

# Kommunikationsnetze

## Klassification

- Local Area Network (Firma, Haushalt) wenige km
- Wide global Internet
- Global, Metropolen
- Field, Personal

## Leistungs-

- + Garantien
- Anslastung
- V.aufbau

Vstationen

Durchschalten /

Kapazität Reservieren

## Bursts

+ diskontinuierliche Übertragung

e.g. Befehl

↔ Paket-vermittlung, teilen Strecke

Steuerinformationen mit Pakete

## Verbindungs orientiert

e.g. ATM-Netzwerke & Telekommunikationsfirmen

↔ PoS

e.g. Internet

Pakete volle Addr.  
unabhängig  
unangekündigt

+ einfache Vermittlung

- Best Effort + unverherstellbar

wann  
wie schnell  
ob überhaft

⇒ Over provisioning to prevent Überlastung

Research Project...

ARPANET - 1960, erstes paketvermitteltes

⇒ NSFNET ⇒ fast kommerzielle ISP:

+ kommerzielle Nutzung

Erfolg durch Flexibilität

mehr unterschiedliche Techniken ⇒ Inter net  
TCP ⇒ Aufteilung, TCP/IP

erst:

E-Mail  
News  
TELNET  
FTP

Forschung, Industrie  
Regierung, Militär

World Wide Web - 1990 Eine Anwendung Internet

Hosts - Zugangsnetze - Inneres  
Endgeräte - Übertragungsbereich

DSL, Mobilfunk

ISP...

Protokolle "Sprache" zwischen Computern  
Was Wie Wann...

Schicht n Dienst für n+1  
konkreter Fall Protokollstapel

## ISP Hierarchie

Tier 1 - alle direkt untereinander

Keine Bezahlbox

2 - zahlen Tier 1 transit, Peering mit Tier 2

3 - zahlen Tier 1/2

# Übertragungsmedium

Netzwerkzugang - direkt Rechner-Rechner

1 Bitübertragungsschicht

Telekomm.-netz

Punkt zu Punkt

LAN e.g. Ethernet, WiFi

logische <sup>Control</sup>LLC  
Media MAC

2 Sicherungsschicht

frame (header, nutzdaten, trailer)

3 Netzwerkschicht

enge Türe IP, Rest Ende zu Ende Rechner Router

Sanduhr Anzahl

4 Transportschicht Segmente Seq. Num.

Anwendungsprogramme Ports

(erlt. Zuverlässigkeit Überlastlk.)

Anwendung

TCP/IP  
alles  
Anwendung

5 Sitzungsschicht

6 Darstellungsschicht

7 Anwendungsschicht

Standardisierung  
80er

Referenzmodell OSI-Modell der ISO

Open Systems Interconnection

International Standards Organization  
Dienst, Interface,  
Protokoll

in Praxis TCP/IP

De-facto Standards durch Marktdominanz

Gremien Tele-ITU

ISO, IEEE, ...

Internet IETF - Internet Engineering Task Force Research

RFC - Requests for Comments

1. Proposed Standard voll. Beschreibung, Interesse

2. Draft Standard Impl, tests

3. 9 RFC zu Standard erklärt

Anwendung Netzwerkanwendung

Programm Dienst erbringung Nutzung

Architektur

Client-Server  $\leftrightarrow$  Peer-to-Peer

- + keine Server(kosten)
- + große Daten
- komplexe Protokolle

Wahl Transportschicht

Zurverlässigkeit

Datenratensteuerung (Min. bei Multimedia)

Echtzeitfähigkeit

Traffic

|           |     |
|-----------|-----|
| Streaming | 68% |
| File      | 13% |
| Browsing  | 3%  |

E-Mail

1. schreiben
2. Nutzer  $\rightarrow$  Mailserver
3. Avenue
4. Zieladresse  $\rightarrow$  Mailserver  $\rightarrow$  IP Adress

Mailserver - Mail Programm

POP3 - nur Löschen der Mailabholung Post Office P.

IMAP - mehr Funktionen Internet Mail Access P.

TCP Port 25 SMTP Protocol - Simple Mail Transfer Protocol

- ↳ in Postfach
- b. Abruf

Server grüßt  $\rightarrow$  220 Statuscode optionaler Fließtext) - Effizienz  $\leftrightarrow$  Verständlichkeit  
HELO servername eigene.domain / EHLO - extended HELO bei Fehler Fallback  
Parameter

250 ok

MAIL FROM: <Absender@mail>

250

RCPT TO: <Empfänger@mail>

250

DATA

354

~~~~~ - 7-Bit-ASCII.

\* - on a line by itself ends mail \* verständlich  
250 \* - escape at beginning of new line ... für,

QUIT

221

) Envelope der Nachricht

Kopfzeile Pattern key: value  
empfangen von Mailserver - Received: from... by... time

Pfeile: ( From: Name <mail>  
To: mail  
Subject:  
Leerzeile  
7-Bit-ASCII Daten

Postel's Law / Robustness Principle

do conservative  
accept liberal

$\Rightarrow$  ignore and forward  
unknown Headers

+ Header die Sonderzeichen ermöglichen

+ MIME - Multipurpose Internet Mail Extension

MIME-Version:

Content-Transfer-Encoding: e.g. base64

Content-Type: e.g. image/jpeg

erlaubt mehrere Teile, verschachtelte MIME-mails, ...

# WWW

(verschachtelten) entl. dynamisch erzeugt (e.g. common gateway interface)  
Webseite aus Objekten z.B. HTML - Hypertext Markup Language mit eingebetteten Bildern, Links

Browser = HTTP-Client - Herunterladen, Anzeigen

## HTTP

Objekttyp egal

URL - Uniform Resource Locator  $http://\underline{\text{www. ... .de}}/\underline{\text{...}}$

Servername Pfad

$\hookrightarrow \text{IP} \Rightarrow \text{TCP Port 80}$

Anfrage - 1 Objekt

Methode Pfad Version  
GET e.g. HTTP/2.1

POST

HEAD, PUT, DELETE

Header  $\begin{cases} \text{Host: servername} & -\text{Pflicht seit HTTP/1.1} \\ \text{User-agent: Browser, OS...} & \text{Verbindungsauftbau 1RTT} \\ \text{Connection: keep-alive} & -\text{persistentes HTTP} \quad n \cdot 2\text{RTT} \\ \text{Leerzeile - Ende Anfrage} & \Rightarrow 1\text{RTT} + n \text{ RTT} \end{cases}$

## Antwort

200 ok  
400 bad request  
404 not found  
505 version

Statuszeile - Version Statuscode Beschreibung

Content-type: text/html; charset=UTF-8

Date: beantwortet

Connection: keep-alive

Leerzeile Content-Length: #Bytes Daten - ~~bei persistenter Verbindung~~ erforderlich,  
da Ende nicht durch  
Verbindung schließen

Objekt

## Latenzen

Deutschland ~5-20ms

Europa ~50ms

$\rightarrow$  Nordamerika ~100-150ms

Weltweit ~500ms

! Lichtgeschwindigkeit

Pipelining  $\Rightarrow 1\text{RTT} + \text{tiefe RTT}$  ! Head-of-Line Blocking

Verzögerung durch noch nicht ausgeliefertes Objekt

$\Rightarrow$  parallele TCP Verbindungen

- teilen Netzwerkkapazität

- overhead

empfehlung max. 2

Daten hochladen

HTTP Zustandslos aber entl. dynamische Gen.

GET - in Pfad kodieren ? key=val & ...

$\Rightarrow$  Cookies bei Client z.B. Einkaufswagen

POST

in Antwort  $\rightarrow$  Set-Cookie: text

header Content-Length:

Request  $\rightarrow$  Cookie: text

$\leftarrow$  -Type

Daten

CDN - Content Distribution Networks

mehrere inhaltsgleiche Server

Nutzer zum besten Weiterleiten

RTT  
Last  
Kosten

Caching

+ weniger Daten

- gleiche RTTs

Antwort  $\rightarrow$  Last-Modified:

Request  $\rightarrow$  If-modified-since:

$\hookrightarrow$  falls nicht Antwort 304

HTTP/2 nicht mehr human readable header komprimiert

Anfrage/Antwort-Streams über 1 TCP-Verbindung + weniger Header Line Blocking (noch auf TCP Ebene)

in frames

Header 31 bit StreamID  
Länge

+ Stream Priorisierung

+ Server kann Stream öffnen

→ Push Betrieb, wahrscheinlich benötigte Objekt

+ beide durch GOAWAY Verb beenden

HTTP/3 nutzt QUIC auf UDP

mischt TCP TLS

## Kryptographie

Verschl

sym. AES K(m)

asym. RSA K<sup>+</sup>, K<sup>-</sup>

Krypt(HL))

Sig

Hush K(m) Kollision

Austausch Diffie-Hellman

$A = g^a \text{ mod } p$

$K = B^a \text{ mod } p$  ! Mit M

Darstellung X.509

Subject  
Issuer  
Validity  
von-bis

⇒ Zertifikat eines Dritten

$K_{id}^+, K_{third}(Id, K_{id}^+)$

? Vertrauen

Ehrlichkeit  
Id. Prüfung  
auf K<sup>+</sup> aufpassen

Regeln - Wer Zert für Wen?

Web of Trust

PKI - Public Key Infrastructure

hierarchisch

Root-Zertifikate selbstsigniert in Programm / OS  
kein neues Vertrauen

Kette zu einer Root länger → Vertrauen schwächer

CRL - Certificate Revocation List

SSL/TLS - Secure Socket Layer / Transport Layer Security  
auf TCP nach oben wie TCP

HTTPS - HTTP mit TLS

DTLS für UDP

BRUNNEN

## DNS - Domain Name System

hierarchische verteilte Datenbank + Abfrageprotokoll  
| skalierbar  
| Redundanz

Anwendungsschicht  
UDP & TCP Praxis UDP, da Nachrichten fast immer in ein Packet  
kein Vb. aufbau

13 offizielle Rootserver (ICANN), landesweit NIC-Network Information Center

## FQDN - Fully Qualified Domain Name

www. .... .de  
Host name              Top Level Domain  
                          Domain

Root-Server  
TLD-Server  
Autoritative-Server für Domäne  
lokale Server + Caching

## RR - Resource Records

Typ z.B. A Hostname → IP  
Name CNAME Alternative  
Wert NS Nameserver  
MX SMTPServer  
TTL - solange Cache bar

## DNS-Nachricht

ID Flags  
# Questions # Answers RR  
# Authority RR # Additional RR

iterative ↗ rekursive Abfragen

## Reverse DNS

### PTR Resource Records

Einbettung in DNS-Namenraum

... .in-addr.arpa  
Bytes in umgekehrter Reihenfolge      ... ip6.arpa  
Habt bytes umgekehrt

## Weitere Funktionen

Hostname → IP-Addr + Lastverteilung

DNSsec

Zeichensätze

...

# Transportschicht

Ende-zu-Ende Protokolle auf unzuverlässigen Programmtransport Host zu Host  
für Netzwerke irrelevant

Socket-API weit verbreitet

# Transportschicht Multiplexing

Portnummern 16-bit

~ Anwendungen well-known Ports (IANA)  
(1024 privilegiert)

TCP  
80 HTTP  
53 DNS  
23 SMTP

## Protokollnummern

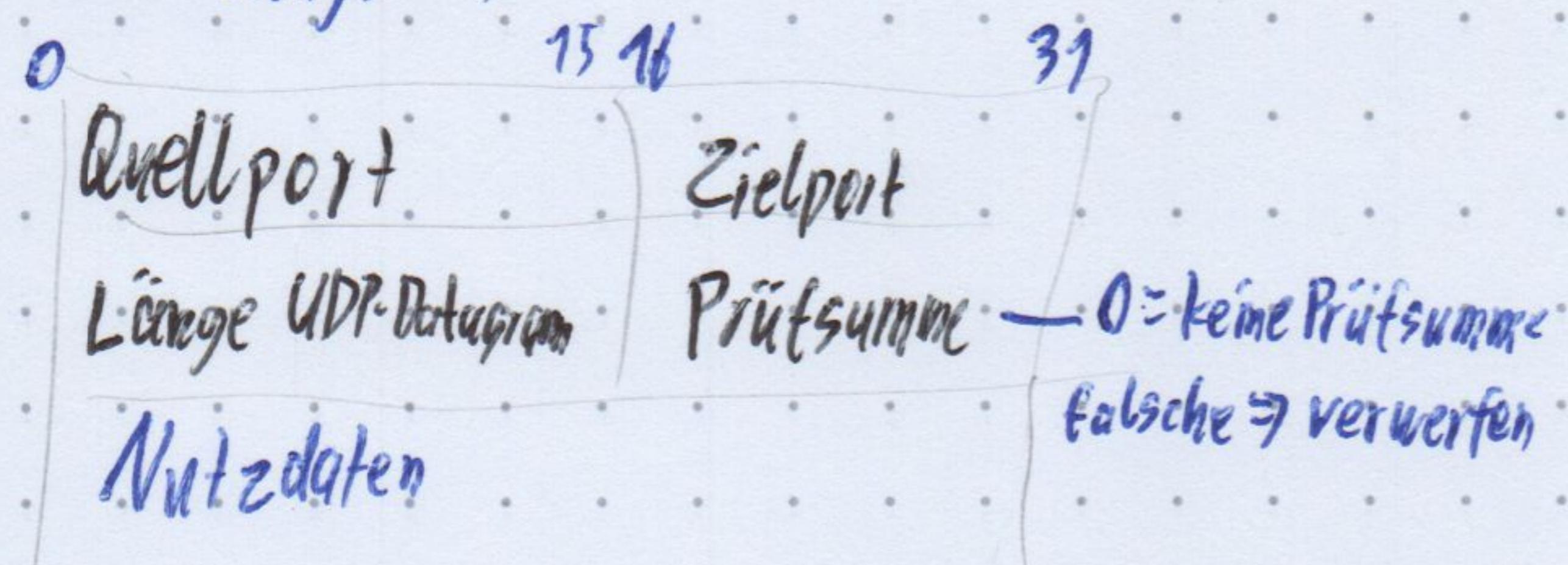
# Netzwerkschicht weibes Trns. P.

b TCP

27 UDP

# UDP - User Datagram Protocol

# verbindungslos



du hast kaum mehrere haken

Socket ~ Quell-IP, -Port

- + speed  $\Rightarrow$  Multimedia, DNS
- + 1 socket <sup>für</sup> mehrere

TCP - Transmission Control Protocol

Zuverlässiger reihenfolgehaltender Bitstrom Voll duplex  
Segmentierung Reassemblierung Unicast

Socket ~ Quell-IP, -Port  
Ziel-IP, -Port

- Ack,NAck : fehlerhaftes Paket  
⇒ z.B. angekommenes nachmal geschickt
- Seq. Num., ACKERNEUTRALISATION  
fehlerhafter Empfang ⇒ wiederhole altes Ack
- ↓ : Paketverlust

! Wiederholung Segnum.  
→ zufällig anfangen mit Nam aus alter Verbindung  
großer Bereich + Security (Pakete einschleusen)

## Stop-and-Wait

↓ Pipelining  $\Rightarrow$  Pakete speichern bis acknowledged

fehlendes Packet 7 tick

## Go-back-n

Empfänger @Empfang: kummulatives Ack für letztes  
in richtiger Reihenfolge

optionales Puffern Pakete falscher Reihenfolge

# Selectives Repeat

Ack für jedes Packet einzeln

**BRUNNEN**

- aufwändiger  
+ (viele effiziente)

## TCP

erstes im Segment enthaltenes Byte

32-bit Seq. Num pro Richtung über Bytes

Kumulative piggybacked Ack's (Flag + Seq. Num) auch wiederholte  
Teile Nutzdaten = 0      erstes fehlendes Byte

~~Erlangspuffer optional~~

Puffern zukünftige Pakete optional

## Wahl RTT

- RTT + 4 · Standardabweichung
- L<sup>er</sup>gleitender Mittelwert aus Ack's
- nächstes Timeout doppel bis z.B. max 60s

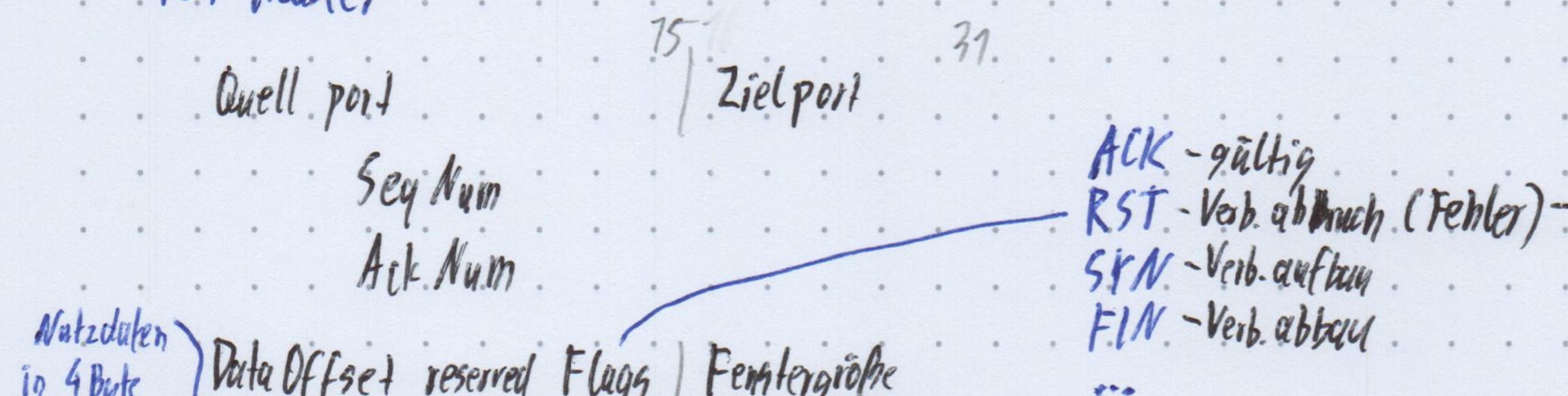
## DACKs - Delayed Ack's

nächstes erwartetes Packet  $\Rightarrow$  Ack bis nächstes Segment / verzögern  
max z.B. 500ms

Fast Retransmit bei triple Duplicate Ack

1 Segments

## TCP Header



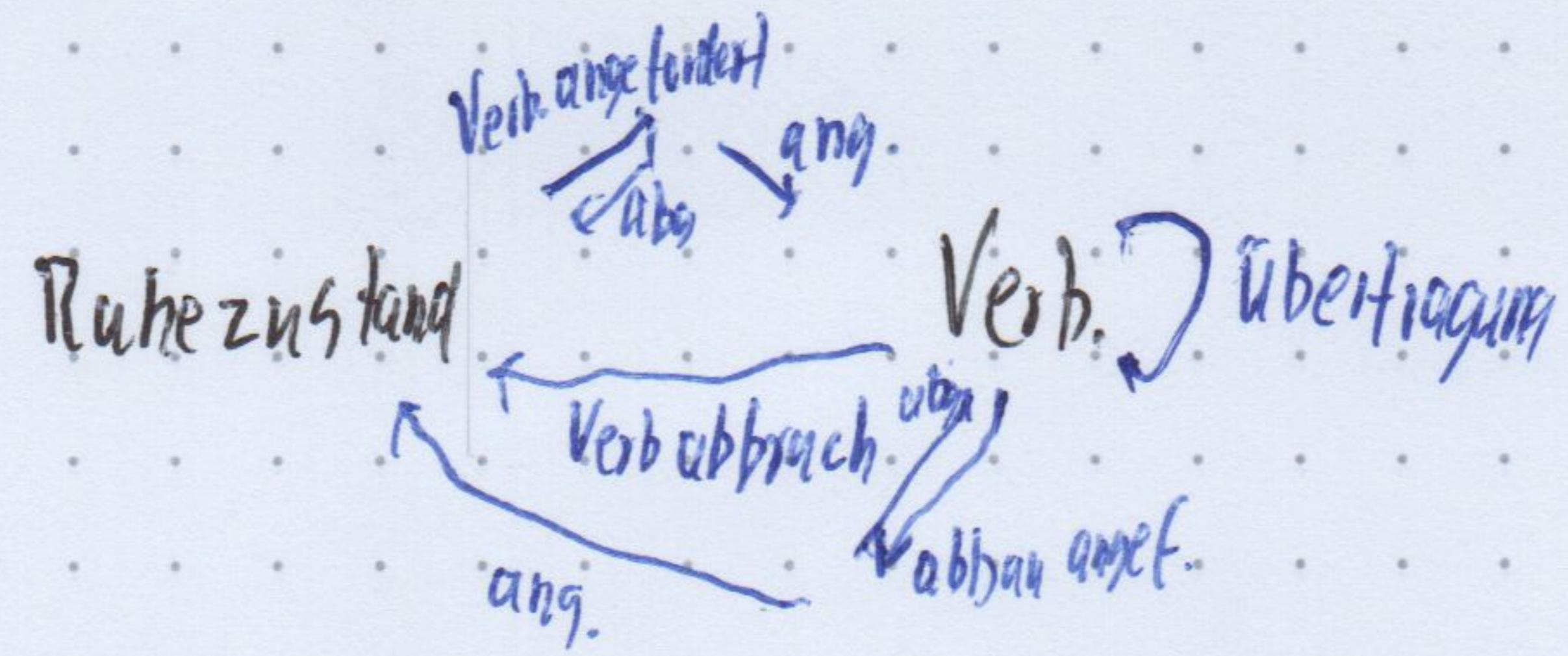
- ACK - gültig
- RST - Verb. abbruch (Fehler) -
- SYN - Verb. aufbau
- FIN - Verb. abbau
- ...

Daten      erster Byte Optiontyp  
Typ ≥ 2 zweiter Byte Gesamtlänge Option in Byte

| Typ | Länge | Wert                                                                                                                                                             |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0   | 1     | Ende Optionsliste - Padding                                                                                                                                      |
| 1   | 1     | leere Option - Lücken                                                                                                                                            |
| 2   | 4     | MSS - für erstes Segment max Bytes, empfangenes Segment                                                                                                          |
| 3   | 3     | Window Scaling      sonst definiert nur 536 Bytes                                                                                                                |
| 4   | 2     | SACK Unterstützung                                                                                                                                               |
| 8   | 10    | Timestamp      Anwendung, wenn bei Client und Server<br>4 Byte aktueller Timestamp      in erstem Segment<br>4 Byte empf. Timestamp<br>$\Rightarrow$ RTT Messung |

## Dienstprimitive für API

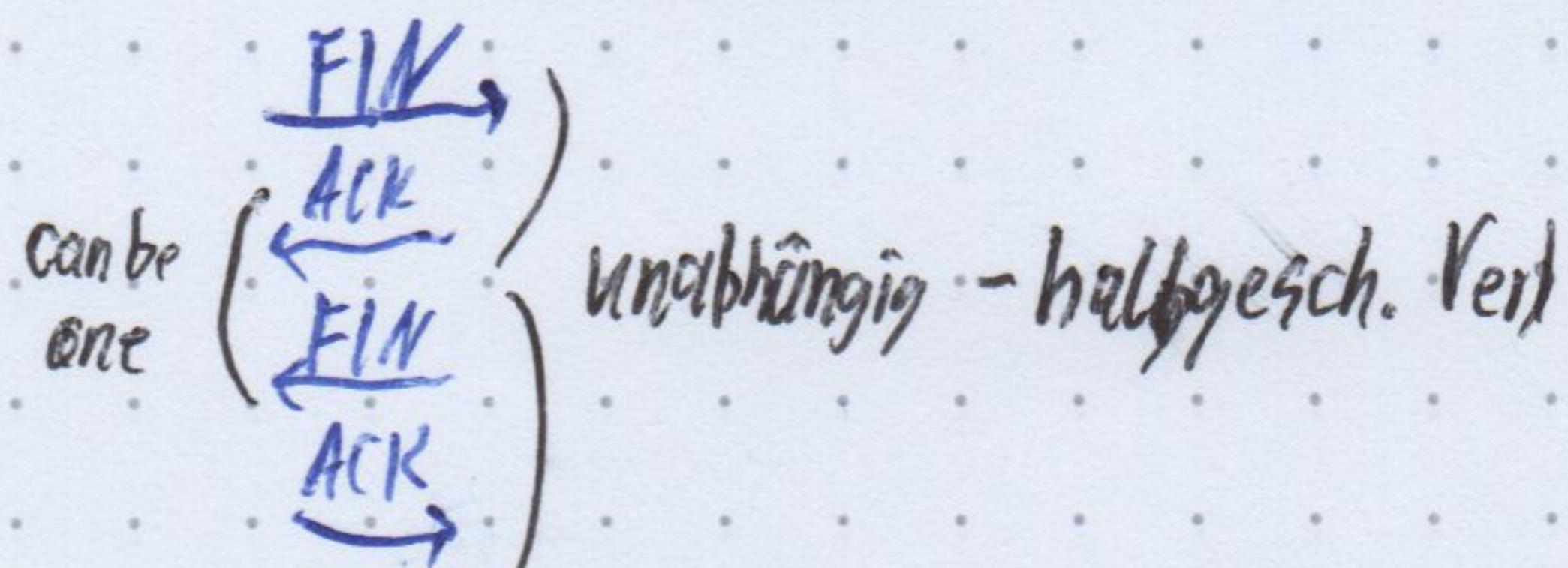
- ? B. LISTEN, CONNECT
- RECEIVE, SEND
- DISCONNECT



## Verbindungsaufbau 3-Wege-Handshake

- SYN, seq=x
- SYN-ACK, seq=y ack=x+1 - halboffene Verb.
- ACK ack=y+1 - bestätigt Leben Client  
z.B. keine alte Nachricht  
(keine Quelladdr fälschen)

## Verbindungsabbau 4-Wege-H.



Falls Verlust schließen nach z.B. 4 Min  
! für Server viel

## Segmentierung

Nagle Algorithmus + vermeidet Tinygram Problem

unbestätigtes Segment unterwegs  $\Rightarrow$  Verzögerung Absenden  
nur Daten < MSS - Max. Segment Size

Reset keine Garantie auch kein Reset (Reset Wert)  
 $\Rightarrow$  nichts mehr senden Zweck Fehlerbehandlung  
empfangen

## Flusskontrolle - wenn Empf. überfordert

Halt/Weiter-Nachrichten ! Daten bis Halt  
hoffentlich ankommen

## Kreditbasierte Flusskontrolle

### Sende Kredit mit jedem ACK

Receive Window L # Bytes nach bestätigten Daten noch in Pufferbereich  
in TCP

! Probe nach Persist-Timer bei voller Empfangspuffer  
~~da nicht abgedrängt~~, da kein ACK mit neuem RWIN mehr  
Probe-Packet 1 Byte groß

Sendepuffer unbestätigt + noch nicht versandt +

Empfangspuffer noch nicht abgeholt + noch nicht bestätigt +  
für Sender unbekannt

(Schlechte Anwendung, aber oft)

Silly Window Fenster immer nur wenige Bytes größer  
Syndrom  
 $\Rightarrow$  Empfänger vergibt nur sinnvoll viel Kredit  
(oder Nagle)

## Maximale Fenstergröße für hoffentlich keine S.Num. Wiederholung

halber Seq Num Bereich ohne Übertragungs Wiederholungen

$\Rightarrow$  besser  $\frac{1}{4} \cdot S$

in TCP durch 16 bit RWIN beschränkt

- beschränkt Pipelining bei großer RTT

$\Rightarrow$  Window Scaling Option in STNs bei Verbaufbau

Pro Verbindung  $0 \leq i \leq 14$  mit  $j=16$ .  $RWIN = S_{\frac{1}{4}}$

$RWIN = RWIN \cdot 2^i$  bessere Granularität

$$\text{Rate} = \frac{\text{Window}}{\text{RTT}}$$

$$\Leftrightarrow \text{Win} = \frac{R \cdot \text{RTT}}{2}$$

BDP - Bandwidth-Delay Product

$$BDP = R \cdot RTT$$

## Überlastkontrolle im Netzwerk

zu max. Datenrate steigt durch Wartezeit in Puffer exponentiell

Drop-Tail-Puffer Empfang = L  
Senden = R Verlust  $p = \frac{L-R}{L} = 1 - \frac{R}{L}$  bei zufälligen, unabhängigen Ankunftszeiten  
Leer, wenn Puffer voll für jede Verbindung über Puffer gleich  
Verbindung mit höchster In-Rate profitiert

Congestion Collapse ? Neuantragungen  $\Rightarrow$  exp. RTO

Signal durch Netzwerkinnere

z.B. Bit in Packet, explizite Senderaten Vergabe

Ende-zu-Ende - TCP nicht in Netz, da komplex  
Netz kennt keine Verb.

TCP Reno

Multimedia

- Sägezahn muster

- schwankende Latenz

- keine Zuverlässigkeit nötig

Annahme: Segmentverlust = Überlast

Senderate durch Überlastfenster (Congestion Win) neben RWin

Congestion Increase  $\left\{ \begin{array}{l} \text{AIMD - Additive Increase } \approx 1 \text{ MSS / RTT} \\ \text{Multiplicative Decrease } /2 \end{array} \right.$  bei jedem Ack  $c_{win} = c_{win} + \frac{\text{MSS}}{c_{win}} : \text{MSS}$

$c_{min} > ssthresh$   $\uparrow$   $\downarrow$   $\rightarrow$  self clocking - keine Timer

$c_{win} \cdot 2$  pro RTT  $\Rightarrow$  bei Ack  $c_{win} = c_{win} + 1$  bei TDACK

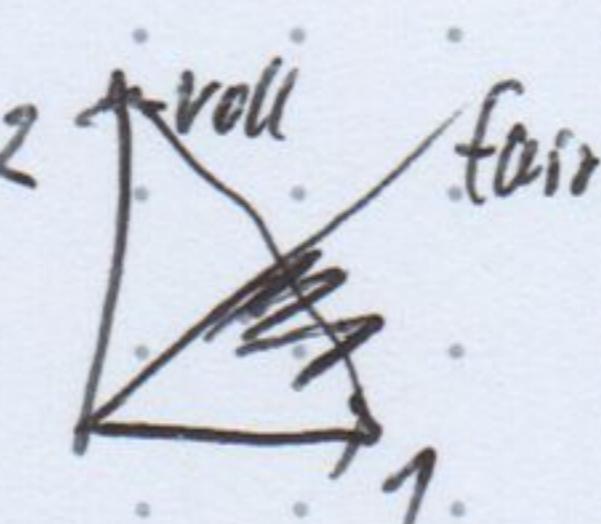
$\downarrow$  moderate Überlast

Sägezahn muster

Fairness im längeren zeitlichen Mittel  
nur TDACKs

in Relation zu RTT gleichmäßig

höhere Rate  $\Rightarrow$  mehr Verlust  
größere cwin Reduktion



TCP macht UPP ohne Kontrolle Platz

paralleler TCP Verb.: Anwendung mit mehr profitiert

TCP

Multimedia

- Sägezahnmuster
- schwankende Latenz
- keine Zuverlässigkeit nötig

drahtlose Netze

Packetverlust + Überlast  
häufig besser wiederholt

Viele Varianten hier TCP Reno

- Ziele:
- kompatibel zu anderen Varianten
- fair

Tahoe kein TDACK

New Reno Seg nach TDACK

Vegas Früherkennung durch RTT

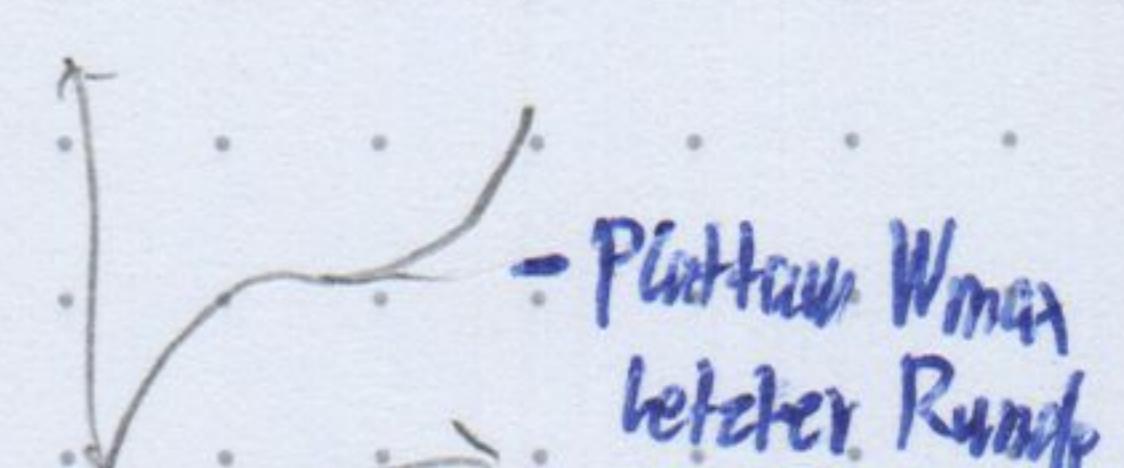
BBR + Probing

Long Fat Pipes (LFP)

- hohe BDP  $\Rightarrow$  geringe Rate
- $\Rightarrow$  lange Periode Sägezahnmuster
- dazu kommen gelegentliche Verluste
- echtes max. RTT nicht erreicht

CUBIC (Linux standard)

fenster(t) = kubische Funktion



+ Long Fat Pipes

+ fair bei unter. RTT da nicht Self Clocking

+ fair zu Reno

## Netzwerkschicht

Routing Algorithmus  $\Rightarrow$  Protokoll  $\Rightarrow$   
JE

Routing-Tabelle  
↓

Paket  $\rightarrow$  Forwarding  $\rightarrow$  Paket

Forwarding port := Leitung

Ziel: Line Speed (z.B. 10Gb/s, 256B Pakete  $\Rightarrow$  4Mio Pakete/s)

$\Rightarrow$  Spezialhardware

CAMS - Content Addressable Memory

Switching Fabric

gemeinsamer Speicher

Bus

Crossbar Switch

Port

Leitungsabschluss

finden Ziel

Warteschlangen

IP - Internet Protocol

ICMP - Internet Control Message Protocol

Netzwerkadapter  $\sim$  weltweit eindeutige 32bit IP Addr. a.b.c.d

Subnetze (Schicht 2, daher Kommunik. ohne Router)

Klassen

A 8bit Netz, B 16, C 24 nach erstem Byte

zuunflexible  $\Rightarrow$  CIDR - Classless + Inter-Domain-Routing

Größe über

Netzmaske

/<sub>m</sub> CIDR-Schreibweise

Netzwerkaddr vhost=0

Broadcast vhost=1

$\Rightarrow 2^m - 2$  Host Addr.

Addr. bereiche weiter unterteilbar

Routing Tabellen Eintrag (Metrik  $\downarrow$ , falls gleiche Prefix Länge)

Addr.;  $\downarrow$  Addr. Maske

Ausgangsport

Gateway (bei nicht lokalen)

Default Route

0.0.0.0/0

Longest Prefix Matching

erläutert - Routenaggregation

reservierte Bereiche

10.0.0.0/8 private Netze

127.0.0.0/8 Loopback (Host interne)

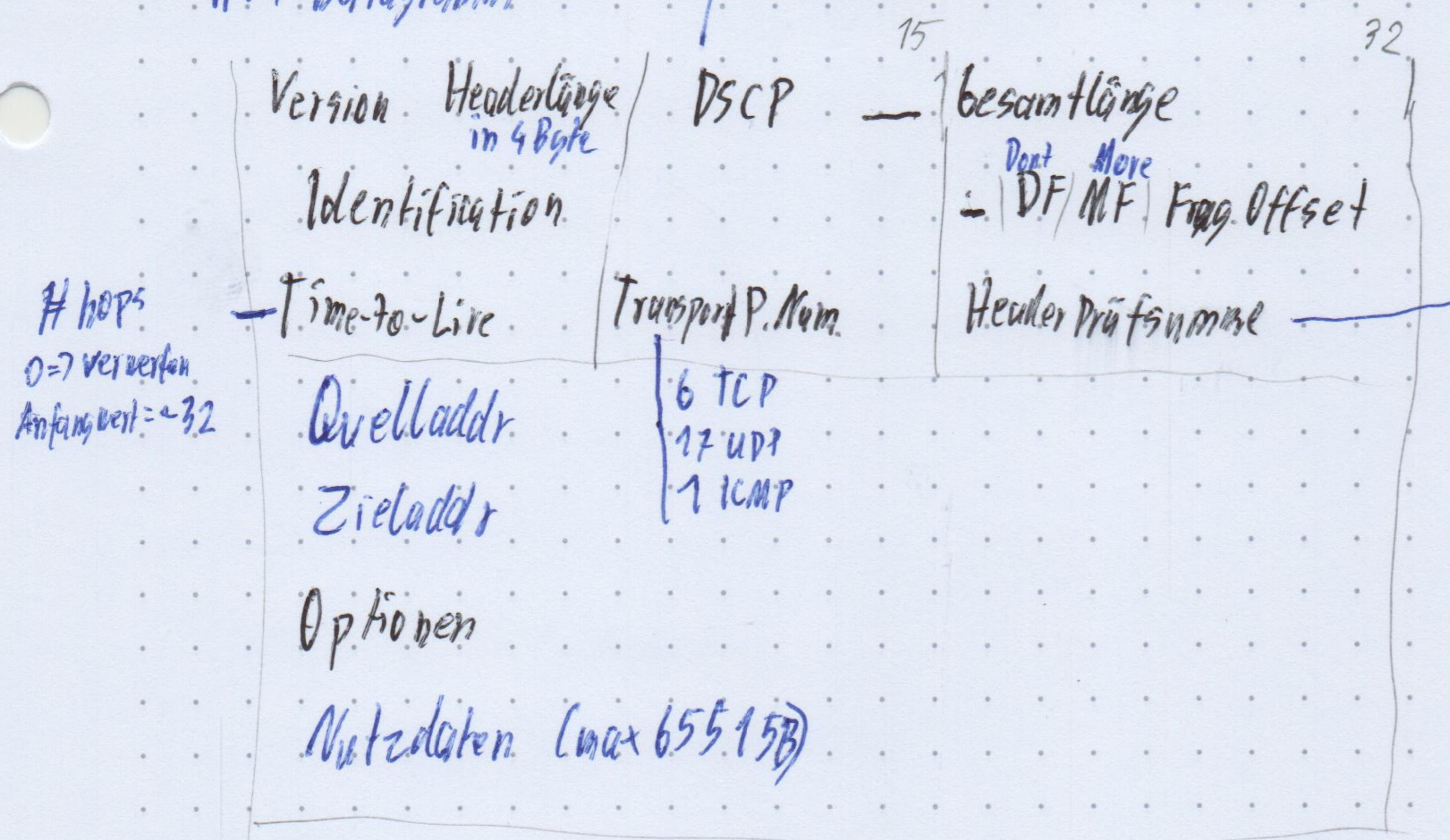
172.16.0.0/12 private Netze

192.168.0.0/16 private Netze

...

## IP v4 Datagramm

## Differential Service Code Point



ECN Flags für Überlast  
Signalisierung

MTU - Maximum Transmission Unit der Sicherungsschicht uneindelig

⇒ Fragmentierung durch Router

Reussemblieren bei Empfänger  
(wenn Vankommen)

Id = Packetnummer = Zugelangigkeit Fragment (neben Addr., Prot. Num.)

DF - lieber verwerfen, ICMP Dest. Unreachable an Sender

MF - nicht letztes Frag

Offset (13bit) - relative Lage erstes Byte in 8 Byte

Pfad-MTU-Ermittlung

gelegentliches Probeing

- Packet Aufteilung eigentlich in höherer Schicht

- Firewalls die ICMP-Fehler filtern

ICMP <sup>Impl.</sup> verpflichtend

z.B. ping (Echo Req., Resp.)

in IP-Pakete gekapselt

| Type  | Code                | Checksum |
|-------|---------------------|----------|
| Daten | je nach Type / Code |          |

z.B. Anfang betreffendes IP Paket

z.B. type

0 Echo Reply 8 Recp

3 Dest. Unreachable

Code

0 Netz nicht erreichbar

1 Host

2 T.Protokoll

3 Port

4 Frag

Time Exceeded

Param Problem

Router Solicitation  
Advertisement

traceroute

Listet Routen durch UDP Pakete mit zunehmender TTL

## IPr4 Address Knappheit

~~Netzwerk~~ (große Teile ungenutzt, unfair verteilt)

⇒ NAT - Network Address Translation

private Netze bei Verlassen

NAT Router Zuordnung

NAT Tabelle

Protokoll Lokal (Port, IP) ent (Port IP) Ziel (IP, Port)

+ Workaround

- IP nicht eindeutig (+ Privacy)

- Router stati

- Schichten (noch schlimmer bei FTP, SIP)

Port & IP in Anwendungsprotokoll

IPr6 1998 + Addr Knappheit

ICMPv6 + Vereinfachung, Router (Frag, IP Optionen, Prüfsumme)

min Linklokal 128 Bit Addr Schreibweise 2 Byte Hex Blöcke : Aggre  
 global n / führende 0 weglassen  
 Schnittstelle : ; n O Blöcke  
 letzte 4 Byte auch IPr4 Schreibweise

Version Verkehrs Klasse Flow Label  
 Nutzdatenlänge Next Header Hop Limit  
 Quelladdr. Zieladdr.  
Daten

Aggregateable Global Unicast

16 16 16 32 16/24  
001 Registry ID | Provider ID | Subscriber ID

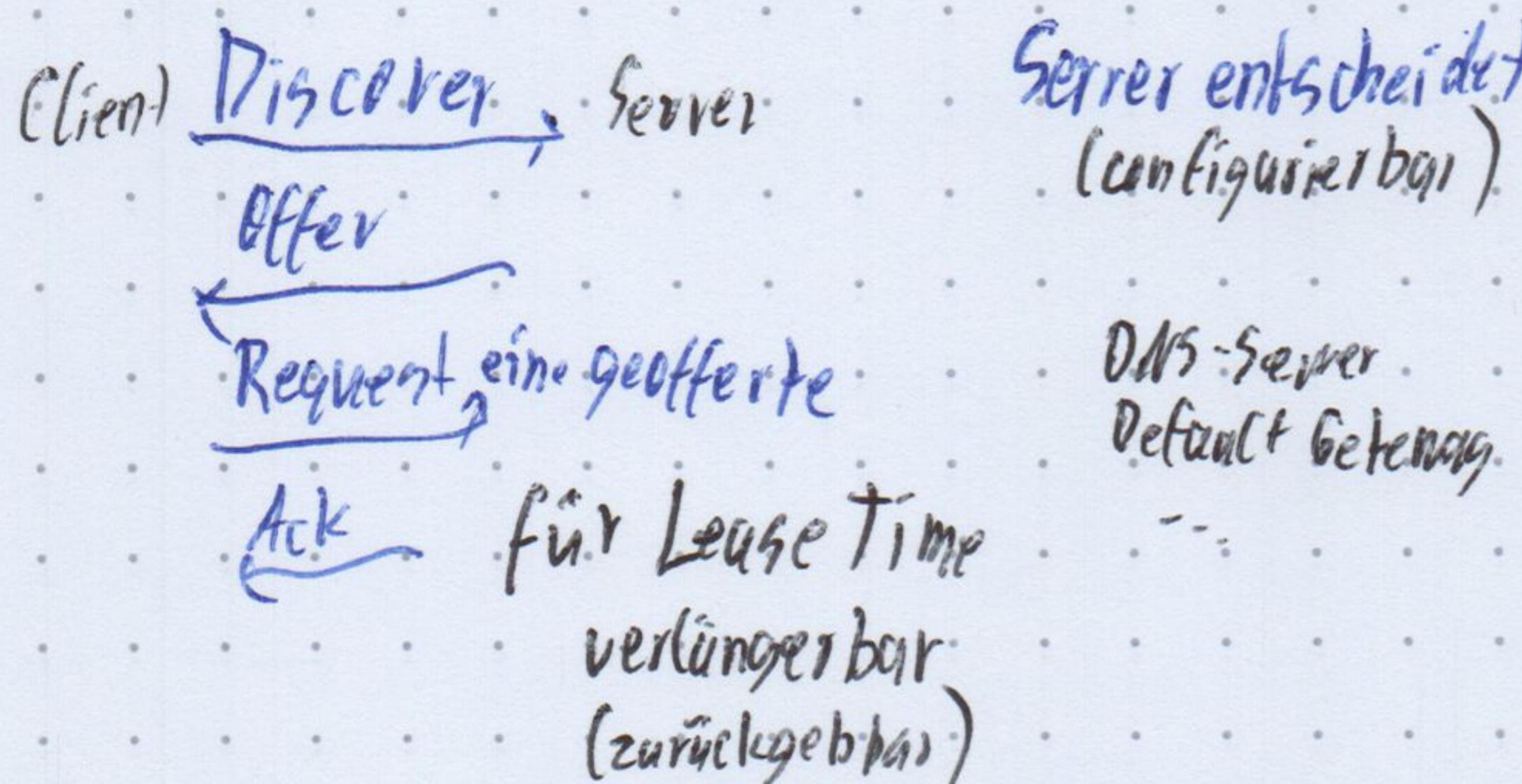
Subscriber Addr.

fe80::/10 Linklokal werden von Router  
 nicht weitergeleitet  
 ::/128 Unspezifizierte Addr  
 ::1/128 Loopback  
 0:0:0:0:0:ffff::/96 Letzte 4 Byte IPr4

Simultane Umstellung unmöglich

⇒ Tunneln

## DHCP - Dynamic Host Config Protocol



IPr6

Subnetz durch ICMP Router Solicitation

Client generiert auf MAC Basis

Überprüfung auf Eindeutigkeit

## Routing Netz=Graph

Kosten  $c(x,y)$  z.B. interne Datennetze

Pfadkosten =  $\sum$  Kantenkosten

Routingalgorithmus = günstigster Weg

statisch  $\leftrightarrow$  dynamisch  
ständig aktiv

global  $\leftrightarrow$  dezentral

Krollständige iterativer Info  
Topologie austausch

Link State Routing global z.B. OSPF - Open Shortest Path First

lokal z.B. Dijkstra kleines Netz, sonst Schleifen

Nachbarn mit HELLO-Packeten finden

periodisch / event: Link-State-Info-Packete (ID sender, Seq., Age haben Lebensdauer  
erl. nur Änderungen Liste Nachbarn, Kosten)  
fluten, duplizitate verwerfen

Distance Vector Routing

z.B. RIP - Routing Information Protokoll

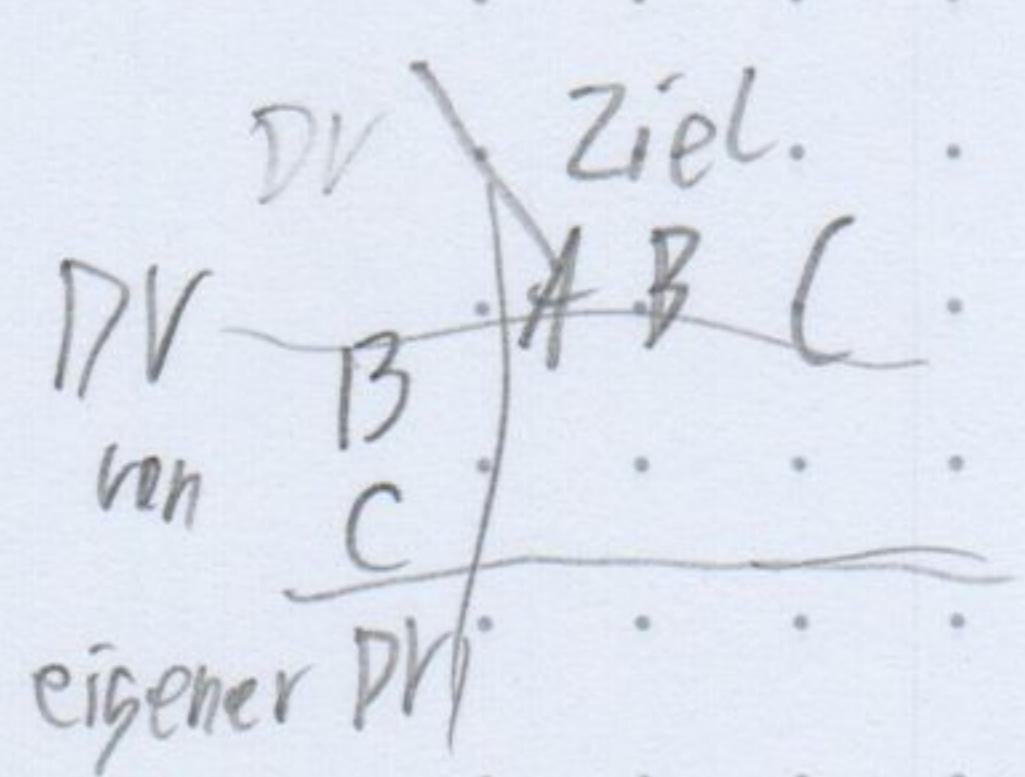
Kosten  $\leq 15$  16 =  $\infty$  30s Updates

Split Horizon Regel: kein Update

an Nachbarn, der Tabelle geändert hat

Bellman-Ford-Berechnung

$$d(x, k) = \min_{v \in \text{Nachbarn von } x} (c(x, v) + d(v, k))$$



Austausch Distanzvektoren an Nachbarn, akt. eigenen

konvergiert mit zufällig selbst 0

Nachbarn kanten  
Rest  $\infty$

+ good news travel fast

- bad " " slow

Routing-Schleife während Count.  $\rightarrow$  infinity

Internet AS-Nummern IANA

z.B. LSR, DVR

$\Rightarrow$  Poisoned Reverse - nicht immer ansprechend

Intra-AS/Domain Routing - ein Interior Gateway

Protocol

Nachbarn über die man Routet diese route als  $\infty$  angeben

Inter-AS - de-facto BGP - Border Gateway Protocol an Batennay Routern! innere müssen wissen zu welchem gateway Router

eBGP = BGP Sitzung = TCP Verb. zu Nachbar AS Router

i intern = TCP & eigenen AS Batennay Routern

Pfadvektor-Routingprotokoll

NEXT-HOP - für intra-AS-Routing

Welche Route? Admin, Länge, nächster NEXT-HOP

Ankündigung (Addt. Bereich über Batennay) Route

BRUNNEN mit Liste durchquerter AS

Hot Potato Routing

freiwillig

AS-PATH

Sicherungsschicht oft (Logical Link Control, Medium Access Control)  
multiplexing, flow control  
acks, errors

Stationen = Rechner

Arten von Links:

Punkt-zu-Punkt Verbindung (Vollduplex unabhängig & Halb "↔ oder →" Broadcast mit 2 Stationen)

Broadcast-Links gemeinsames Medium  
⇒ Medienzugriffsteuerung

Rahmenbildung (Bitübertragungsschicht in der Regel unzuverlässige fehlerbehafteter Bitstrom) Point-to-Point

Rahmengrenzen 2<sup>Byte</sup> eng

Anfang von Gruppen  
Bitübertragungsschicht?

Zeichenzählung bekannt: Bytegrenzen

Länge in Rahmenlänge

- Fehler → dauerhaft falsch

Character Stuffing

Flagzeichen für Anfang, Ende

Escapezeichen für Codetransparenz

Bitstuffing nur Bitstrom

z.B.

Flag 0 111 1110

Esc 0 an alle 5x1 dranhängen / entfernen

MAC-Addressierung ~ Netzwerkadapter local eindeutig zur Verbindung global eindeutig durch Hersteller IEEE

Ethernet 6 Byte hexhex...

Broadcast 8 bits 1

ARP Address Resolution Protocol meist IP → MAC aber sehr flexibel

Broadcast Req. IP

Unicast Reply erkennende Station

ARP Cache  
mit ~~Table~~  
HL

## Fehlerbehandlung

Kanal  
Anschluselektronik

Fehler können nur unwahrscheinlich gemacht werden

Fehler hier gekippte Bits

| Einzelfehler z.B. Rauschspitzen  
| Mehrfach-/Bündel-/Block-/Burgt - schwerer

Paritätsbits <sup>1-bit</sup>  
zu gerade  
ungerade

- nur ungerade viele Fehler

## CRC-Cyclic Redundancy Check

Addition = Subtraktion = XOR

bits als Koeffizienten eines Polynoms von Grad  $k-1$  über  $\mathbb{F}_2$   $x^2 + x^2 = 0$

Sendet  $b_0 x^{k-1} + \dots + b_{k-1} x^0$

Ergänze bits, sodass Polynom durch Generatorpolynom <sup>6 mal 9</sup> teilbar

g Den anhängen

Polynomdivision

Rest draufaddieren bzw. Anhang trennen

$\text{CRC}-1 = x + 1$  = Paritätsbit

$\text{CRC}-7 = x^3 + x^2 + 1$  SD-Karte

$\text{CRC}-16-\text{CCITT} = x^{16} + x^{12} + x^5 + 1$  Festplatte, Blocktakt

$\text{CRC}-32-\text{Ethernet} = x^{32} + \dots$

$1100110000 \div 11001 =$   
 $11001 \text{ XOR}$   
 $00001 \text{ kein XOR, wenn beginn}$   
 $00100 \text{ begin } \Rightarrow$   
 $01000$   
 $10000$   
 $11001 \text{ XOR}$   
 $01001 = \text{Rest}$

Empfänger

teste Rest = 0

alle Einzelfehler

Doppel " "  $\Leftrightarrow$  G nicht durch  $x^k + 1$  teilbar

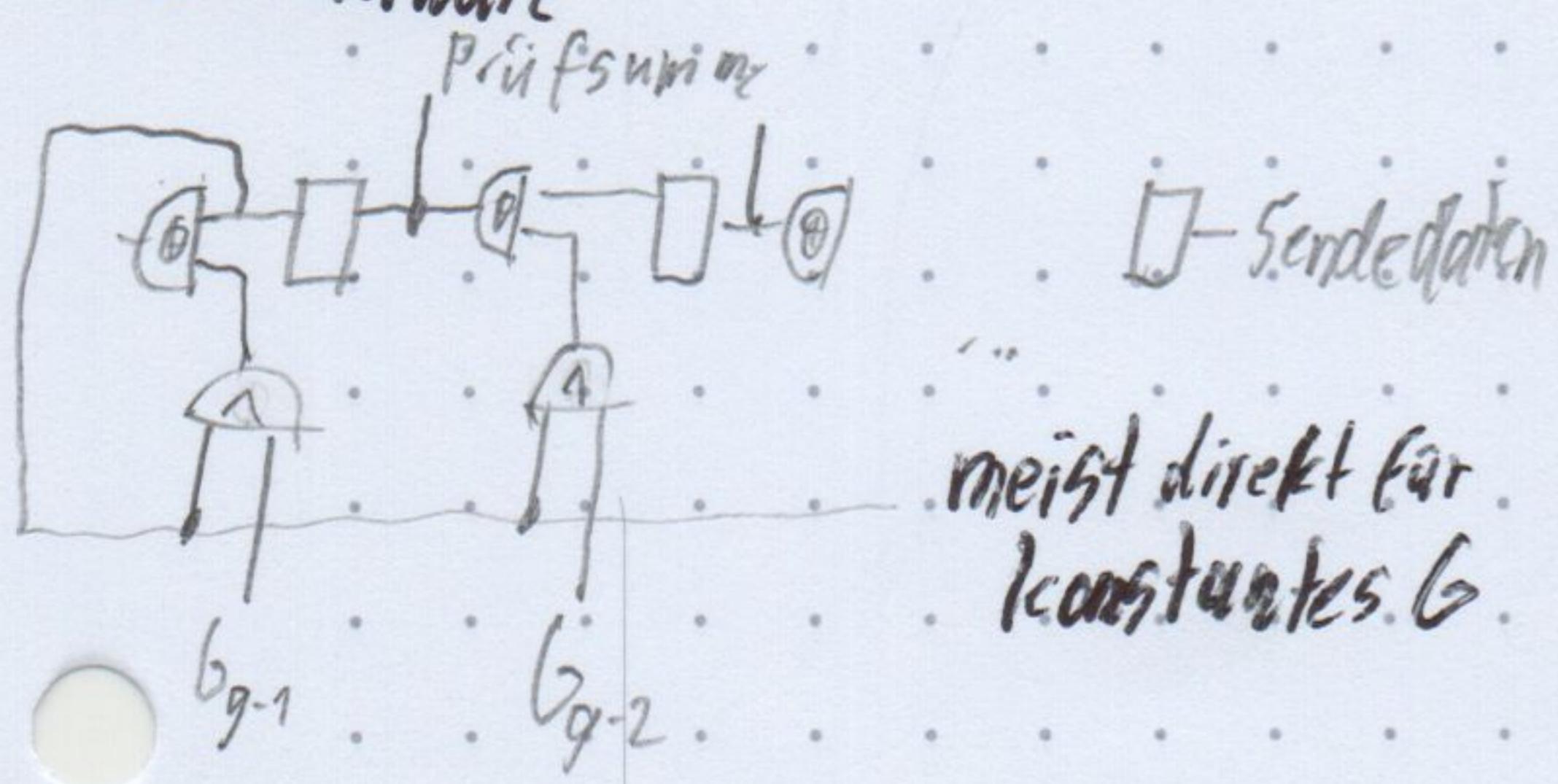
Fehler ungerader Anzahl  $\Leftrightarrow$  G durch  $x+1$  teilbar

Burstfehler Länge  $\leq g$

$\Rightarrow \text{CRC}-16-\text{CCITT}$  all die

99,9% Burst fehler > 11

## CRC-Hardware



meist direkt für konstantes G

Knotenfach-/  
Kanal-/  
Medienzugriffsverfahren

keine Überlagerungen  
Fairness

statische Kanalzuordnung

Zeit-, Frequenzmultiplexing

- Auslastung  
if schwankender Sendebedarf  
if |Stationen| < |Kanäle|

dynamischer Zugriff

zentral z.B. Polling reibum - fehleranfällig

dezentral z.B. Vorab-Reservierung

Token-Passing-Protokolle (Ring, Bus)

wahlfreier Zugriff jeder darf senden  $\Rightarrow$  Kollisionen möglich

ALOHA Stationen unabhängig unkoordiniert - max 18% Zeit sinnvoll genutzt

Slotted ALOHA festgelegte Zeitschlitze (Taktsignale durch eine Station) + keine Teil-Überlappungen  
bei Kollision mit p in nächsten Slot max 36,8%

C.SMA - Carrier Sense Multiple Access

niemanden ins Wort fallen

bei Kollision durch Latenz Abbruch, Backoff Zeit

Kollisionsfenster Jam-Signal, damit es jeder & Prüfsumme zerstört) mitbekommt

(CSMA/CD Ethernet

+ einfache Impl.

Ethernet II

7B Präambel - Sync 1010...

1B Rahmenbegrenzer - Start of Frame

6B Zieladd.

6B Quell

2B Typfeld ehemals Länge  $\leq 1500$

4bit daten

Füll bytes

4B Prüfsequenz

0806<sub>16</sub> IPr4

0806 ARP

8100 VLAN

86DD IPr6

Verkabelung

alle Tapen in einen Ether ! Kabelbrüche alles läuft

Hub sternförmig Punkt zu Punkt + erweiterbar  
rein Schicht 1 Verstärker + nur 1 Verb. Lohn  
+ debuggen

Switches Layer 2 + getrennte Kollisionsdomäne  
oft full duplex station-Switch-Port  $\Rightarrow$  keine Kollisionen

Switching Tabelle Port  $\rightarrow$  MAC-Liste  
Lernt aus Quell-MAC, timeout  
unbekannt  $\Rightarrow$  Broadcast außer Sender  
transparent für Endsysteme  
Flusssteuerungsmechaniken  
sehr Router ähnlich