# Netze

Stern Topologie hat sich gegenüber Ring, Bus und Fully Mashed durchgesetzt

MAN: Metropolian Area Network

# Medien

c=300’000km/s 🡪 2/3 🡪 200’000km/s oder 0.2m/ns

Transatlantikdistanz: ca. 80-100ms / via Satellit 600ms

Kollisionen gibt es nur im Half Duplex

|  |  |  |  |
| --- | --- | --- | --- |
| **Thick Ethernet (Yellow Cable: 10Base5)** | **Thin Ethernet (Cheapnet: 10Base2)** | **Twisted Pair** | **Fiber** |
| Max 500m | Max 185m | Einige 100m bis km (mit Repeater)  10BaseT: 10’240m 100BaseT: 1024m 1000BaseT: 102.4m | Für grosse Distanzen |
| 10Mbit/s | | 10 – 1000Mbit/s | 2.4Gb/s bis Tb/s |
| Immer Halfduplex | | Shield und Unshielded | Übersee sind 8 Fasern verlegt (4 in jede Richtung) |
| Kollisionserkennung wenn Spannungsüberlagerung  Bei längeren Leitung würde das Signal zu stark gedämpft werden | | Kollisionserkennung wenn gleichzeitig gesendet und empfangen wird. Sendervorgang wird abgebrochen und JAM Signal wird gesendet. Danach exponentieller Backoff und danach erneuter Versuch 🡪 Viele CRC Fehler |  |
| Nur Negative Pegel: 0 bis -2V | | -0.2V bis +0.2V | Grössere Bandbreite  Weniger Dämpfung Längere Strecken |

Koaxialkabel benötigen einen Abschlusswiederstand, damit das Signal nicht reflektiert wird und mit sich selber kollidiert.

# Codierungen

|  |  |  |
| --- | --- | --- |
| **10Mbit/s** | **100Mbit/s** | **1000Mbit/s** |
| Manchester Codierung:  Flanken in Bitmitte   * Steigend = 1 * Fallend = 0 | 4B/5B mit MLT3 MLT3= Pegelwechsel bei 1 über 3 Pegel (-1V, 0V, +1V) | 8b/10b mit PAM5 PAM5: Pegelwechsel gemäss Mapping Tabelle über 5 Stufen (-2V, -1V, 0V, +1V, +2V) |
| 4 Adern | 4 Adern | 8 Adern |
| 0.1us oder 100ns | 0.01us oder 10ns | 1 Bit pro 1ns |

# Frequenzen

Telefonie: 300Hz bis 3400Hz 🡪 Frequenzbereich/Bandbreite von 3100Hz  
UKW Radio: 30MHz bis 30MHz  
Fernsehen: 6MHz  
Menschen hören bis ca. 15kHz (mittleren Bereich am empfindlichsten)

# Signale / Oszylloskop

Oszilloskop ist zum Messen und Analysieren von analogen Signalen  
Der Spektrum Analysator wird zur Anzeige verschiedener Frequenzanteile eines Signals verwendet

Bandbreite: Mengen an Daten pro Zeit (potenziell)  
Durchsatz: Menge an Daten pro Zeit (effektiv)  
Latenz: RTT als Zeit

Signalbandbreite (Bs): Differenz zwischen höchster und tiefster Frequenz  
Kanalbandbreite (Bk): Durchlassungsbereich eines Übertragungskanal 🡪 Je kleiner die Kanalbandbreite, desto schlechter ist das Signal: Wenn BK kleiner als Bs kommt es zu Bitfehler

Bautrate [Bd] = Anzahl Signalschritte pro Sekunde = Schrittgeschwindigkeit = Symbolrate  
  
Nyquist Rate / Abtastrate = r = 2 \* Bk

Abtasttheorem: Abtastrate muss mehr als doppelt so hoch Bandbreite des Signals sein

Kanalkapazität nach Shannon: C= Bandbreite[Hz]\* log2 (1+S/N)

Wird die Kanalbandbreite erhöht steigt die Kanalkapazität

Frequenz = Anzahl Schwingungen(Perioden) pro Sekunde 🡪 Frequenz [Hz] = 1 / Periodendauer [s]  
Pulsdauer = 1 / Geschwindigkeit [Bit/s] (ACHTUNG: Bei Manchester muss noch /2 gerechnet werden. Da Wechsel in Bitmitte)

Trigger Source: Von welchem Channel kommt das Signal  
Trigger Level: Y-Achse: Wie viele Volt  
Trigger Zeitpunkt: X-Achse (Zeit)  
Trigger Edge/Slope: Steigend, Fallend, beides  
MS/s = Millionen Sample pro Sekunde

Fourierreihe: Summe von ganzzahligen Vielfache der Grundfrequenz in Form von Sinus und Cosinusschwingungen

# Organisationen

ITU: Vergabe von Sendefrequenzen  
ICANN: Koordiniert Vergabe von Namen und Adressen  
IANA: Vergibt Namen und Adressen, sowie Well Known Ports

# Ethernet Berechnungen

Zusätzliche Bytes um MAC SDU zu übertragen: 12B Interframe Gap + 8Byte Premable/Interframe Gap + 14B MAC Header + 4 Byte CRC

Framerate (Frames/s): (Geschwindigkeit in Bit/s ) / (Framegrösse in Bit)  
Maximal mögliche Datenrate = SDU in Bit \* Framerate  
  
Datenmengen sind immer: 1024 Bytes  
Datenrate: 1000 Bit /s

Dauer berechnen: (Datenmenge \* 8 \* (1024)) [Bit] / Datenrate [Bit/s]

Maximal mit TCP Übertragbare Datenrate = Receive Window Size / RTT

# Multiplex

Nutzung gemeinsamer Übertragungskanäle durch mehrere Kommunikationspartner

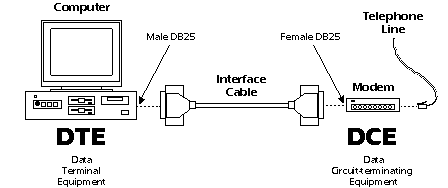
Raum (SDM): physisch getrennte Kabel/Funksignale  
Frequenz (FDM): Unterschiedliche Frequenzen  
Zeit (TDM): Nacheinander nutzen (ISDN, ADSL)  
Code (CDM): Unterschiedliche Codierungen (funkgesteuerte Zentralverriegelung bei Autos)  
Wellenlänge (WDM): Unterschiedliche Wellenlängen bei Glas

# Traffic

Streaming Traffic: Kontinuierlicher Verkehr  
Bursty Traffic: Stossweise, Ruckartiger Verkehr

# Serielle Schnittstellen (RS232)

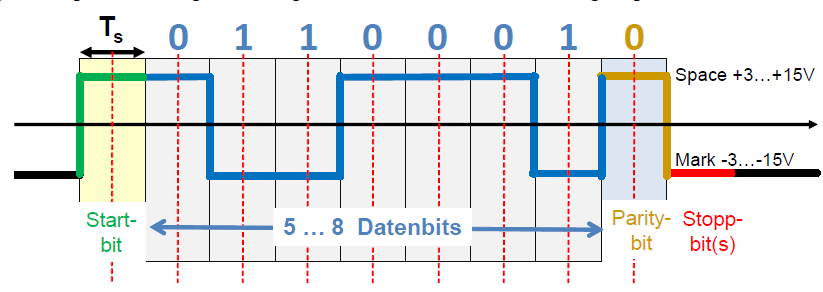
* DCE = Data Communication Equipment = Modem = Männlicher Buchse
* DTE = Data Terminal Equipment = Computer oder Terminal = Weibliche Stecker



* Vorläufer von USB (Zum Anschluss von Maus, Drucker, Modem, Router Konfiguration)
* DB9 = 9 polig und DB25 = 25polig
* Grundsätzlich ist eine Vollduplex-Verbindung möglich, da für Sendung und Empfang getrennte Datenleitungen zur Verfügung stehen.
* Least Significant Bit wird zuerst übertragen
* Es wird der NRZ Code verwendet
* Der Takt wird nicht mitgesendet. Empfänger berechnet Bitmitte anhand des Startbits
* Start und Stoppbits werden häufig auch Framing Bits genannt
* Startbit besteht immer aus einem Wechsel vom tiefen zum hohen Pegel (Trigger Positive Flanke)
* Stoppbits (ein oder mehrere) haben immer einen tiefen Pegel
* Kontrollbits sind so gewählt, dass es zusammen mit den Datenbits eine gerade oder ungerade Anzahl Einsen gibt. (Prüfsummen / Parity Bit)
  + Even: Summe aller 1 Bits gerade = 0
  + Odd: Summe aller 1 Bits ungerade = 1
  + No Parity: Kein Paritätbit
  + Space Parity: Partitätsbit konstant 0
  + Mark Parity: Paritätsbit konstat 1

**Mark**: Tiefer Pegel = 1 🡪 -3V bis -15V

**Space**: Hohe Pegel = 0 🡪 +3V bis +15V



### Pins:

|  |  |
| --- | --- |
| **DB9** | **DB25** |
| 2 = Rx Receive  3 = Tx Transmit  5 = Signal Ground | 2 = Tx  3 = Rx  7 = SG |

# WLAN: 802.11

WLAN Standard sind b 🡪 g 🡪 n 🡪 a/c

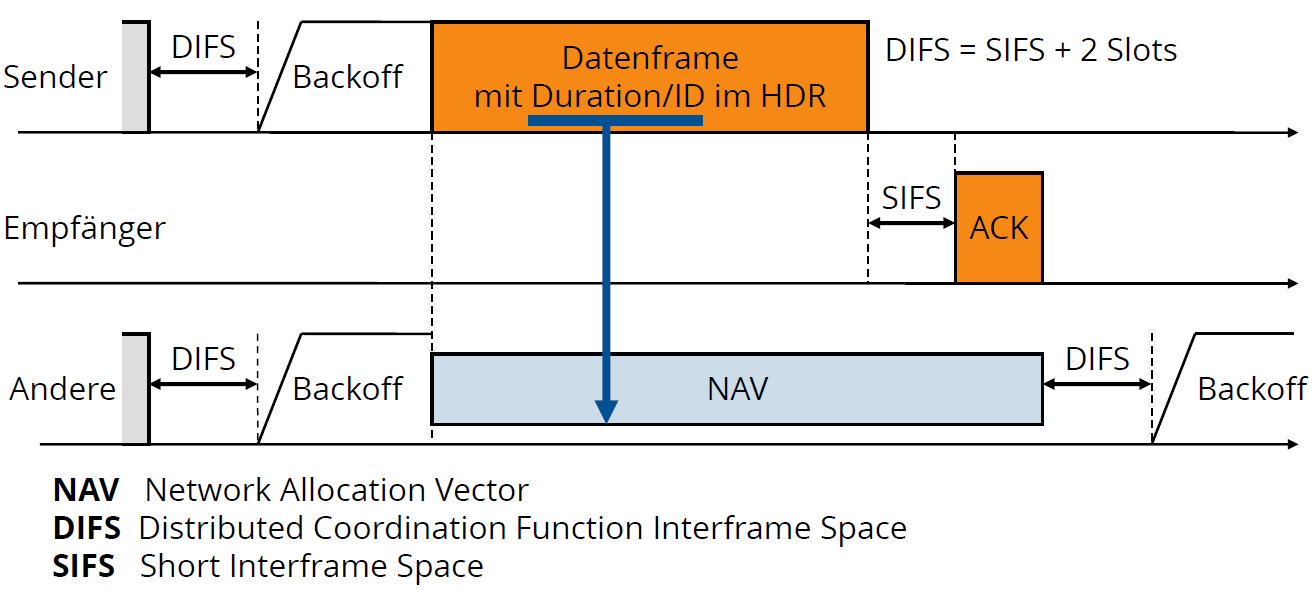
SSID = Service Set Identifier  
BSSID = Basic Serivce Set Identification = MAC Adresse des Access Points  
BS = Basic Service Set = Access Point mit Stationen   
DS = Distribution System  
ESS = Extended Service Set = Mehrere BSS mit gleicher SSID verbunden via DS

WLAN ist immer Half Duplex. Ein AP kann mehrere 802.11 Standards betreiben

Hidden Node Problem: Nicht alle Station können sich hören

* Lösung: Übertragung mit RTS/CTS oder Sendeleistung reduzieren

## CSMA/CA (Carrier Sense Multiple Access Collision Avoidance)



RTS/CTS wird nur für grosse Datenframes verwendet   
Wenn das Medium länger als ein SIFS frei, kann gesendet werden  
Der NAV wird mittels der Duration im Datenframe aktualisiert

Bei RTS/CTS wird nach dem Backoff ein RTS gesendet. Der Empfänger antwortet mit einem SIFS und mit einem CTS. Danach gibt’s beim Sender auch ein SIF und die Daten werden gesendet.

Bei einer Kollision wird ein EIFS (SIFS + DIFS + ACKEmpfänger) lang gewartet bis die Daten erneut gesendet werden. Wenn eine Kollision aufritt wird eine Random Zeit gewartet. Bei mehreren Kollisionen wird der Backoff exponentiell erhöht

## Passive Mode:

* Router senden Beacon Frames an alle Stationen im Umkreis, mit Infos über den Access Point
* Ein Beacon Frame enthält SSID, MAC des AP, Unterstütze Datenraten und verfügbare Funkkanäle
* Station hört Kanäle ab um Access Points zu finden

## Active Mode:

* Mindestens 6 Frames
* Stationen senden Probe Request um Access Point zu finden
* Access Point antwortet mit Probe Response
* WPA/WPA2 erfolgt erst nach Assoziierung (WEP verwendet noch den Authentication Request und Response. Dieser ist jedoch veraltet)
* Während der Association wird die Verbindung effektiv hergestellt
  + Eine Station kann genau bei einem AP assoziiert sein
  + Es gibt eine Reassociation falls innerhalb eines ESS von BSS zu BSS verschoben wird

|  |
| --- |
|  |
|  |

## 2.4 GHz Band:

* Kanal 1,6,11 überlappen sich nicht
* 22 MHz breites Signal
* 5 MHz Abstand zwischen den Signalen
* Gute Ausbreitungseigenschaften
* Viele Störquellen (Mikrowellen, Babyphone, Bluetooth)

## 5GHz Band:

* Kanal 36, 40, 44, 48, 52, 60, 64 überlappen sich nicht (indoor usage)
* Wird auch für Radaranwendungen benutzt, diese dürfen nicht gestört werden

## Datenrate:

* Anten ausrichten, Mehrere Antennen, Kanäle optimal legen
* Die maximal erreichbare Datenrate hängt von folgenden Punkte ab
  + Signal bzw. Kanalbandbreite
  + Signal zu Geräuschleistung (S/N)
  + Anzahl parallel nutzbare Ausbreitungswege (MIMO)
* Die Signalleistung beim Empfänger hängt von folgenden Punkte ab
  + Sendeleistung des Senders
  + Verdünnung mit zunehmender Distanz (^D2)
    - Verdoppelt man die Frequenz so vervierfacht sich die Dämpfung
  + Dämpfung durch Wänder, wärmeisolierte Fensterscheiben)

## MIMO: Multiple Input Multiple Output

* Anzahl parallel nutzbare Ausbreitungswege (Mehrere Antennen)

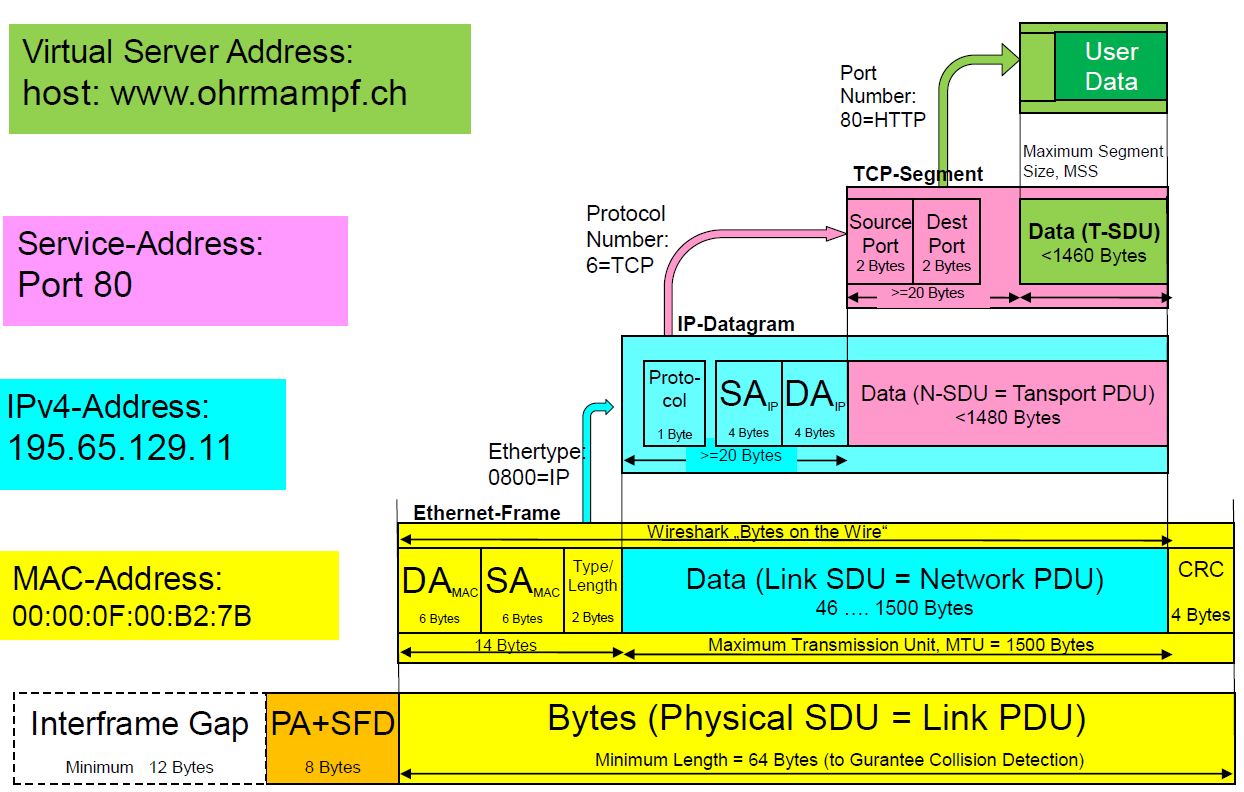
# CSMA/CD (Carrier Sense Multiple Access Collision Detection)

Carrier Sense Multiple Access / Collision Detektion

* Jede Station kann jederzeit versuchen, Pakete abzuschicken.
* Carrier Sense
  + Vor dem Zugriff aufs Medium wird überprüft, ob nicht eine andere Übertragung im Gange ist.
* Multiple Access
  + Falls eine andere Übertragung im Gange ist, wird gewartet; sonst beginnt man mit dem Absenden des Pakets.
* Collision Detection
  + Falls zufällig mehrere Stationen mit Senden begonnen haben, kollidieren ihre Pakete. Dies wird aber von den sendenden Stationen detektiert und der Sendevorgang wird abgebrochen. (JAM Signal)
  + Collision Resolution: Nach dem Zufallsprinzip (Binary Exponential Backoff) wird bestimmt, in welcher Reihenfolge nach Kollisionen nochmals gesendet wird.

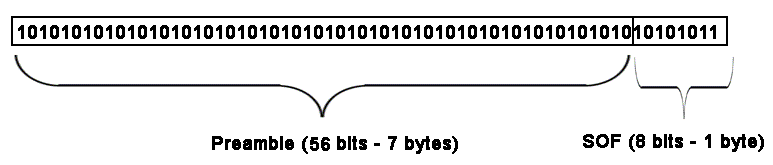
# OSI Schichtenmodell

PDU (Protocol Data Unit): Kompletter Layer (inkl. Header), der an den unteren Layer gegeben wird  
SDU (Service Data Unit): Layer ohne Header oder PDU des übergeordneten Layer



# Layer 1: Physical / Bits

Die Preamble dient der Bit Synchronisation



## Hub / Multiport Repeater:

* Bilden eine grosse Kollisionsdomäne
* Nur Half Duplex möglich
* Weiterleitung an alle Teilnehmer
* Signal wird verstärkt
* Nur gleiche Datenraten aber unterschiedliche Kabeltype
* Bei Vielen Kollisionen schaltet der Hub den Port ab
* Wird eine Kollision detektiert sendet er auf alle Ports ein JAM Signal

# Layer 2: Data Link / MAC / Frames

PDU muss mindestens 64 Byte lang sein. Ansonsten wird mit Padding aufgefüllt. Es werden somit Kollisionen erkannt

MAC SDU oder Network PDU darf maximal 1500Bytes lang sein, danach wird auf Layer 3 (Network) fragmentiert

Bei einem CRC Fehler wird das Frame verworfen

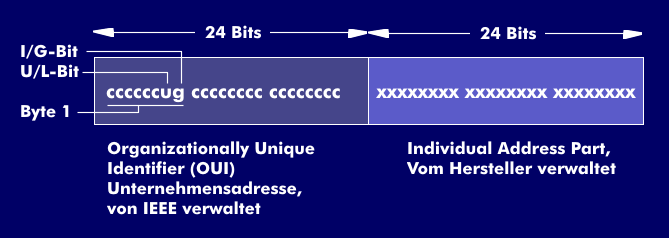
Link Integrity Pulse: Sind zwei Stationen miteinander verbunden (Normal Link Pulse)  
Fast Link Pulse: Speed und Duplex Autonegotiation  
Für Autonegotiation müssen beide Endstation auf Autonegotiation eingestellt sein. Ansonsten ist der Default Half Duplex. Speed Mismatch ist nicht möglich!

## Type/Length:

E-Type: Ethernet II 🡪 0x0800 (2048): IPv4 und 0x86DD (34525) IPv6  
Length: IEE802.3: 0-1500Bytes (1500 = 05DC)

## **MAC Adresse:**

48 Bit lang / 6 Byte = 3 Byte OUI (Organisationally Unique Identifier) 🡪 wird von der IEEE zugeteilt



1 Bit (Least Significant Bit): Individual/Unicast = 0 oder Group/Multicast = 1  
2 Bit (Least Significant Bit): Global/Universal = 0 oder Locally Administrated = 1

## ARP (Address Resolution Protocol)

* Liefert innerhalb der Broadcast Domain, zu jeder IP die passende MAC Adresse
* Anfrage an Broadcast «Who has IP». Die Antwort wird dann im ARP Cache des Senders gespeichert
  + Sender des ARP Reply trägt Sender des ARP Requests ebenfalls ein
  + Wenn die gesuchte IP nicht im selben Subnetz, wird die Anfrage vom Client direkt an das Default Gateway gesendet
  + ARP-Request erstellt somit einen Eintrag im Zielsystem

## Switch / Multiport Bridges:

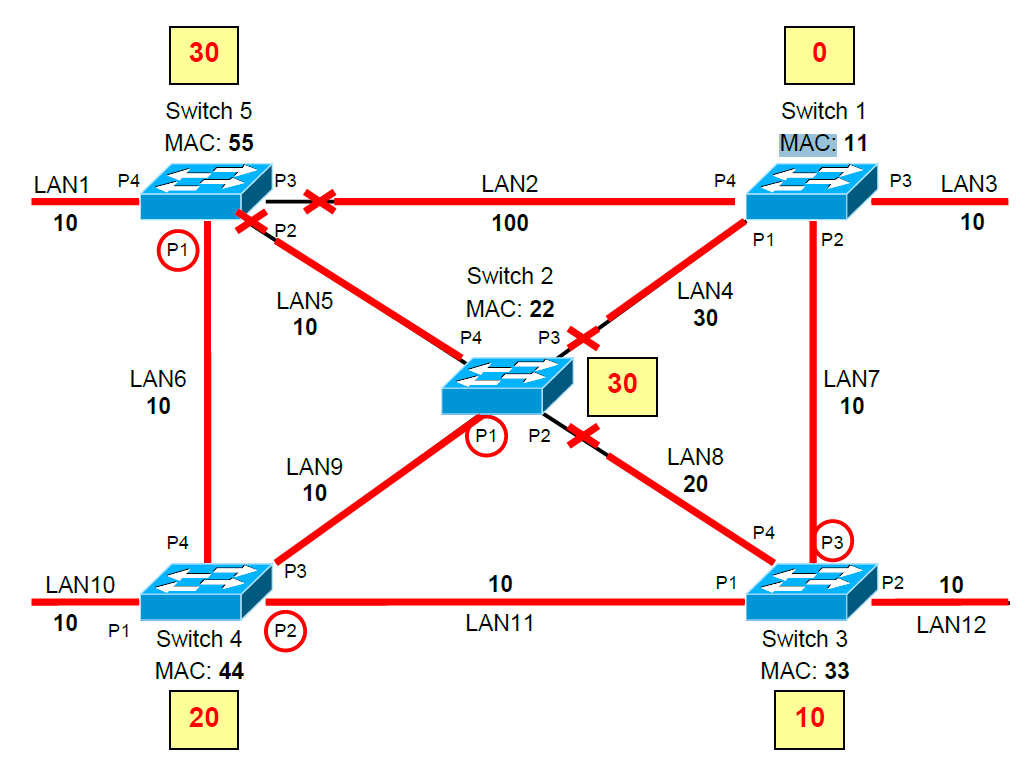
* Trennen Collisionsdomains ab
* Bilden eine grosse Broadcast Domain
* Unterstützt unterschiedliche Datenraten und Speeds
* Leiten Pakete anhand der MAC Destination Addr weiter (gem. Forwarding Table)
* Falls nicht in Forwarding Tabelle 🡪 Flooding an alle (MAC Broadcast: FF:FF:FF:FF:FF:FF)
* Switches brauchen für ihre Grundaufgaben (weiterleiten & filtern) weder eine MAC noch IP-Adresse
* LLDP (Link Layer Discovery Protocol) dient zur Veröffentlichung von Funktion und Konfiguration von Switches
* Link Aggregation bezeichnet die parallele Bündelung von LAN- Interfaces. Dies dient der Steigerung des Durchsatzes (Alle Ports gleich Duplex/Speed)
* Rapid Spanning Tree Protocol: Alternative Pfad wird zuerst berechnet. Danach wird der neue Baum innert < Sekunde aufgeschalten
  + Voraussetzung: Jeder Port muss als
    - Edge Port
    - Point to Point Port
    - Shared Port
  + definiert sein
* Multiple Spanning Tree Protocol: Erweiterung RSTP: Mehrere Spanning Tree pro VLAN
* VLAN: Mehrere Port als separates Netz zusammenfassen
  + Tagged: Gruppierung über das Tag Feld im Ethernet Header auf Layer 2
* Port Security: Port für eine MAC-Adresse reservieren. Lernmodus auch möglich

## Switch Forwarding:

Die Forwarding Tabelle wird mit der MAC Source Adresse und der Switch Portnummer aufgebaut

* Early Cut 🡪 Nur MAC-Dest Addr
* Cut Through 🡪 MAC Destination und Source Addr werden gelesen
* Fast Forward 🡪 MAC Destination, Source Addr, Length und Teil der PDU
* Store and Forward 🡪 komplettes Frame wird gelesen

## Spanning Tree:

* Root Switch: Niedrigste MAC/Bridge ID und höchste Priorität (falls vorhanden)
* BPDU (Bridge Protocol Data Units): Werden verwendet um den Spanning Tree aufzubauen
* Aktiviere die Links mit der kürzesten Verbindung (gem. Kosten) zu Root und bestimme Root Ports und Designated Ports pro Segment. Die Restlichen Port werden geblockt. (Jene mit der höchsten Bridge ID/Port Nummber)
* Wenn ein geblockter Port nach 20s (default) keine BPDU’s mehr empfängt wechselt er in der Forwarding Mode
* 

## Link Aggregation

Link Aggregation ist ein Verfahren zu Bündelung mehrere physikalischer LAN Schnittstellen zu einem logischen Kanal

# Layer 3: Network / Datagrams

Header ist 20Bytes lang  
Datagrams können mittels dem Header Feld Differentiated Service priorisiert werden  
Mit der Identification Number werden zusammengehörige Fragmente erkannt

## Protocol ID:

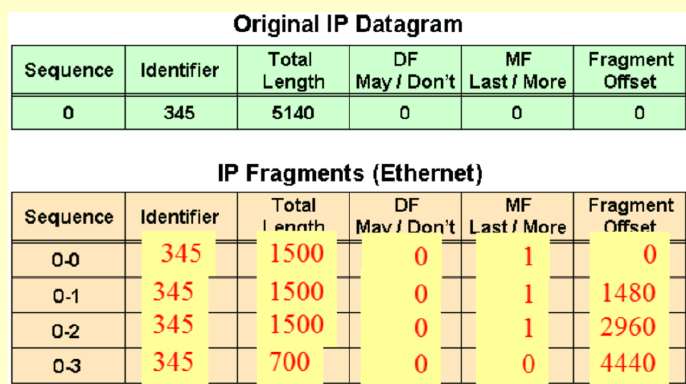
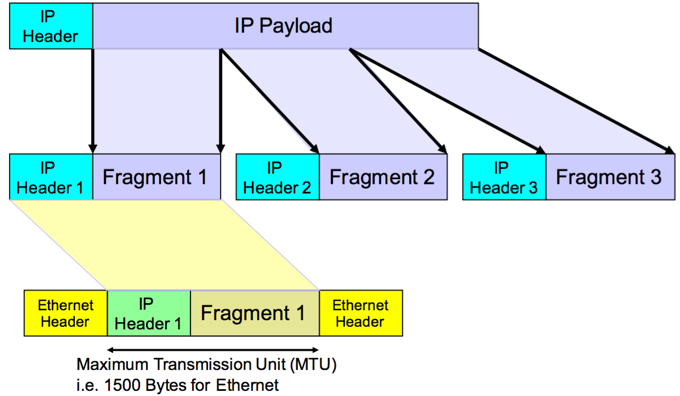
* TCP = 6
* UDP = 17
* ICMP = 1

MTU = Maximum Transmission Unit = MAC SDU = max. 1500Bytes 🡪 Stellt sicher, dass die Stationen den Kanal nicht zu lange benutzen (bei Half Duplex)

## Router / Layer 3 Switch:

* Trennen Broadcast Domains ab (Leiten den Broadcast nicht weiter
* RIP, OSPF
* Speichern Datagrams in Buffer / Warteschlange und leiten gemäss dem Netzteil der IP an korrekten Port weiter (IP-Destination Addr)
* Wenn der Buffer voll, werden Datagrams verworfen
* Verändert den IP Header nur bei NAT/PAT, zählt TTL runter und passt IP Checksum an
* Verwirft Pakete mit TTL = 0

## Fragmentierung:

* IP Payload wird in Fragmente + Header unterteilt
* 
* 

## NAT/PAT (Network Address Translation):

* Wegen Knappheit von öffentlicher Adressen
* Versteckt interne Netzwerk Strukturen
* Mapping wird via NAT Tabelle realisiert
* Es existieren zwei Arten:
  + SNAT (Source NAT)
    - Interne Source IP wird durch öffentliche IP des Routers ersetzt
    - Interner Source Port wird durch freien Port des Routers ersetzt
  + DNAT (Destination NAT)
    - Port Forwarding: Mehrere Dienste unter einer public IP
* Bei PAT werden die Portnummern zusätzlich gespeichert
* Bei ICMP wird der Identifier genommen (da Port Nummer nicht vorhanden)

Probleme bei NAT/PAT:

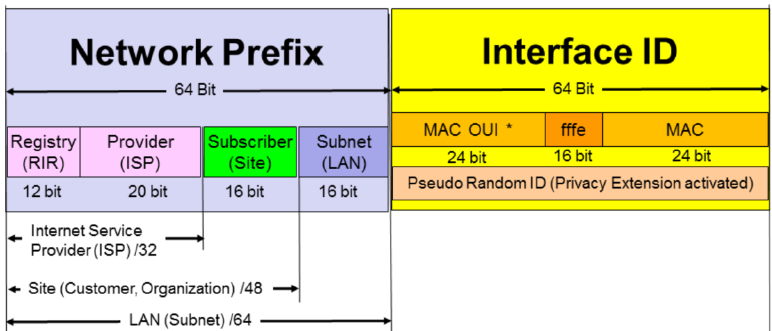
* Vermischung von Layer 3 und 4
* ICMP hat keinen Port
* FTP im aktive Mode
* IPv6

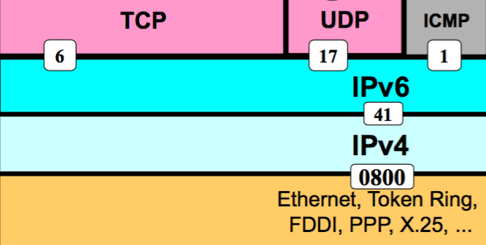
## IPv4:

* 32 Bit lang, 4 Byte lang
* Unterstütz hierarchische Adressierung
* Das Klassensystem ist veralten und wurde durch CIDR (Classless Inter Domain Routing) ersetzt
  + A = 0
  + B = 10
  + C = 110
  + D = 1111
* Non Routable Addresses
  + 10.0.0.0/8 (Class A)
  + 172.16.0.0 / 12 (Class B)
  + 192.168.0.0/16 (Class C)

## IPv6:

* 128 Bit lang, 16 Byte lang
* Non Routable
  + Loopback: ::1/128
  + Multicast: ff00::/8



* Neighbor Solicitation zu Bestimmung der umliegenden MAC Adressen
  + Meldung an Multicast «Bin ich unique»
* Neighbor Advertisement: Anwort auf Solicitation
* Router Solicitation: ICMPv6 Nachrichten um besten Router zu finden
* Router Advertisements: ICMPv6 vom Router für Autoconfiguration
* Nullen können weggelassen werden. 0-Quartetts können durch :: ersetzt werden
* Network Prefix ist 64 Bit und die Interface ID ist 64 Bit lang
* Privacy Extension: Random ID anstatt MAC Adresse
  + Wenn FF:FE in der Mitte eingeschoben: MAC Adresse sichtbar bis auf ein inverted Bit (7Bit des ersten MAC Blocks wird invertiert)
* Übergang IPv4 nach IPv6
  + 
  + Dual Stack: Unterstützung beider Protokolle
  + Teredo Adapter (Tunneling): IPv6 in IPv4
  + 6to4 Relay Servers

## Subnetting

Kleinstmögliche Subnetzmaske berechnen:

1. IP Adressen in Binär umrechen
2. Bis zum Bit gehen, dass sich ändert -1 = /Subnetzmaske

# Layer 4: Transport / Segments

Header ist 20Bytes (TCP) oder 8 Bytes (UDP) gross

MSS = Maximum Segment Size = Network SUD = 1460 Bytes oder 1472 Bytes

## Ports:

Well Known: 0-1023  
Registered: 1024 – 49’151  
Dynamic/Private Ports: 49’152 – 65’535

|  |  |
| --- | --- |
| **TCP** | **UDP** |
| FTP (20,21) Telnet (23)  SSH (22)  DNS (53)  POP3 (110)  IMAP(143)  SMTP (Simple Mail Transfer Protocol) (25) HTTP (HyperText Transfer Protocol) (80)  HTTPS (443) | TFTP (69)  DNS  NTP (Network Time Protocol)  LDAP  NBNS (Netbios)  SNMP (Simple Network Management Protocol) (161)  RTP (Real Time Transport Protocol) VoIP |

## ICMP (Internet Control Message Protocol)

ICMP dient zum Management von Netzwerk Layer Verbindungen  
ICMP Pakete können über die Nicht-Erreichbarkeit von Servern informieren  
ICMP Echo Request Header ist 8 Bytes gross

## TCP:

* Verbindungsorientiert (kein Broadcast)
* Zuverlässige Zustellung (Seq. Nr und Acknowledgenummer)
* Flusskontrolle / Sliding Window 🡪 Window Feld
* Verbindungsaufbau / abbau (SIN, ACK, FIN Flag)
* Options Feld: Wird verwendet um den Window Scaling Factor zu übertragen
* Flow Control beim Empfänger und Congestion Control beim Sender
* Flow Control
  + Pufferung der Segmente vor Weitergabe an Layer 5-7 beim Empfänger
  + Sortierung der Segmente
  + Receive Window Size dem Sender mitteilen
* Definierte Reihenfolge
* Flusskontrolle
* Header ist typischerweise 20 Byte lang
* 3-Way Handshake
* Das Window Feld im Header definiert wie viel Byte vom Empfänger noch aufgenommen werden können (max. 65'000 Bytes)
* Geht ein Segment verloren, wird nach dem Retransmission Timeout (RTO) das Segment noch einmal gesendet
* Die RTO ist die Zeit während der ein Sender auf ein ACK wartet bevor er das Segment erneut sendet(default ca. 3s)
  + Die RTO hängt von der Roundtrip Time ab und passt sich immer wieder an

## ACK:

* Fast Retransmit: Nach 3 Duplicate ACKS (Zuletzt korrekt empfangenes ACK nochmals senden) wird ein Segment erneut gesendet. Dabei wird die RTO nicht beachtet
* Selective ACK (SACK): Nur das fehlende Segment wird neu angefordert via Option Feld im TCP Header
* Delayed ACK: Erhöhung der Performance: ACK wird erst nach Ablauf des delayed ACK Timer (0.5s) für alle bisher erhaltenen Segmente gesendet. (1 ACK für mehrere Segmente)
* Cumulative ACK: Nur jedes zweite Segment wird bestätigt. Die höchste durch ACK bestätigte SEQ-Nr gilt
* ACK Clocking ist das Automatische Anpassen der TCP Datenrate an den langsamsten Link

## 3-Way Handshake:

|  |  |
| --- | --- |
| ../../../../../Desktop/Screen%20Shot%202016-01-14%20at%2008.5 | Empfänger muss im passive open Zustand sein   1. Sender sendet Segment mit SYN Flag und rnd SEQ-Nr (x) 2. Empfänger sendet Segment mit SYN & ACK Flag zurück. Die ACK-Nr ist dabei 1 grösser als die empfangene SEQ-Nr (x+1). Die gesendete SEQ-Nr ist ebenfalls eine rnd Zahl (y). 3. Sender schickt ein ACK zurück mit inkrementierter SEQ-Nr (x+1) und inkrementierter ACK-Nr (y+1) |
| 1. Sender sendet ein Segment mit FIN Flag und SEQ-Nr. Z 2. Empfänger sendet Segment mit ACK, ACK-Nr z+1 und SEQ-Nr s 3. Empfänger sendet Segment mit FIN Flag und SEQ-Nr s+1   Sender sendet Segment mit ACK Flag, ACK-Nr s+2 und SEQ-Nr z+1 | ../../../../../Desktop/Screen%20Shot%202016-01-14%20at%2009.3 |

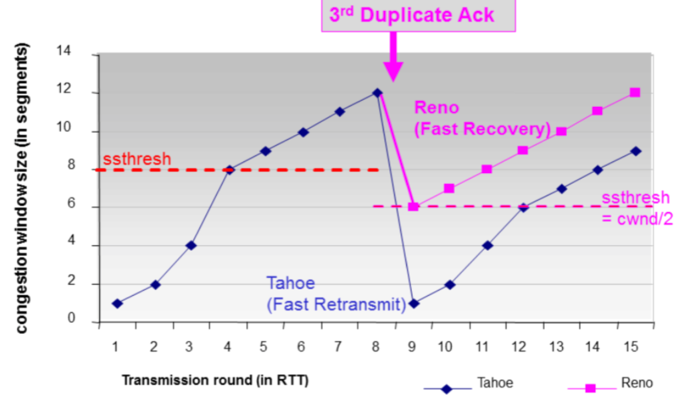
## TCP Performance:

* Interframegap 12 Byte zwischen Paketen
* TCP Durchsatz hängt von Payload Grösse, Receive Windows Size und Round Trip Time und Congestion Control Verfahren ab.
* Durchsatzrate = Maximum Segment Size/RTT) \* (1/sqrt(Loss)
* Durchsatz im Idealfall = TCP Payload (1460B) / Komplette Framegrösse (1538B) = 95%

## Sliding Window

Receive Window Size gemäss ACKs

## Congestion Control / Slow Start:

* Verhindert, dass der Sender die Kapazität des Netzwerks überlastet
* Exponentielle Vergrösserung bis Segmente verloren gehen, danach linear
* Es wird auf jedes ACK gewartet bis vergrössert
* Wenn die Receive Window = 0 wird der Slow Start neu initialisiert
* 
* Solange kein Duplikate ACK 🡪 Congestion Window erhöhen

## Congestion Avoidance:

* Exponentiell bis Congestion Threshold, danach linear
* Bei 3 Duplicate ACK zurück zum Anfang oder halbieren des Conscession Window

## UDP:

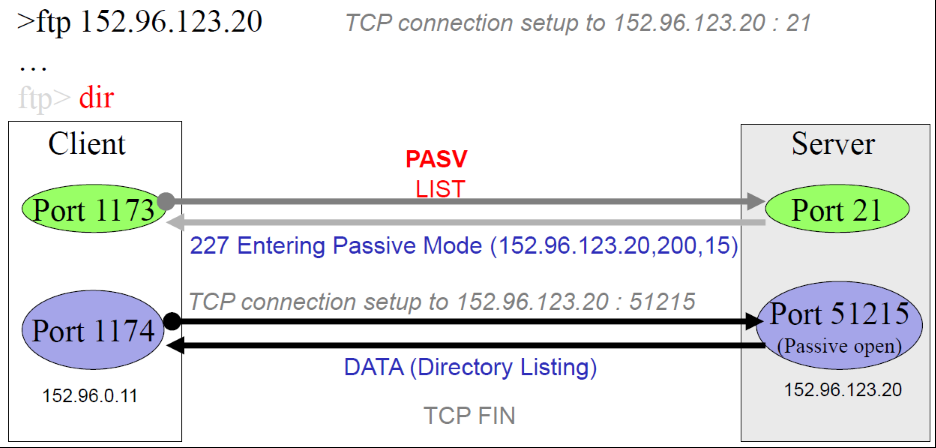
* Keine Flusskontrolle
* Verbindungslos
* Kein Lastausgleich
* Keine garantierte Reihenfolge
* Wenn eine Antwort ausbleibt wird einfach noch einmal ein Request gesendet (((auf oberem Layer
* Erlaubt Broadcast und Multicast
* Header ist 8 Byte lang
* Source Port optional

# Layer 5-7: Application Layer / Messages

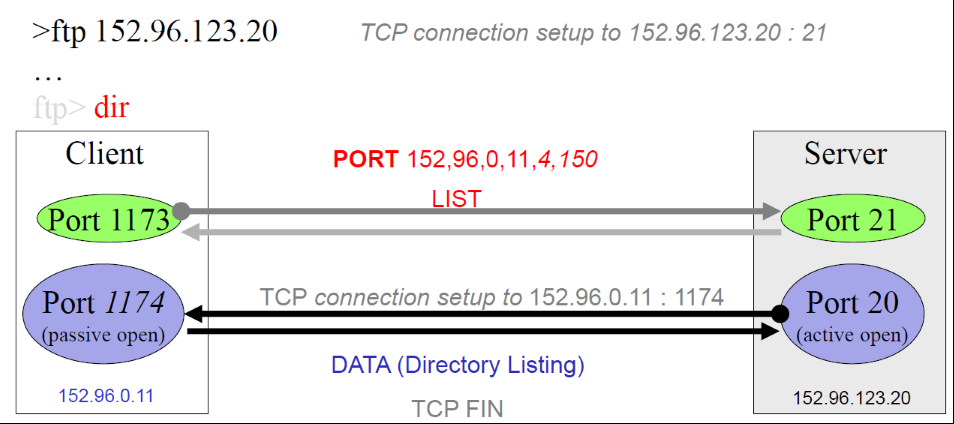
## TFTP (69)

* Keine Authentifizierung
* UDP
* VoIP Telefone holen sich Config via TFTP
* Client sendet read Request an Server auf Port 69
  + tftp -i tftp.test.ch GET /samples/test.txt
* Server wählt dann einen Port für die restliche Übertragung und sendet erstes Packet (512B) zurück mit Block Nr 1 (falls das File existiert)
* Client sendet Acknowledgement für Block 1
* Server sendet zweites Packet mit Block 2 usw.
* Letztes geschicktes Packet vom Server muss <512B sein 🡪 Flag für End of File

## FTP

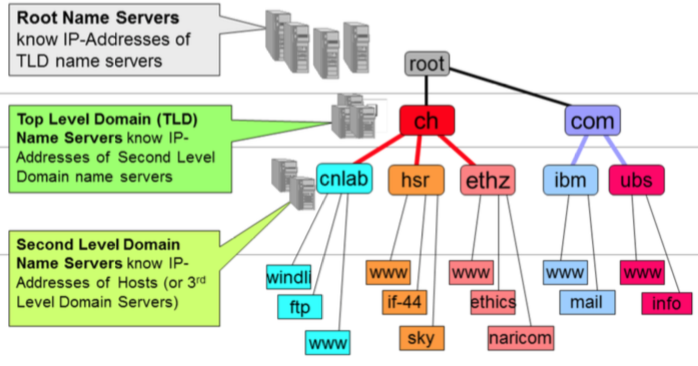
* TCP
* Username und Passwort
* Server Port 21 für Befehle
* Max: 512 Bytes 🡪 Ansonsten End of Data
* Passiv:
* 
  + Server lauscht auf Port 21 (Commands)
  + Client verbindet zu Port 21 und öffnet somit zweites Port auf Server für die Daten
    - IP,200,12 🡪 200\*256 + 15 = 51215 = Server seitiger Daten Port

### Aktiv: (Probleme mit NAT)

* 
  + Client verbindet zu Port 21 (Commands)
  + Server öffnet Datenverbindung zu Client auf übergebenem Port
    - IP, 4, 14 🡪 4\*256 + 15 = 11741 = Client seitiger Daten Port

# DNS: Domain Name System (53):

-nslookup [host] > set type=NS/MX/A/CNAME



* RIR: Regional Internet Registrars (RIPE für Europa)
* LIR: Local Internet Registrars (SWITCH für Schweiz)
* FQDN: Fully Qualified Domain Name
* G TLD: Generic Top Level Domain (com, org, gov, mil, edu)
* N TLD : New Top LEvel Domain (info, onion, xxx, biz)
* CC TLD : Country Code Top Level Domain ( ch, de, uk)
* Es gibt kein Recht auf eine Domain : Domain Grabbing/Cybersquatting/Typosquatting

A: IPv4 Address Eitnrag  
AAAA: IPv6 Address Eintrag  
MX: Mail Server  
NS: Name Server  
LOC: Location (GPS)

Nicht autorisierte Antwort: Wenn die Antwort vom Caching Server zurückgeliefert wird und nicht bei den dafür zuständigen Top Level DNS Server angefragt werden

UDP: Client an DNS / TCP: DNS an DNS (Zonentransfer)

## **Resolve** Process:

* .host File
* DNS Cache beim Client (DNS Spoofing)
* Default Name Server abfragen
* Anfrage an Root Server

Recursion: DNS Server holt sich nötige Informationen direkt bei Top Level DNS

Authentication: Nur «fälschungssichere» Antworten

TTL: Wie lange bleibt der DNS Eintrag im lokalen Cache beim Client

## CDN: Content Distribute Network:

* Für die Reduktion der Antwortzeiten eines Webservers
* Im DNS Response wird nahegelegenen Akamia Server geliefert mit gesuchter Ressource

## HTTP:

1. DNS Lookup Webserver
2. TCP Connection
3. http Request / Response
4. Render Response
5. Close Connection

Benötigt mindestens 3 RTT 🡪 3TT + File size / Data rate

|  |  |
| --- | --- |
| ../../../../../Desktop/Screen%20Shot%202016-01-16%20at%2009.2 | **1.0**  - Nach jeder Übertragung musste die Verbindung erneut aufgebaut werden  **1.1**  - Persistente Verbindungen: nur eine TCP Verbindung für alle Inhalte  - Pipelining: Nur ein http-Request für alle Inhalte  **2.0**  SPDY („Speedy“)  Verkürzt Ladezeiten durch Komprimierung der Daten, Multiplexing, Priorisierung (kein FIFO mehr), Verschlüsselung, Push (dh Server kann Daten senden, ohne das Client requesten muss) |

Persistance Connection: (Keep-Alive im Connection Type ab v1.1 🡪 Server lässt Verbindung offen)

* Ohne Pipelining: 1 TCP Connection: 1 HTTP Request pro Objekt
* Mit Pipelining: 1 TCP Connection: 1 HTTP Request für alle Objekte
* Parallel Pipelining : Mehrere TCP Connections und ein HTTP Request für alle Ressourcen
  + Eine TCP für HTML Seite, mehrere parallele TCP für einen HTTP Request

|  |  |  |
| --- | --- | --- |
|  |  |  |

E-Tag: Ist ein Hash über die Ressource. Wenn nicht geändert, wird das Objekt nicht mehr erneut übertragen

Conditional GET: Client Side Cache: If-modified-since: <cache-date> 🡪 Request an Server 🡪 Server antwortet mit 304 Not modified oder 200 OK

## Answer Codes:

* 2xx = Success
* 3xx = Redirect
* 4xx = Client Error
* 5xx = Server Error

## Unterschied GET und POST:

* GET wird im Header (LOCATION) übertragen und POST im Body

## Virtuelle Server:

* Mehrere Webseiten auf einem Server/IP hosten
* Host im HTTP Header Feld

## HTTP Proxy:

* Reverse Proxy
  + Transparent für Client
  + Entlastung des Servers
* Forward Proxy
  + Cache Funktion
  + Filter (Sperrung von Webinhalten, Suche nach Malware)
  + Anonymisierung (Server sieht nur Proxy nicht Client)
* Proxy Variationen
  + Transparent: leiten tatsächliche IP weiter
  + Anonym: leiten keine IP weiter, Proxy kann aber als solcher identifiziert werden
  + Elite: Empfänger erkennt nicht, dass Verbindung über Proxy geht

## Cookies:

* Sind kleine Textdateien die beim Client hinterlegt werden und dienen dem Speichern von Informationen über den Nutzer (Begrüsung, Letzter Seitenbesuch, Sprache, etc.)

## Peer2Peer:

* Kommunikation zwischen Gleichgestellten ohne Server

## NVT (Network Virtual Terminal)

* Virtuelle Konsole die es erlaubt eine TCP Verbindung zwischen zwei Stationen aufzubauen, über welche man standard 7Bit ASCII characters übermitteln kann (z.B PuTTY)

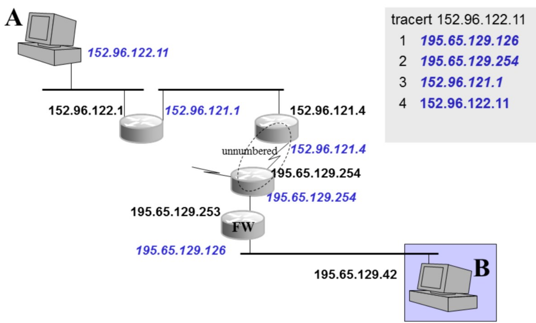
## Telnet

* Command Mode
  + Verbindungsaufbau
  + Optionen mit Konfigurationen
  + Beenden des Programms
* Session Mode
  + Übertragen von Nutzdaten (ASCII)

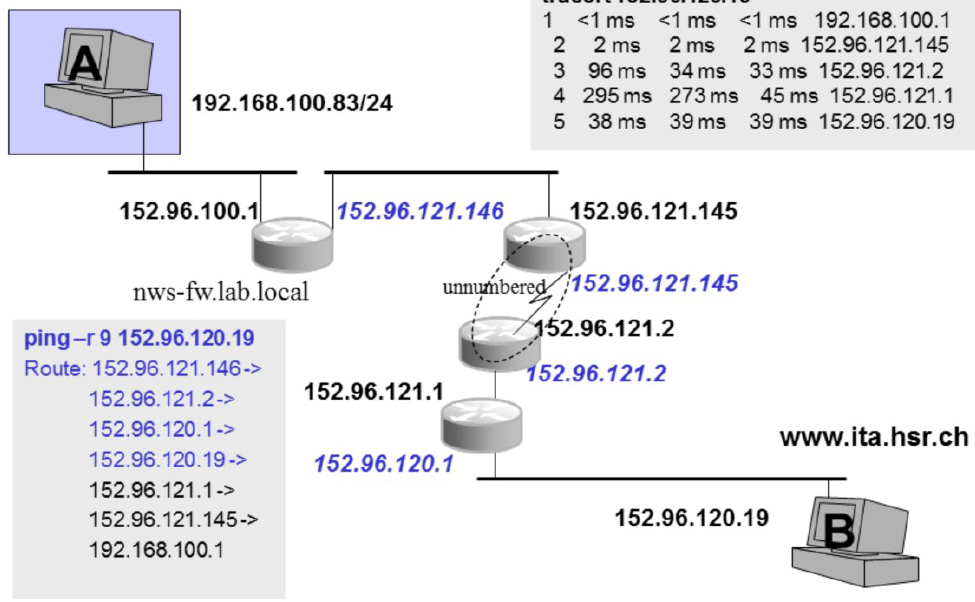
## Port Knocking:

* Portknocking ist ein Verfahren, einzelne Serverdienste abzusichern. Die Kommunikation auf dem gewünschten Port wird dabei zunächst von einer Firewall vollständig blockiert. Um diesen Port zu öffnen, sendet man mehrere SYN-Pakete mit zuvor vereinbartem Inhalt und in der richtigen zeitlichen Abfolge an den Server (der sogenannte „Knock“).

# Tracert:

* Sendende Station sendet ICMP Echo Request und setzt TTL=1
* Empfangender Server/Router zählt TTL runter und Antwortet mit ICMP Time Exceeded Paket mit dem Inhalt der IP Adresse und TTL min=Hops zurück oder besser = 255
* Station wiederholt Request mit TTL =2 usw. Bis gesuchte Station antwortet
* 4RTT = Es werden pro Station immer 3 Requests für die Antwortzeiten gesendet (Dies dient dazu, dass allfällige andere Wege genommen wurden) + 1 Reverse DNS Lookup
* 

# Ping -r [0-9]:

* Gibt die Gegenseite des Tracert aus
* 

## Route:

* Route print

# Wireshark: TCP Trace Graph

Wireshark liest die Daten zwischen der Ethernetkarte und dem Betriebsystem mit pcap

|  |  |
| --- | --- |
| http://packetbomb.com/wp-content/uploads/2014/06/tcptrace1.png | http://packetbomb.com/wp-content/uploads/2014/06/tcptrace2.png |

Zeit zwischen den Segmenten (horizontal) = RTT

Server: ACK ziemlich direkt nach Segment (bei Upload)

Client: AKC dauert länger bis es angezeigt wird (bei Upload)

Speed: TCP Segment/Zeit

4 Byte CRC ist im Wireshark nur bei einkommenden Paketen sichtbar

Bytes on the Wire = Layer 2 ohne CRC rsp. 1500B + MAC DA, MAC SRC, 2B Type/Length

Checksum Errors bei Wireshark: Moderne Ethernet-Karten unterstützen Checksum-Offloading. Da die Netzwerk-Layer Software bei Aktivierung von Checksum-Offloading keine Checksumme berechnet, enthält ein Paket in diesem Feld auf der sendenden Station noch nicht den richtigen Wert, wenn es (auf dieser Station) von Wireshark verarbeitet wird

Wireshark in einem Netzwerk detektieren: Wireshark spezifische DNS Abfragen