ethernet-Frame = 0x0800

Selten wird IEEE 802.2/802.3 Encapsulation verwendet:

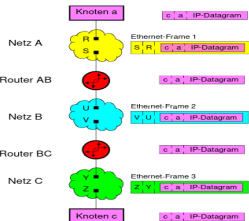
- Zusätzlicher Logical Link Control / Subnet Access Protocol (LLC/SNAP) Header vor dem IP Paket eingefügt, z.B. für Flow Control, Error Detection and Retransmission
- Die MTU ist damit 1500 – 8 = 1492 Bytes



Übertragung mit Encapsulation

Was geschieht bei der Übertragung genau?

- Knoten a sendet ein IP Paket an Knoten c → das Paket enthält die IP Adressen von a und c
- Knoten a konsultiert die Routing Tabelle und sieht:
 - dass c über den Router AB erreicht werden kann, und
 - Kennt nun die IP Adresse von Router AB
- Knoten a generiert ein Ethernet Frame, welches an die Hardware-adresse S von Router AB gesendet wird
- Router AB empfängt das Ethernet Frame, packt das IP Paket aus und modifiziert den Header (TTL)
- Router AB konsultiert die Routing Tabelle und sieht:
 - dass c über den Router BC erreicht werden kann, und
 - Kennt nun die IP Adresse von Router BC
- etc.

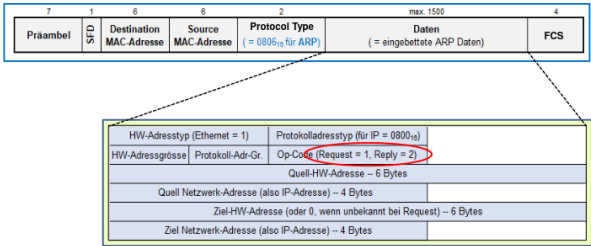


Adressauflösung mit Address Resolution Protocol (ARP)

- NICHT teil von IP selbst
- Löst: Host a möchte die Hardware Adresse für die IP Adresse 160.85.20.33
- jeder Host führt lokalen ARP-Cache für bekannte Adressen

- commands:
 - arp -a / ip neigh show
 - arp -d <ipaddr> / ip neigh del
 - arp -s <ipaddr> <hwaddr> / ip neigh add
- Gratuitous ARP (unnötig/unbegründet)
 - z.B. von Windows nach dem Starten
 - nach Adresszuweisung wird ARP request an eigene IP gesendet für Konflikterkennung
 - gratuitous reply für e.g. Cache-Refresh / Broadcast

- ARP-Request und ARP-Response sind je in genau einem Ethernet Frame enthalten mit Type 0806
- Beim Request ist die Destination Address FF-FF-FF-FF-FF-FF (Broadcast Frame) und die Hardware Address of Target ist 0

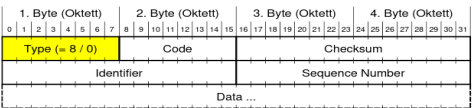


Internet Control Message Protocol ICMP

- Fehlermeldungen auf Internet Layer (TTL = 0)
 - Test ob anderer Host erreichbar ist (ping)
 - Gekapselt in IP Paketen aber wird zu Network Layer gezählt
 - gebräuchliche Typen (in ICMP header)
- Fehler:
- 3 Destination Unreachable - e.g. Dont Fragment gesetzt aber MTU zu klein
 - 5 Redirect - Router merkt dass Host direkteren Weg nehmen könnte → Host sendet vervollständigt Routingtabelle und nimmt direkten weg
 - 11 Time Exceeded - Router setzt TTL = 0 oder Timeout für Fragmente
 - 12 Parameter Problem: Bad IP Header - IP Header hat ungültigen Wert
- Information:
- 0 Echo Reply - Host erhält Echo Request → Echo Reply mit gleichen Daten
 - 8 Echo (Request) - Host sendet Echo Request (ping)
 - 7 Timestamp - Wie Echo aber mit Timestamp austausch
 - 13 Timestamp Reply

ICMP Echo / Reply

- Test, ob Host "up" ist
- Host antwortet auf Echo Request (Type 8) mit Echo Reply (Type 0), mit gleichem Inhalt wie der Echo Request
- Format
- Identifier: Erlaubt Zuordnung von Reply zu welchem Echo gehört
 - Sequence Number: Wird innerhalb eines Identifiers jeweils um 1 erhöht
 - Data: Beliebige Daten, werden vom Empfänger gespiegelt

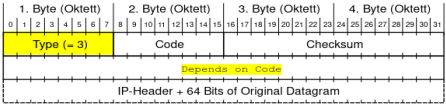


ICMP Destination Unreachable

Vom Router/Zielhost an Absender gesendet, wenn Paket nicht weitergeleitet werden kann

Feld	Wert/Semantik
Type	3
Code	0 = net unreachable, 1 = host unreachable, 2 = protocol unreachable, 3 = port unreachable, 4 = fragmentation needed and DF set, 13 = communication administratively prohibited
Checksum	Prüfsumme über die ICMP Meldung
IP Header + 64 Bits of Original Datagram	Information für den Empfänger zur Zuordnung der Meldung zu einem gesendeten IP Paket

Welche Codes werden von einem Router und welche vom Zielhost generiert? Welche vermutlich von einer Firewall?



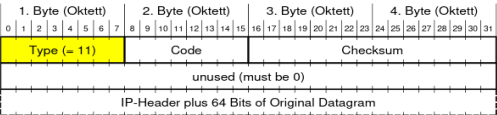
ICMP Time Exceeded

Wird auch für traceroute verwendet, durch inkrementelles erhöhen der TTL können die IP-Adressen der zwischen liegenden Router aus den ICMP Fehlern herausgefunden werden.

Linux command beispiel: traceroute -n -q1 <ip-addr>

Time Exceeded wird von einem Router oder Zielhost in diesen zwei Fällen gesendet:

- Router setzt TTL-Feld von 1 auf 0 → Paket wird verworfen und der Absender informiert (Code = 0)
- Zielhost kann ein fragmentiertes Paket nicht innerhalb nützlicher Zeit reassemblieren → Fragmente werden verworfen und der Absender informiert (Code = 1)



Transport Layer

schliesst Lücke zwischen IP & Applikation

UDP effiziente, schlanke Übertragung (verbindungslos)

TCP zuverlässige Übertragung (verbindungsorientiert)

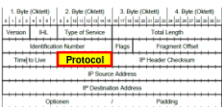
Vergleich Layer 2 und 4

Problem	Schicht 2	Schicht 4	Massnahmen bei TCP
Nachrichtenverlust	Pverlust = FER	Pverlust >> FER	Positives ACK
Telegramm-Reihenfolge	fix	kann variieren	Sequenznummern
Round Trip Time	konstant, µs .. ms	variabel, ms .. s	Adaptiver Retransmission Timeout
Überlast des Empfängers	kommt vor	kommt vor	Sliding Window mit dynamischer Fenstergrösse
Überlast des Netzwerks	direkt beobachtbar (Medium)	nur indirekt beobachtbar	Slow Start (Congestion Window)
Neustart von Hosts	direkt beobachtbar	nur indirekt beobachtbar	3 Weg Handshake, Initialisierung Sequenznr.

Kapselung

Applikationsdaten werden mittels Transport Layer in ein IP-Paket gekapselt.

Das gekapselte Protokoll wird im IP-Header im Feld Protocol angegeben



Feld 'Protocol' im IP-Header:

Decimal	TP-Protocol	Binary
0001	ICMP	0000'0001
0005	TCP	0000'0110
0017	UDP	0001'0001

Port Nummern

- einzelne Applikationen auf den Hosts werden durch Ports identifiziert (de-/multiplexen)
- jedes TCP/UDP Paket enthält source & destination port

Unterteilung der Port-Nummern in 3 Bereiche

- **Well-known Ports** (Bereich 1 - 1'023) sind durch IANA / IETF **standardisiert**
 - Well-known Ports sind meist zugleich für **UDP und TCP** reserviert
 - Well-known Ports sollen nicht für andere Anwendungen als die dafür vorgesehenen "missbraucht" werden
- **Registered Ports** (Bereich 1'024 - 49'151)
 - Reservierter Bereich für herstellerspezifische Applikationen
- **Dynamic / Private Ports** (Bereich > 49'152 bis 65'536)
 - können nach Belieben verwendet werden

Well-Known und Registered Ports kennt das Betriebssystem

- Windows: c:\Windows\System32\drivers\etc\services
- Linux/Unix: /etc/services

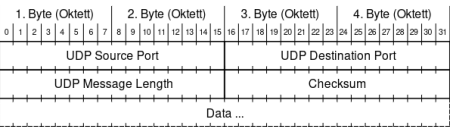
geläufige Well-known Ports

Port	Protocol	Beschreibung
20 / TCP	FTP - Data	File Transfer Protocol – Data
21 / TCP	FTP - Control	File Transfer Protocol – Control
22 / TCP	SSH	Secure shell
23 / TCP	Telnet	Telnet
25 / TCP	SMTP	Simple Mail Transfer Protocol
43 / TCP	WHOIS	Protokoll zum Abfragen von Internet-Domains und IP-Adressen
53 / UDP/TCP	DNS	Domain Name System
80 / TCP	HTTP	Hypertext Transfer Protocol (Web)
67 / UDP	BOOTPs / DHCPs	Boot Protocol (Server) / Dynamic Host Configuration Protocol (Server)
68 / UDP	BOOTPc / DHCPc	Boot Protocol (Client) / Dynamic Host Configuration Protocol (Client)
69 / UDP	TFTP	Trivial File Transfer Protocol
110 / TCP	POP3	Post Office Protocol Version 3
143 / TCP	IMAP4	Internet Message Access Protocol
443 / TCP	HTTPS	HTTP over SSL/TLS
465 / TCP	SMTPS	SMTP over SSL/TLS
993 / TCP	IMAP4S	IMAP over SSL/TLS
995 / TCP	POP3S	Post Office Protocol Version 3 over SSL/TLS

UDP - User Datagram Protocol

Verbindungslos Daten werden in UDP Datagramme eingefügt und direkt gesendet
Unzuverlässig keine Massnahmen gegen Verlust oder Vertauschen (gleich wie IP Pakete)

UDP Header



- **UDP Source Port** (16 Bits, 2 Bytes)
 - Identifiziert sendende Applikation
- **UDP Destination Port** (16 Bits, 2 Bytes)
 - Identifiziert die Applikation des Empfängers
- **UDP Message Length** (16 Bits, 2 Bytes)
 - Länge des Datagramms (inkl. Header) in Bytes
 - Maximale Länge eines UDP-Datagramms: 65'535 Bytes
- **UDP Checksum** (16 Bits)
 - Prüfsumme über einen Pseudo-Header, UDP-Header und Daten
 - Kann Null sein (keine Prüfsumme)
 - Pseudo-Header: IP Source- und Destination Address, Protocol Feld, Länge des Datagramms
 - Damit ist es u.A. möglich, fehlgeleitete Datagramme zu erkennen
 - Dies kann z.B. auf Grund eines Bit-Flip der Destination Address im Memory eines Routers auftreten

TCP - Transmission Control Protocol

Verbindungsorientiert vor Datenaustausch ist Verbindungsaufbau nötig
Zuverlässiger Verbindungsaufbau beide Endpunkte müssen den Verbindungsaufbau aktiv bestätigen

Hohe Zuverlässigkeit kein Datenverlust & richtige Reihenfolge
Vollduplex gleichzeitige, unabhängige Übertragung in beide Richtungen
Punkt zu Punkt immer direkt zwei Applikationen / kein Broad-/Multicast

TCP Nachrichtenaustausch

In beide Richtungen:

Sequence Numbers für **gesendete** Bytes, Position der Datenbytes im Gesamtstrom

- richtige Reihenfolge der Daten
- verlorene Daten erkennen

Acknowledge Numbers für **empfangene** Bytes (von der anderen Seite), Sequence Number der nächsten erwarten Bytes

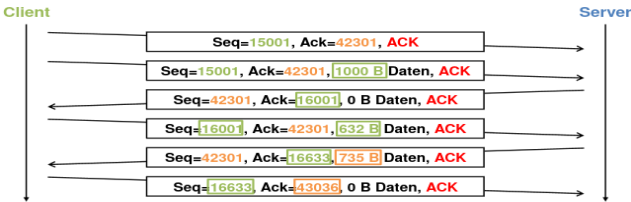
- Bestätigung korrekt empfangener Daten
- verlorene Daten erkennen

Flags steuern Verbindungsauf-/abbau, signalisieren Gültigkeit von Infos im Header und besondere Situationen

SYN/FIN Verbindungsauf-/abbau

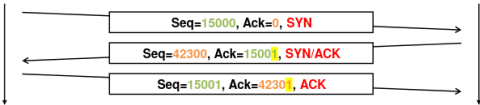
ACK Acknowledge Number im empfangen Segment ist gültig

PSH Daten sollen schnellstmöglich an Applikation weitergeben werden



TCP Verbindungsaufbau

- **Server** „horcht“ (LISTEN) auf einer bestimmten Port Nummer (z.B. 80 für einen HTTP Server)
- **Client** sendet Segment mit SYN=1 und zufälliger initialer **Sequenznummer a** (z. Bsp. 15'000) (ACK=0, weil Acknowledgement Nummer ungültig)
- Server bestätigt Sequenznummer mit **Acknowledgement Nummer a+1** (15'001) und **ACK=1** und wählt zufällige initiale **Sequenznummer b** (z. Bsp. 42'300) und setzt SYN=1
- Client bestätigt b mit **Acknowledgement Nummer b+1** (42'301)
 - Erstes Byte vom Client zum Server hat Sequenznummer a+1
 - Erstes Byte vom Server zum Client hat Sequenznummer b+1

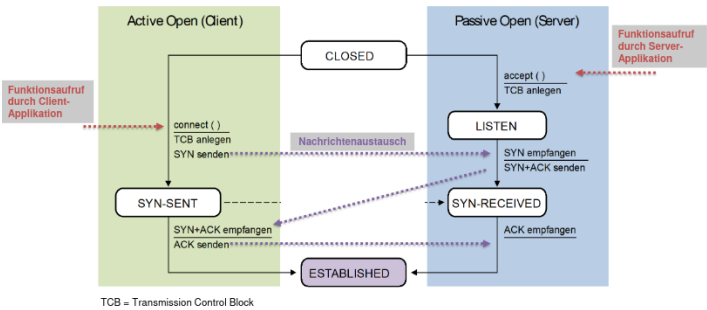


TCP Zustandsdiagramm

- TCP State machine existiert auf Client & Server
- Applikationen ändern state via listen(); connect(); close()
- die State Machines signalisieren sich Events mit den SYN, ACK, FIN flags

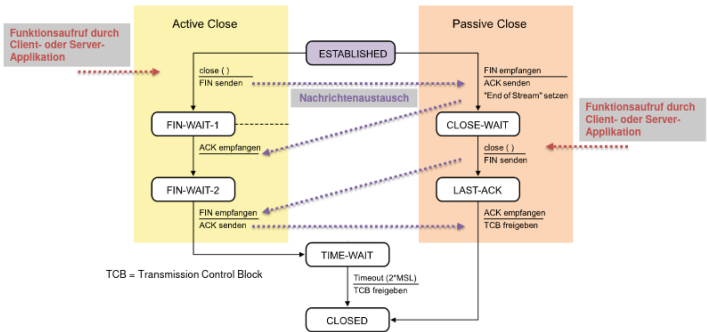
Verbindungsaufbau:

3- Wege Handshake



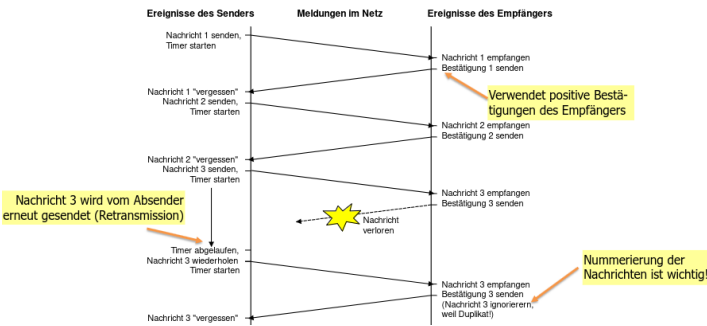
Verbindungsabbau:

Rollen (Active / Passive Close) beliebig



Verlorene Nachrichten Erkennen / Retransmission-Timeout (RTO)

Stop & Wait:



Vor/Nachteile:

- + keine Überlast beim Empfänger weil Sender zuerst warten muss
- Retransmission-Timeout(RTO) ist hoch variabel anhand von Netzwerkbedingungen. Zur Linderung misst TCP Round-Trip-Time (RTT) und berechnet gewichteten Mittelwert mit Streuung für idealen RTO
- viel Zeit mit warten verbracht

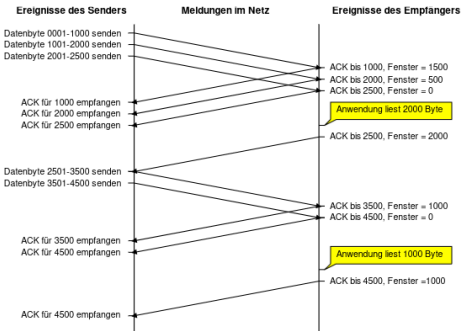
Berechnung des RTO anhand gemessener Round-Trip-Time:

- gewichteter Mittelwert SRTT (Smoothed Round-Trip-Time)
 $SRTT = (1 - \alpha) * SRTT + \alpha * RTT \quad | \quad \alpha = 0.125$
- Streuung RTTVAR, gewichteter Mittelwert der Abweichungen
 $RTTVAR = (1 - \beta) * RTTVAR + \beta * |SRTT - RTT| \quad | \quad \beta = 0.25$
- Retransmission-Timeout RTO
 $RTO = SRTT + 4 * RTTVAR$

TCP Flow-Control / Sliding Window

Weil bei Stop & Wait zuviel Zeit mit Warten/Netzwerklatenz verschwendet wird → mehrere Pakete direkt nacheinander schicken solange diese ins Fenster passen (& nicht durch Congestion Window limitiert).

- beide Seiten haben eigenes Fenster
- Fenstergrösse wird in Anzahl Bytes kommuniziert
- initiale Fenstergrösse wird bei Verbindungsaufbau mittgeteilt
- bei jedem ACK wird der verfügbare Fensterplatz (Bytes) mittgeteilt → Fenstergrösse = 0 heisst keine Daten mehr senden

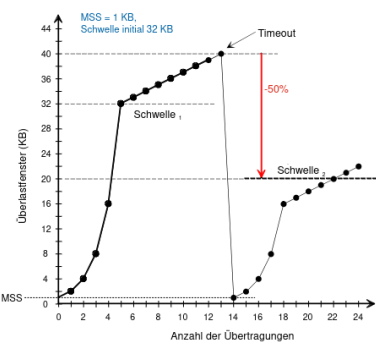


- Annahmen:**
- 2'500 Byte Empfangspuffer
 - 5'000 Bytes Daten
- Ablauf:**
- Fenstergrösse des Empfängers wird im **Window-Feld des TCP-Headers** übermittelt
 - Der TCP Header wird in Kürze behandelt
 - Wireshark gibt dieses als **Advertized Window Size** an
 - Sender-Applikation benötigt **nur einen Aufruf von listen()** für die gesamten 5'000 Bytes

TCP Congestion Control

Flow-Control schützt nur jeweils einen Empfänger, Congestion Control soll das Netz an sich vor Überlast schützen.

Mögliche Lösung: Slow Start (unter der Annahme, dass Überlast die meisten Fehler verursacht)

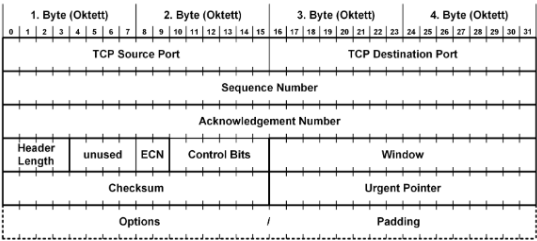


- Ein **Überlastfenster** (Congestion Window) limitiert **zusätzlich** die Grösse des Sendefensters
 - **Lokale Variable des Senders!**
 - Sendelimit: Das Kleinere der beiden Fenster
- Der Sender "testet" die Grenze aus
 - Algorithmen hierzu füllen Hunderte von Fachartikeln
- Original **Slow Start** Algorithmus
 - Fenstervergrößerung ausgehend von MSS (Maximum Segment Size)
 1. Exponentiell bis Schwelle
 2. Danach linear
 3. **Timeout:** Neudefinition der Schwelle und Neustart bei MSS

Kritikpunkte von Slow Start:

- "Sägezahnkurven" verschiedener Session neigen sich zu synchronisieren
- Bei Wi-Fi sind bitfehler das grösste Problem, nicht Überlast
- effektiv Stop & Wait bei kleinem Congestion Window
- für kurze Sessions immer langsam
- abhängig von Round-Trip-Time

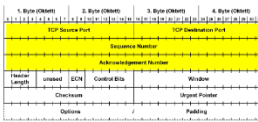
TCP Header



- TCP Header Länge: Mindestens **20 Bytes** plus mögliche Optionen (+ max. 40 Bytes)
- Eine TCP-Verbindung besteht aus **je einem Datenstrom in jeder Richtung**
- Header beinhaltet Information für die **"Vorwärtsrichtung"** (Sequenznummer etc.) und für die **"Rückwärtsrichtung"** (Acknowledgement Nummer, Window)

TCP Source/Destination Port (je 16 Bits)

- Bezeichnet jeweils die Ports auf Sender- und Empfängerseite
- Verbindungsaufbau: Destination Port bezeichnen den **gewünschten Dienst** auf der Serverseite



Sequence Number (32 Bits)

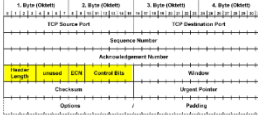
- Sequenznummer des **ersten Bytes** der Daten in der TCP-Nachricht innerhalb des gesamten Datenstroms
- Dies erlaubt dem Empfänger, die Daten richtig zu **ordnen**

Acknowledgement Number (32 Bits)

- Bezeichnet die Sequenznummer **des nächsten Bytes**, welches der Empfänger erwartet
- Ist die Acknowledgement Nummer n+1, so sind alle Bytes **bis und mit n** beim Empfänger korrekt und vollständig angekommen

Header Length (4 Bits)

- Länge des Headers in Double-Words (32-Bit Einheiten → Faktor 4)



ECN Flags (2 Bits) - Explicit Congestion Notification

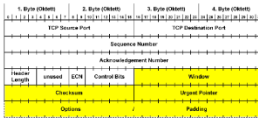
- Bit 8: **CWR**, Congestion Window Reduced (RFC3168)
- Bit 9: **ECE**, ECN-Echo (RFC3168)

Control Bits (6 Bits), für Verbindungsaufbau und -abbau und Angabe, welche Felder im Header eine Bedeutung haben

- **URG** **Urgent-Pointer** enthält gültigen Wert
- **ACK** **Acknowledgement Number** enthält gültigen Wert
- **PSH** **Push**: Empfänger soll Daten sofort ohne buffern an die Applikation weiterleiten z.B. bei der letzten TCP-Nachricht eines Web-Requests oder einer Reply
- **RST** **Reset**: Verbindung rücksetzen
- **SYN** **Synchronize**: Verbindung aufbauen
- **FIN** **Verbindung abbauen**

Window (16 Bits)

- Zeigt der anderen Seite die aktuell verfügbare **Puffergrösse im Empfänger** an (in Bytes) → so viele Bytes darf der Sender zur Zeit noch senden (ab der angegebenen Acknowledgement Nummer)



Checksum (16 Bits)

- **Prüfsumme** über einen Pseudo-Header, TCP-Header und Daten (wie UDP)

Urgent Pointer (16 Bits)

- Falls URG-Flag gesetzt: gibt Position (Byte Offset) von **"Urgent Data"** in den Daten an (z.B. für Interrupt Zeichen wie Ctrl-C)

Options (max. 40 Bytes)

- Häufigste Verwendung: **MSS (Maximum Segment Size)**, die empfangen werden kann