

(Nur S - quite small, somewhat disorganized)

distributed system - one specified system by closely cooperating computers
fleischende Branche computer network - distributed system

Information

Abstract world
↓ conventions for representation

Data

Physical world
↓ conventions for representation

Signals - characteristic changes of physical variables

stored in wire

propagation delay d - how long to receiver $d = \frac{\text{distance}}{v}$

propagation speed v - (Electromagnetic wave in copper $\approx c$)

data rate r ($\frac{\text{bits}}{\text{second}}$) $\frac{\text{Data size}}{r} = \text{End-to-end transmission}$

Error rate (also error patterns)

Simplex - only one party transmits

Half duplex - parties alternate as sender/receivers

Full duplex - both parties (may) send all times

TDD - Time division duplex

on-demand: each sender preannounces length, break symbol?

two cables expensive overhead

short distance Glasfaser, Frequency Division Duplex FDD

or wireless

FDD - time division duplex - if digital + $\frac{\text{transmission}}{\text{rate}} \geq 2$ user interval

Connect multiple computers:

each with each - way too much, does not scale

structure of end systems and switching elements/routers

circuit switching + simple, guaranteed date flow, no queuing delay
- Resource dedicated, even through pauses, setup needed

packet switching

chop data up into packages with administrative information
continuous data flow \rightarrow occasional package

typically implemented on higher layer

store-and-forward: receive+store \Rightarrow find next hop \Rightarrow forward

Broadcast medium and multiple access

common characteristic

- only one sender

achieved through: multiplexer

Medium Access Control (MAC)

needs common rules

Multiplexing - regulate access to shared resource of multiple users

downward - multiple connections to multitude one

upward - enable many connections over one
demultiplexer needed

TDM - time division multiplexing: one package after another

FDM - frequency division multiplexing:

mainly
wireless

C.D.M - code division multiplexing

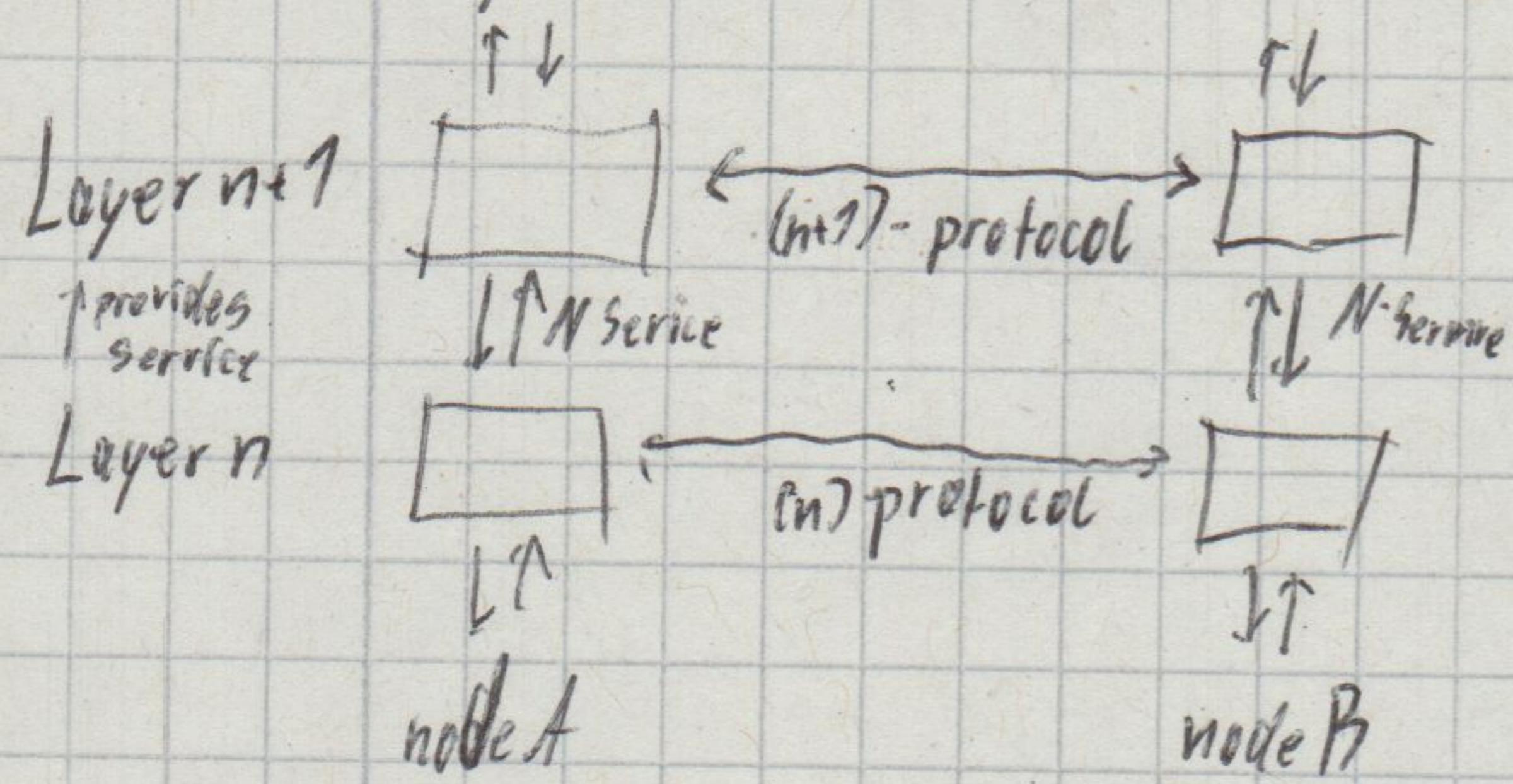
S.P.M - space division multiplexing

Abstraction

Network Layer Models

- ISO's OSI model (Open Systems Interconnection)
- Internet Layer model (basically subset)
- makes some problems transparent = "Distributed Machine"
- AS - autonomous systems
- (SS - communication subsystems)

Layered Architecture



Layer 0 has to perform actions itself

- + independently exchangeable, interoperability, I.B.
 - Overhead
- Tradeoff: overhead \leftrightarrow exchangeability, clarity, simplicity

1. Physical Layer

Non-secure bit stream between adjacent systems

- Signal representation of bits, de-activation of lines
- Transmission time of a bit
- Standards for: plugg, cables
- Protocols: z.B. RS232-C = ITU-T V.24

2. Data Link Layer

Error-recovering stream of frames between adjacent systems

- boundaries detected (frames?)
- recognizes/recover transmission errors
 - through sequence nos., checksums
 - acknowledgment or timeout and retransmission
- severe errors deferred to higher layers
- may flow control - goal: protect slow receiver
 - z.B. sliding window protocols, something to avoid stop and go for satellite

Broadcast Networks LAN 2 Sublayers

- LLC Logic Link Control - error detection
- MAC Medium Access Control - orderly access to simple medium

Protocol z.B. Ethernet

3. Network Layer

Packet stream between end systems

- routing (store & forward switching \leftrightarrow circuit switching)
- upward & downward multiplexing
- error correction & flow control (z.B. Hops different quality)
- congestion control (protect network)

Protocols: z.B. IP (connectionless), X.25 (connection oriented)

BRUNNEN

\rightarrow Overhead

Internet: 5,6 integrated in 7. (layer 3 and 4 somewhat confused)

Protocol

defines format and order of messages exchanged between ≥ 2 communi. entities and actions taken on diff. messages (action rules)
dargestellt in message sequence chart

may contain fragments

OSI terminology

PDU - Protocol Data Unit (Message)

(N)-PDU - semantics understood by peer entities of N-service

PCI - Protocol Control Information: only used by peers

(N)-PDU = (N)-PCI + (N)-SDU

SDU - Service Data Unit = payload optionally carried in PDU

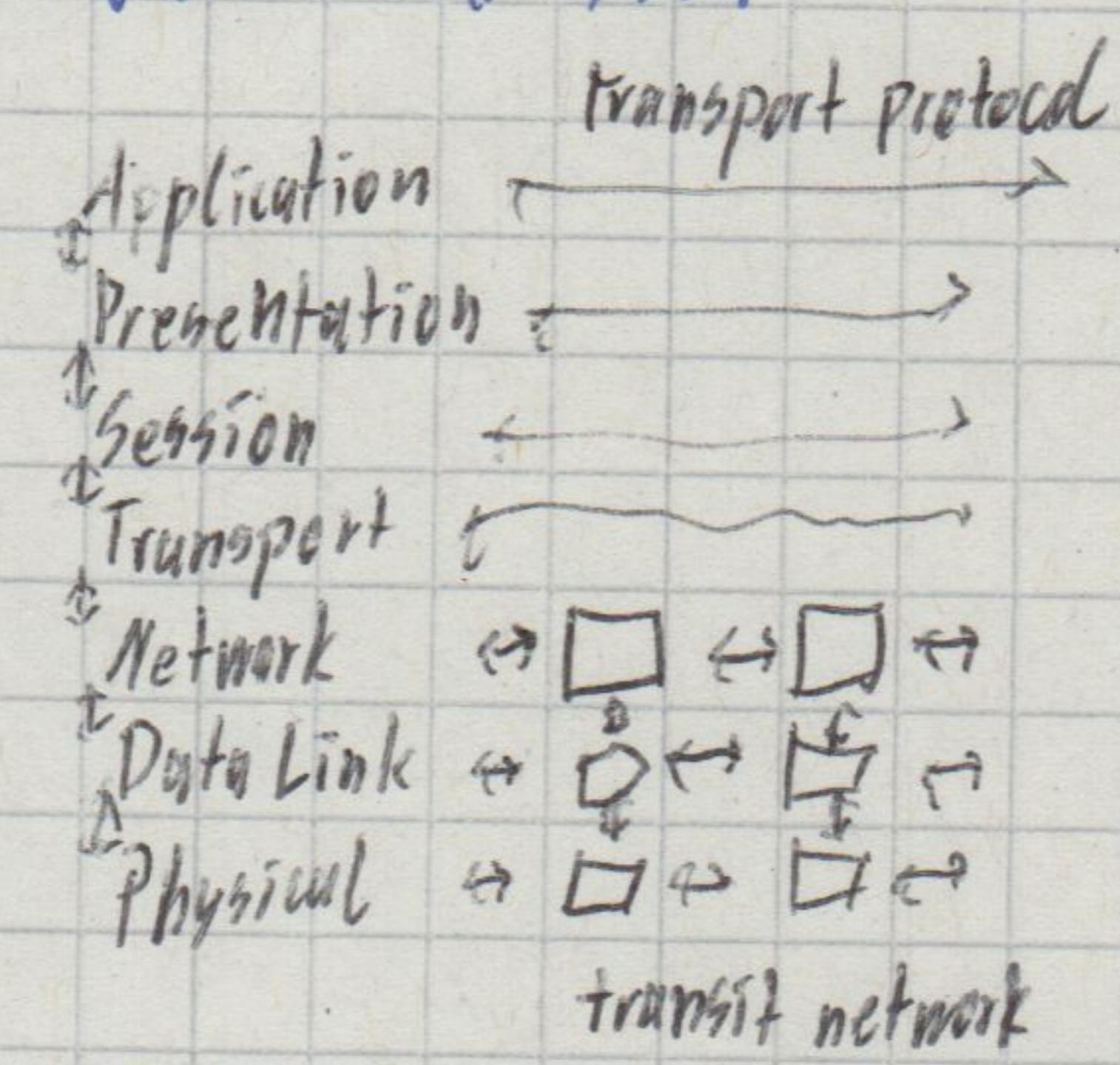
(N)-SDU = (N+1)-PDU

packet - "unit of transportation"

datagram - (connectionless) send packet

frame - with fixed envelope, relay based (deeper)

cell - small packet of fixed size



4. Transport Layer

Logical connections between individual processes (ports, sockets)

Multiplexing, Packet assembly, Address handling, error correction, flow control

Protocol: z.B. TCP

more hide network details, quality diff.
shorter packets allow parallel (pipelining)

5. Session Layer

manages consecutive connections

Tokens (right to request)

communicate about checkpoints (for recovery/rollback)

hardly used (SNMP, PTT telematic services such as teletext)

6. Presentation Layer

harmonize data representations in different end systems

OSI standards: ASN.1 (Abstract Syntax Notation) Compare: (structures) BER (Basic Encoding Rules): common representation between end systems

may encryption

Internet: in Application Layer more pragmatic standards of IPDs

7. Application Layer

service elements (SE) for common tasks

association control

CCR - commitment concurrency and Recovery

ROSE - Remote Operation Service Element

for domains

File transfer (ftp)

Electronic mail (X.400)

huge OSI-pragmatic Internet

Protocol z.B. HTTP

Types of Networks

Connection oriented - CO

~ telephony service
phases

connection establishment
handshake

partners are connected ~ distributed

- distributed state, held on both end points

data transfer can implement flow control
reliability

connection teardown

e.g. TCP

+ better service

- handshake can be significant
in short communications

Connectionless - CL

~ postal service
only

data transfer

no flow control, reliability
congestion control

UDP

→ stateless → scalable

Routing

Routing: determine route

Forwarding: move packets from routers
input to appropriate output

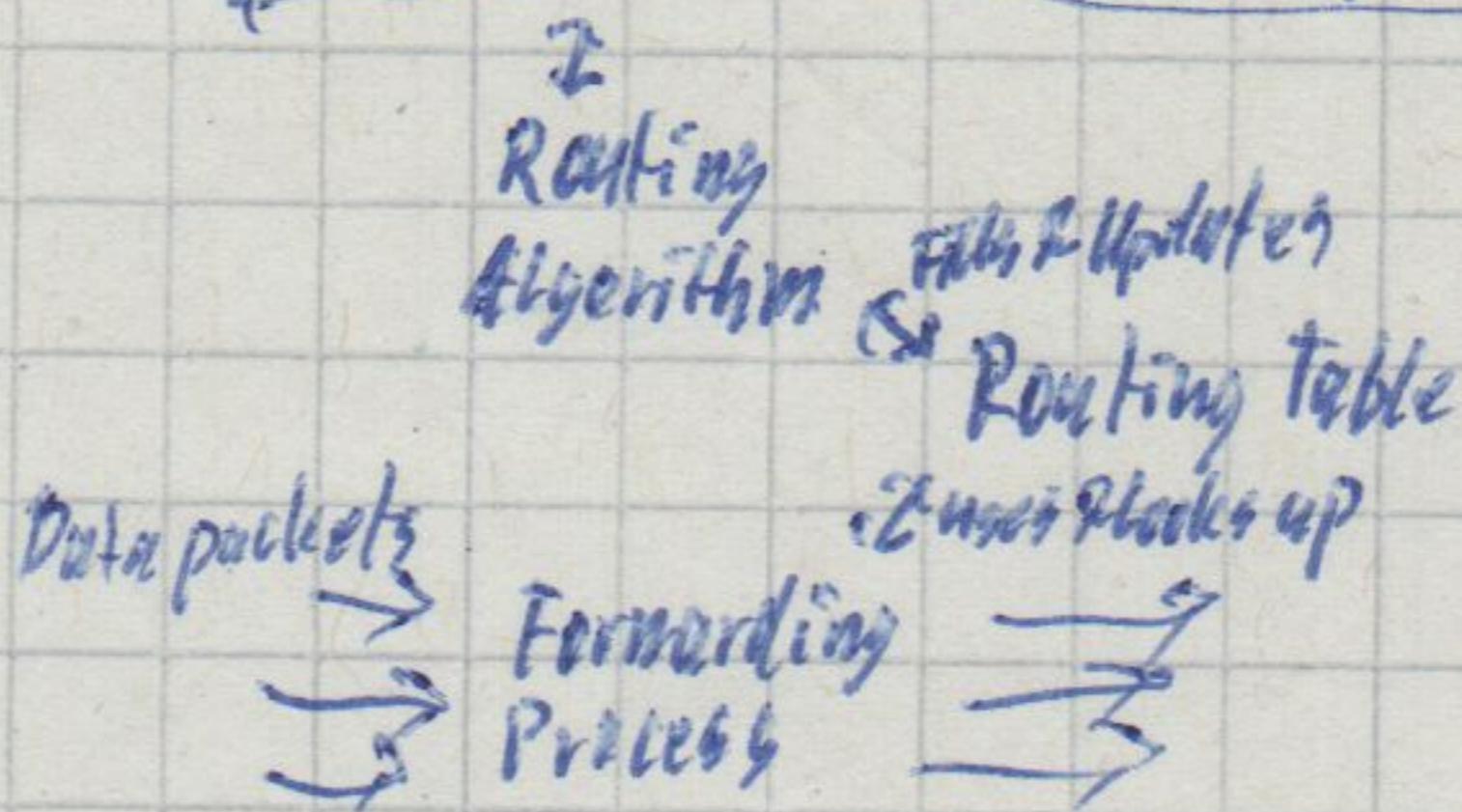
"cheap"

"good"

find route in network of endsystems & routers

Router { control plane $> 10^3$ pps or control agent
data plane $> 10^9$ pps

routing information



SDA controller

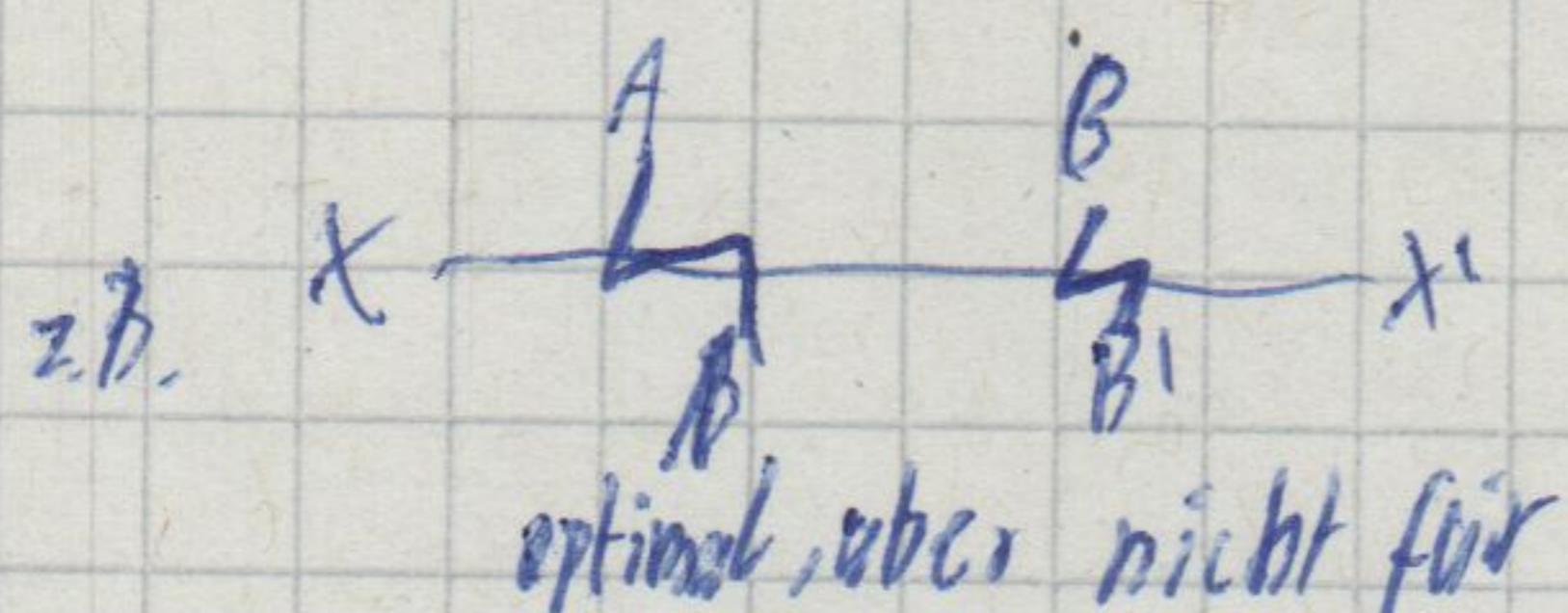
CL vs CL on Network Layer Services (NLS) - CLNS every intermediate system knows connection

CLNS - datagram service: routing at setup, forwarding through (short) "circuit ID"

CLNS - "virtual circuit": routing algorithm usually periodically or at first for destination

Goals:

Correctness
Simplicity
Robustness
Stability
Fairness
Optimality



optimal, aber nicht für

often using metrics:
average package delay
total throughput
Individual delay

practise:
mostly minimize
hops

Routing Algorithms

Non-adaptive routing algorithms: don't base on current state

Flooding: sent to everyone but source and endsystems

reducing packages:

- TTL: max number hops (needs time long enough to find dest) (erstmallich benötigt)

- sequence numbers: only forward once

+ fast changing networks

+ multicast to many anywhere

z.B. mobile

z.B. Application (if hops greater)

distributed database

Static Routing: preconfigure all possible dest.

+ in static, predictable environments

- networks change frequently

- if traffic varies → huge over provisioning

→ Adaptive Routing Algorithms: use network state (traffic-adaptation only with care, because it changes fast, more overhead)

Centralized adaptive routing:

Routing Control Center RCC

periodically routers forward links+state
center computes (z.B. Dijkstra)
dispatches to all routers

- vulnerability: from RCC

- scalability

- Bottlenecks: traffic around RCC

- Aging: close-by routers learn earlier

Isolated adaptive routing

Backward Learning Routing

source address and hop counter in header
remember link with minimal hop count from ~~to services~~
start with random routing: hot potato offloading

"soft state": periodically forget
to adapt to link failures, ...

Hot potato routing: get rid of packet as fast as possible

- usually select port with shortest queue
- does not care where output line leads

- not very efficient
- + frequently used (e.g. peering)

Distributed adaptive routing

Graph abstraction: nodes - routers, edges - links (link cost, Path cost = sum of link costs)
find shortest path (not always guaranteed)

delay

Decentralized:

router knows its links and cost

iterative computing, exchange

distance vector: e.g. RIP protocol, BGP protocol (pathvector)
algorithms

static: routes change slow

dynamic: routes change more quickly

- periodic update or

- respond to changes

Global:

all routers full topology

"Link State" Algorithms

Dijkstra's algorithm

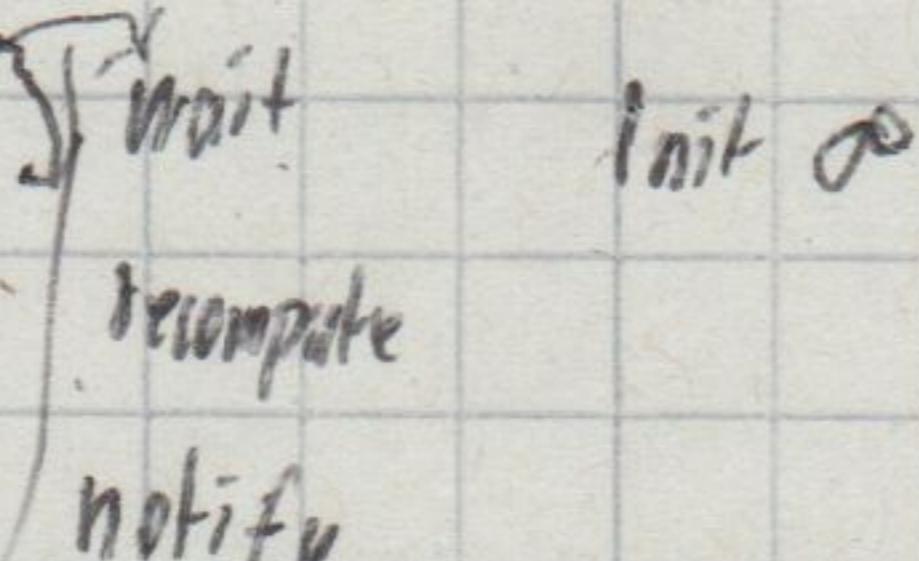
OSPF protocol

DVR

Distance Vector Routing (distributed, self-terminating)

data structure: Row for each possible destination

Column for each direct neighbour



$$D^x(y, z) = \text{dist } x \xrightarrow{\text{via}} z$$

$$= c(x, z) + \min_{w \in N} \{ D^x(y, w) \}$$

distance matrix

DTS

Neighbors

cost

routing table

link, cost

cost

distance vector

dest, cost

dest, cost

changes: good news travel fast

bad news slow - "count to infinity" - problem

if 2 routes through x to y , x tells y its distance to x is ∞

Split Horizon / \Rightarrow Poisoned reverse: if something gets worse don't fix

sends it with no fix in bigger network

Q-B

Q

Q

doesn't even send update \Rightarrow Split Horizon: don't advertise to neighbour that send update to full fix

errors propagate through network

Link State Routing LSR (global)

input for algorithm
graph (V, E)
nodes \rightarrow costs
mapping $c(v, w) = \text{cost}$
2. B. Dijkstra

links
exchange: link state broadcast
 $O(n \cdot E)$ send each time
each router builds link state packets and floods them
cost to all neighbors
sequence number, age field
Speed of convergence is linear $O(n^2)$ algorithm.

Hierarchical Routing

Growth of net may be fast for flat network

Administrative autonomy: each network admin may want to control intern routing

Increasing number is a problem

Potential prob

Autonomous Systems (AS) (~60,000 today)
f. Stub AS: small corporation (only one link to internet)
f. Multihomed AS: large corporation (multiple links, but no transit)
Transit AS: provider

everyone unique number
every AS must know a route to every network
diff relations z.B. equals or provider/customer

solves scalability

Gateway Routers

inter with other gateway routers
intra with routers of AS

can focus on performance

intra-AS routing: choice of each admin

z.B. RIP - Routing Information Protocol: Distance Vector 1983; hop count as metric; poison reverse

OSPF - Open Shortest Path First: Link State open-publicly available; routing info through TCP; 2 metrics; integrated multicast support

IGRP - Interior Gateway Routing Protocol: Distance Vector (Cisco proprietary)

one for best-effort realtime

of subnet and gateway

policy may dominate

inter-AS routing

BGP - Border Gateway Protocol: Path Vector (allows loop avoidance)
de facto standard

gateways need to propagate reachability info to all routers
often hot potato routing: choose gateway with least intra-cost
that can reach x

BGP peers have BGP session (semi permanent TCP connection) (not 1:1 to physical links)

advertising premises to forward

i BGP - between AS's

l route = Prefix + attributes

j BGP - between routers of AS

l AS-PATH

session

l NEXT-HOP (specific gateway router)

gateways accept/decline after import policy

Potential Problems

Sub-optimal routing

advertising after policy

Route instability

z.B. a dual homed customer may
not want to forward between its providers
only to/from its customers

Route oscillation

Route flapping

Security

blackholing

redirecting

Mobile Routing

Dynamic Source Routing - DSR (IETF RFC 4729)
step towards reactive routing designed for up to 200 nodes
source routing → sender determines path

path discovery

only if needed and unknown
flooding with destination and unique id
every hop adds its own address
destination returns packet to sender
(without bidirectional paths flooding back)

path maintenance

eventually
only if useful, unused paths get deleted

or

layer 2 acknowledgement (e.g. WLAN)

Carrier sense: listen if other station forwards (doesn't work with
Request explicit acknowledgement directed antennas)

in case of problems in intermediate node

inform sender
or backup new path

Overlay Routing z.B. peer-to-peer, most cloud/distributed shared memory setups

virtual network topology overlaid over real network z.B.
Application overlaid on top P2P over IP over network links
typically means ~~over~~ IP network
on top

Routing: same in principle

stretch: path through overlay longer than underlying network

metric: minimize overlay hops
stretch

assumptions for P2P

nodes cooperate

nodes arrive and depart randomly (churn)

nodes and resources are anywhere on net

degrees of freedom: node identifier, neighbor selection

Discovery of resources
know either name or location

solution classes (strategies adapt to variation of assumptions)

Organization:

flat/random, hierarchical, structured (with mapping?)

Delegation

recursive, ~~dele~~ iterative

Internet Network Layer

Routing Protocols → Forwarding Table

Internet = Connectionless

forwarding nach dest. host address

IP Protocol - addressing conventions

Datagram format

Packet handling conventions

ICMP protocol - error reporting

"Router signaling"

IP Header, in 32 bit words | priority info
rarely used

total datagram length (bytes)

decrement at
each router
max number
of hops

upper layer v
protocol to
deliver payload
to

version	head length	type of service	length	
76-bit identifier	flags	fragment	flags	fragment offset
time to live	protocol	internet checksum		header checksum
32 bit source IP-address				hole line remarkably for fragmentation
32 bit destination IP address				
Options if any				
data (variable length typically TCP or UDP segment)				

z.B. für

source routing

record route

Fragmentation

all fragments share 16-bit identifier

Flags: DF: Don't Fragment

MF: More Fragments (set by all but last fragment)

Fragment Offset = original position in octets (unit)

MAC
ARP, IP
Port
Host name

flat, permanent
topological (mostly)
more... de hierarchical

Flat namespace: - conflicts, - central work, - inflexible

Hierarchical Name Space: decentralising, Partitioning in Zones to delegate, recursive (z.B. DNS)

IP address classes (two layers of hierarchy + subnets)

classfull: A

0 | Net | Host

8 bits 24 bits

B

10 | Net | Host

7 bits 16 bits

C

110 | Net | Host

24 bits 8 bits

D

1110 | Multicast address

E

1111 | Reserved

specials Network Address: all host bits 0

Broadcast Address: all host bits 1

Loopback 127. anything for loop back testing
processed locally

Private IP addresses 10/8

172.16/12

192.168/16

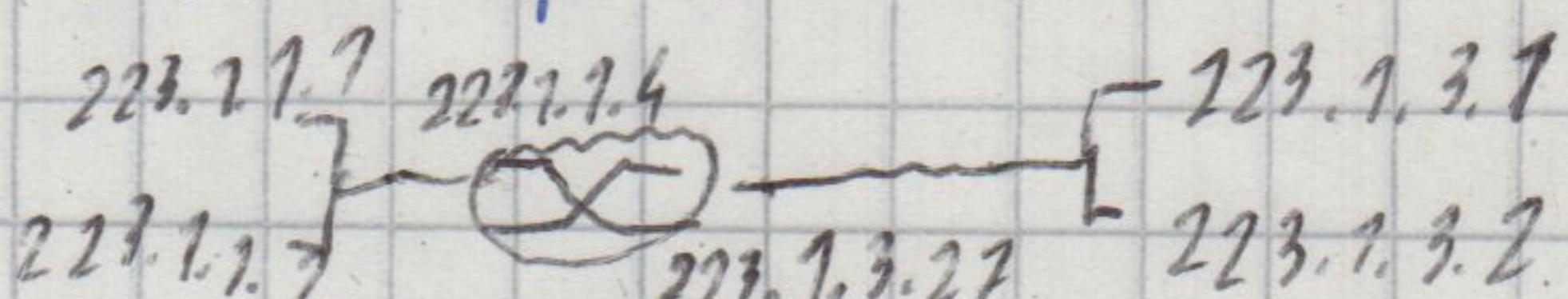
Link local 169.254.0.0/16

Network Prefix = IP & Net Mask \Rightarrow IP/H1 in Net Mask (immer fürend)

Routers usually multiple interfaces each with own IP

\Rightarrow IP for host and router interface

\Rightarrow connection between two routers is own subnet!!



Subnet: device interfaces with same subnet prefix

can physically rely on each other without router

+ organization may want to split departments
each admin own address space

BRUNNEN

all detached islands from router

+ some departments may be smaller

class A can have only 2 subnets \Rightarrow VLSM (variable length subnet mask)

multiple subnet hierarchies possible

CIDR = Classless
InterDomain Routing 8

CIDR: supernetting (discard class boundaries)
longest match routing
ISP assigns contiguous group of class C blocks
advertise may as one
with either IP or MAC

ICANN
Internet Corporation for Assigned Names and Numbers
- non profit org Los Angeles
Carries IANA function
numbers Authority
Allocate addresses
Manage DNS (delegate to existing orgs)

→ ISP

→ Organization

→ Host

static: hard coded by sys admin
or retrieved from local server RARP

reverse DNS
DHCP: Dynamic Host Configuration P.
configures IP address
default gateway
(+ plug and play =)

example for forwarding table

Prefix Match	Link Interface
1100 100 ...	0
... 000 11000	1
... 00011	2
otherwise	3

NAT - Network Address Translation - Motivation: Shortage of IP addresses

router übersetzt interne Adressen nach externen in single durch Zuweisung ports

+ only one (external) IP for network

+ IP address external intern sind unabhängig

+ Devices not explicitly addressable (Security?)

private IP addresses nur noch in lokalem Netz eindeutig

linklayer

router muss outgoing datagram replace IP and port
remember in NAT translation table

WAN side addr

LAN side addr

- router needs to use layer 4
- violates 2E principle (z.B. complicated for P2P applications)
- no servers behind NAT (server classes reached by port)
- IPb better solution

IP → MAC

Address Resolution Protocol

ARP: Broadcast IP asking for MAC
cache

1. broadcast DEST MAC =

2. unicast back

3. caches in IP-to-MAC pair in ARP table
with timeout

+ plug and play

- ARP cache poisoning

IPr6

128 bit addresses
Routing and switching works in same way
new addressing architecture
new protocols

reasons: more IP addresses
(get rid of NAT?)

(Practical limit of 250M)
technical reasons:

"Traffic class" + "flow labels" may
enable (better actually used) QoS
flexible number of hierarchy levels
wireless plug and play possible
end-to-end, IP-layer authentication & encryption
better for mobile IP ("triangle routing") possible

ICMPv6

Error Messages

Destination Unreachable
Packet Too Big
Time Exceeded
Parameter Problem

Informational Messages

Echo Request
Echo Reply

NDP - Neighbor discovery protocol
MLD - multicast listener discover

NDP replaces ARP (bad mapping attack vectors)

used for

Discovery routers, prefixes

network param

Autoconfiguration

DAD - duplicate address detection

NUD - neighbor unreachable detection

Address Resolution

Messages

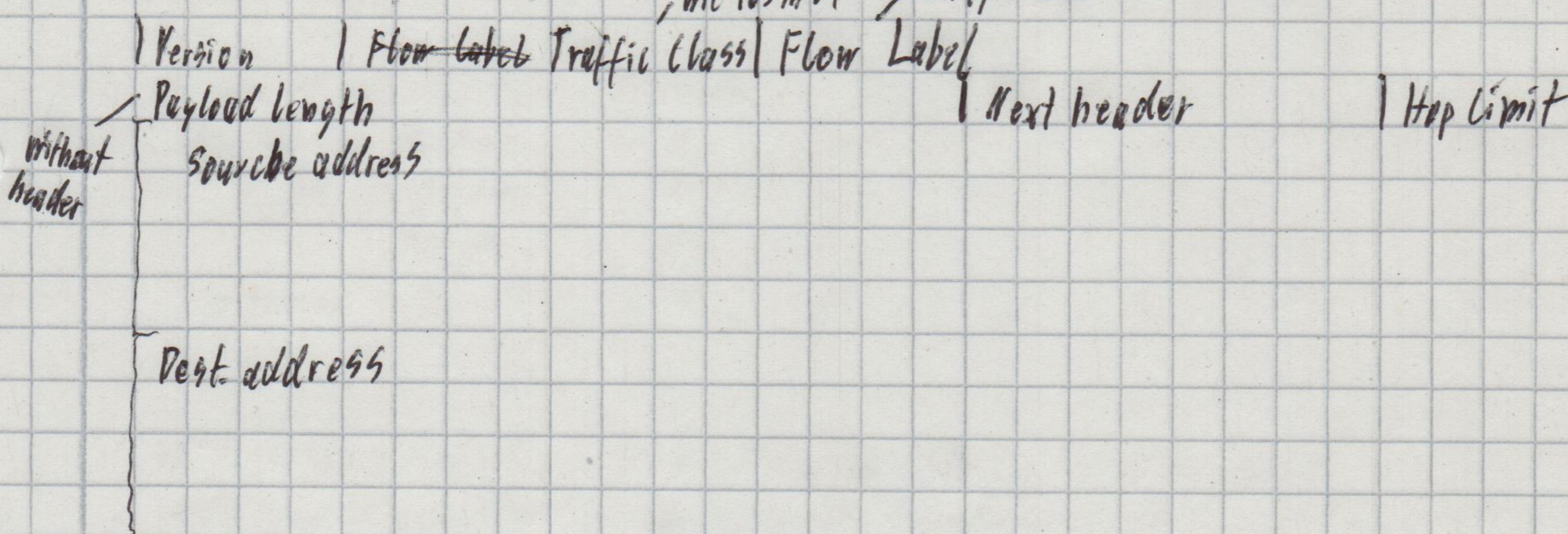
NS - neighbor solicitation

NA - neighbor advertisement

RS - router solicitation

RA - router advertisement

with Tos in v4, identify connections but forbidden now!



Next header: TCP, UDP or one of several v6 extension headers

IPr6 Addresses

128 bits: written as 16 bit hex numbers one == ellipsis filled with 00 bis 128 bit leading zeros required

Aggregate Global Unicast Addresses

~~0011TLA ID~~

Next level site-level may be based on NLA (FFFF inserted)

7th bit inverted)

but that would be
big privacy concern

0011TLA ID | Reg | NLA ID | SLA ID | Interface ID

3

13

9

24

96

64

Top Level Aggregation (ith NA aggregated global IP)

reserved for extension of TLA and/or NLA

affects related protocols

Transition from IPr4 to IPr6

nicht alles auf einen Schlag umgestellt
→ interoparable

ARPD erweitert durch NDP oder ND uses ICMPv6

Link-local addresses mandatory

FE80::/10 prefix MAC, DAD, allows IP multicat

Tunneling

IPr6 packets carried as payloads in IPr4 packet

IPr6 router usually also can IPr4

DIICPv6 exists but SLAAC recommended

Stateless Address Autoconfiguration

Router announces its network (RA)

remaining 96 bit determined

a) automatically via EUI-64

b) privacy extensions SHA1 (current ATP to

MAC) + 64bit

DAs: new AAAA records
more new functions

Transport Layer

Logical communication between processes (run in end-systems)

Service primitives to access SAP

semantic

Name: CONNECT, DISCONNECT, DATA, ...

Type: request, indication, response, confirm

Parameters: (parameters) header fields, payload

prim. type characterizes primitive

Usually described in Time Sequence Diagrams ('Message Sequence chart')

• req., conf only for confirmed services

Prefix for Layer

Addressing

OSI suggests identify remote process via local CEP

Problem: How to initially identify?

Idea: operating systems PID

Problem: arbitrary; changes with restart, only one service per process

→ abstract service name or address (e.g. fileserver)

Internet: abstract service addresses = Protocol Ports

Process can register ports. OS queues packets for ports

unique Socket = (<IP-address>, <port number>)

(<local socket>, <remote socket>) fully identifies connection \Rightarrow violates OS!

Demultiplexing in Internet

Host receives IP datagrams what if IP
TCP inspects full socket pair replaced?

well known ports 0 - 1023 IANA (requires root!)

echo 7/udp

ftp 20/tcp filetrans

time 37/udp

21/tcp control

mem 42/udp

22/tcp ssh

dns 53/udp

25/tcp smtp

short lived, ephemeral ports

dos 53/tcp

http 80/tcp

443/tcp https

registered

79/tcp finger

110/tcp pop3

143/tcp imap

199/tcp rsh

427/tcp rlogin

513/tcp rcp

1024-65535/tcp dynamic

1024-65535/udp dynamic

1024-65535/tcp dynamic

Errors during Establishment

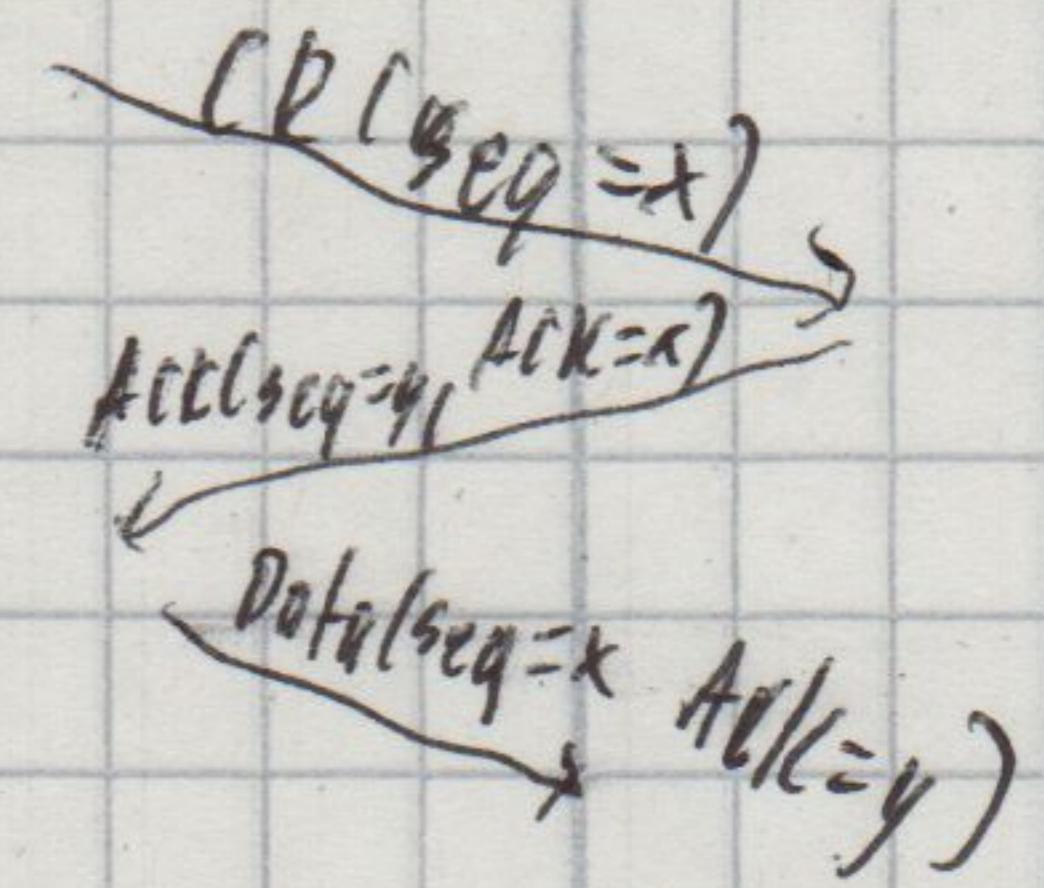
Loss of CR \Rightarrow Timeout resend but Duplication

Loss of CC \Rightarrow Three-Way Handshake (ACK or Data(DT) acknowledges CC)

but delays, duplication or replays \Rightarrow sequence numbers

reuse after packet-lifetime
or timestamps

only establish with right sequence



Connection Rejection

Connection Release

Tear down before data can cause data loss

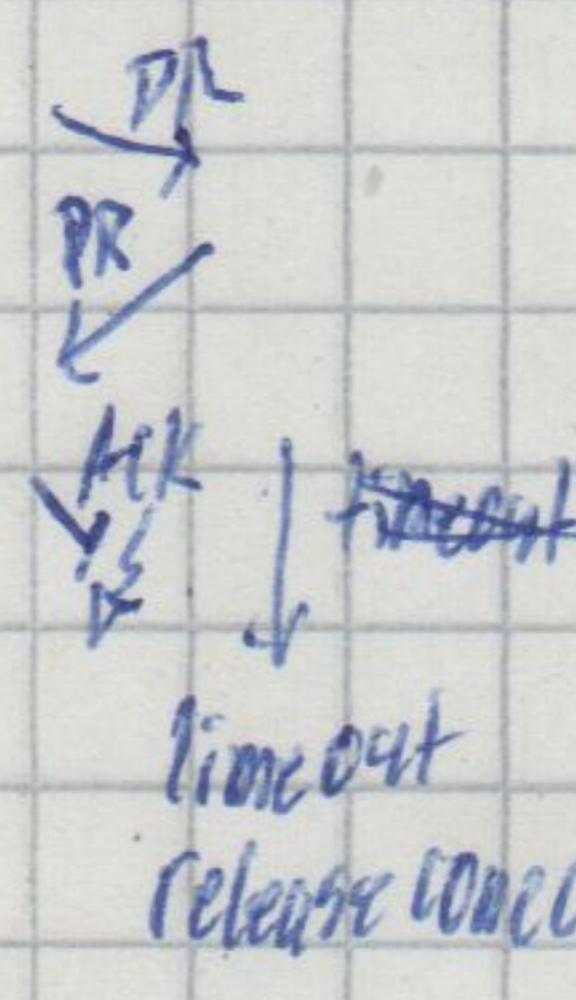
Implicit: Tear down of network layer connection

Explicit: connection release with Disconnect-TPDUs

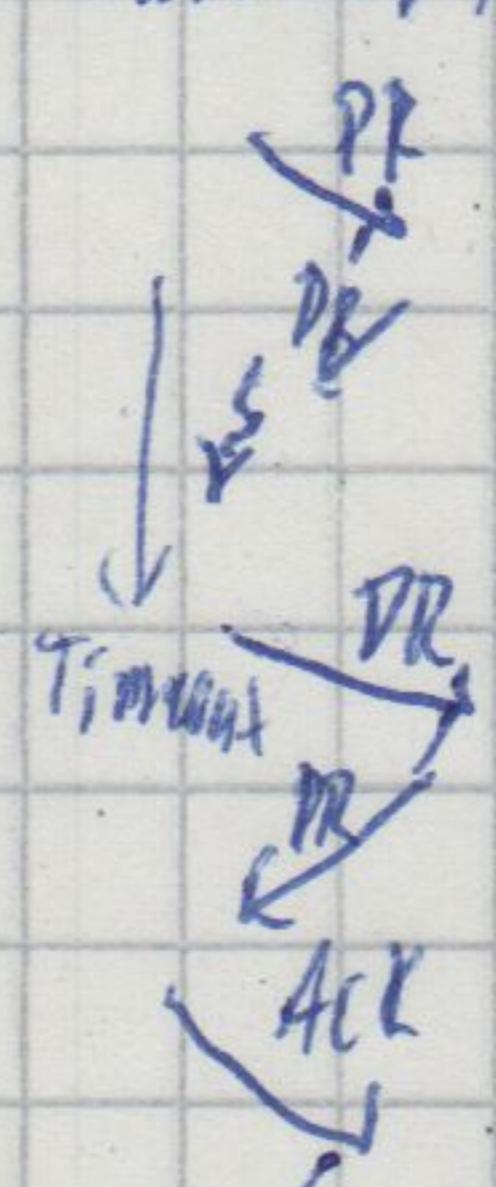
(slide 06 p. 36)

but how to be sure the ~~both agree~~ on teardown \Rightarrow two army problem impossible

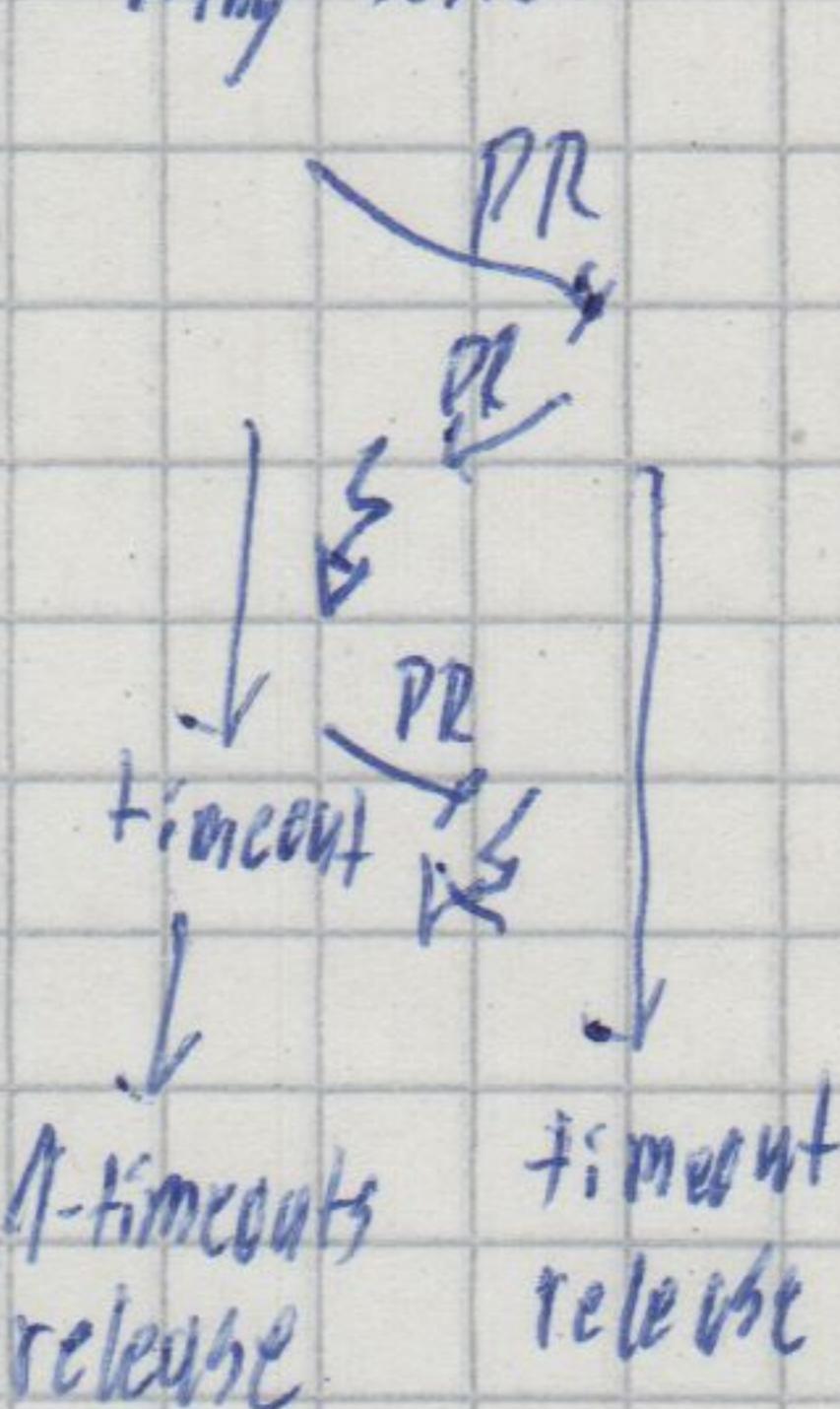
Praxis: 3WHS scenarios lost ACK



lost 2. DR



many losses



Possible errors

dropped packets (bit errors, congestion, ...)

erroneous

receiver slower than sender

Flow Control - protect slower receiver

on Art-Layer: PDUs size differs widely

processes might take time to answer \Rightarrow T-layer needs rebuffer

parallel connection \Rightarrow dynamically

can't assume receiver

is fast enough has enough buffer

manage sending rate

slow when error

sender reduces

receiver gives credit

Alternating Bit Protocol

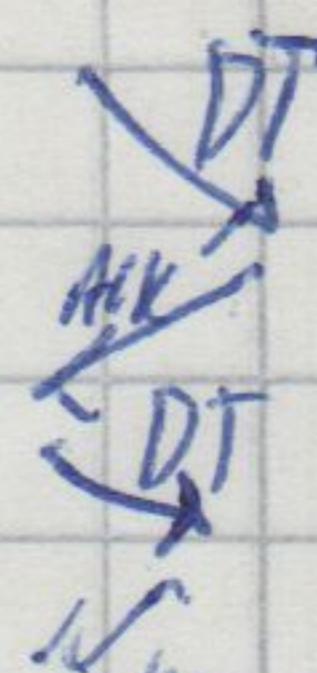
reliable

assumes free buffer after receiving

stop and go!

Only ACK if processed

only 2 packet numbers



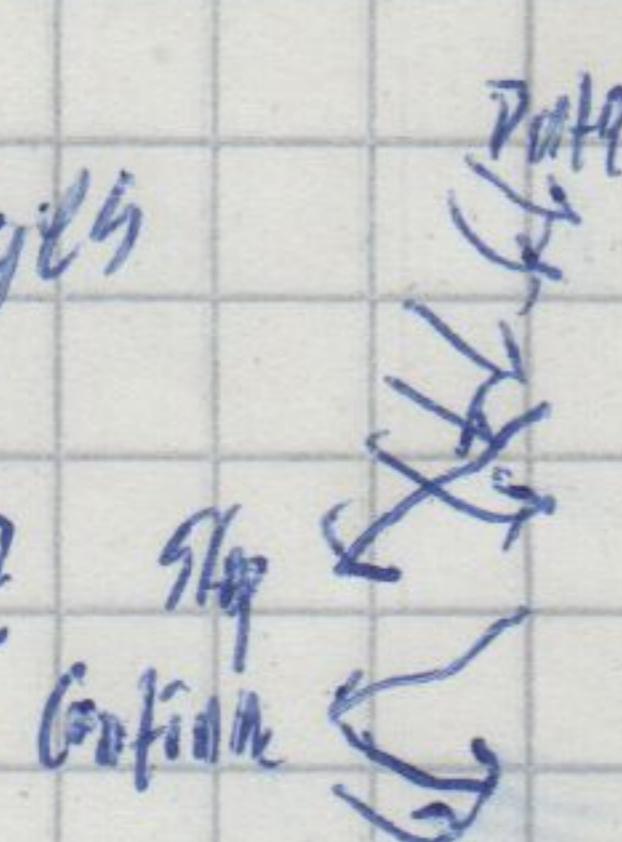
short Timeout? \rightarrow late retrans.

- slow

Step and Continue Messages

e.g. XON/XOFF protocol

arrives stop before start???



BRUNNEN

$$T_{prop} = \frac{d}{v}$$

Länge Weg
geschwindigkeit

$$T_{trans} = \frac{\text{size}}{\text{capacity}}$$

bit
size

rate-based - receiver indicates max rate

+ lower overhead - rate can get messed up by delay
- no confirming file - no error handling

credit-based - receiver indicates max packets
absolut credit - sender only sends with credit

- might get received at different time
credit window (sliding window)
credit in relation to acknowledged packets

credit based

primitives?

Permitset

ACK

L indicates

ack \Rightarrow stop timeout, delete local copy

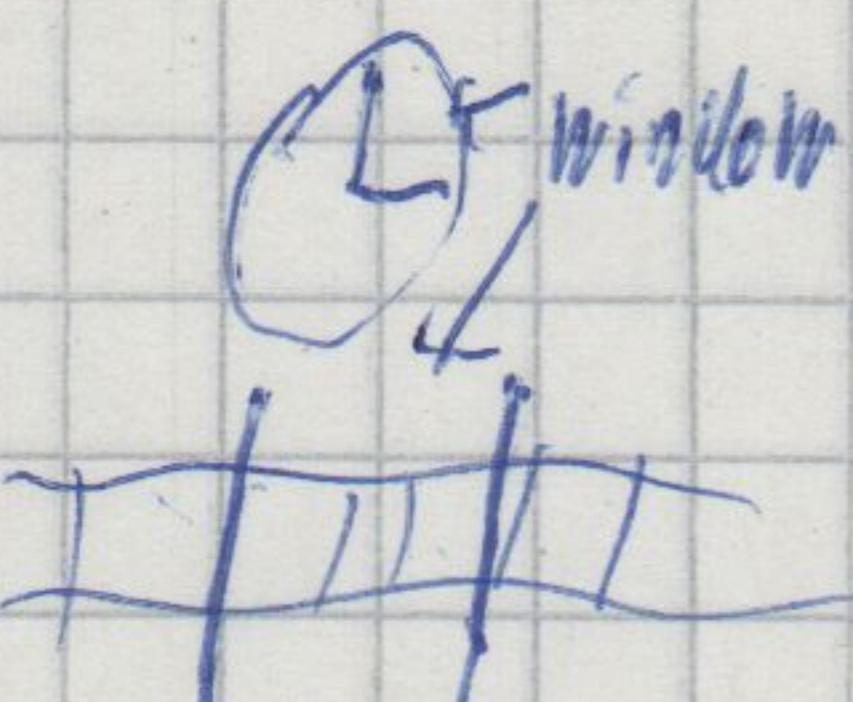
to process more

flow advance \Rightarrow advance window, possible send more data

problem if ack retained

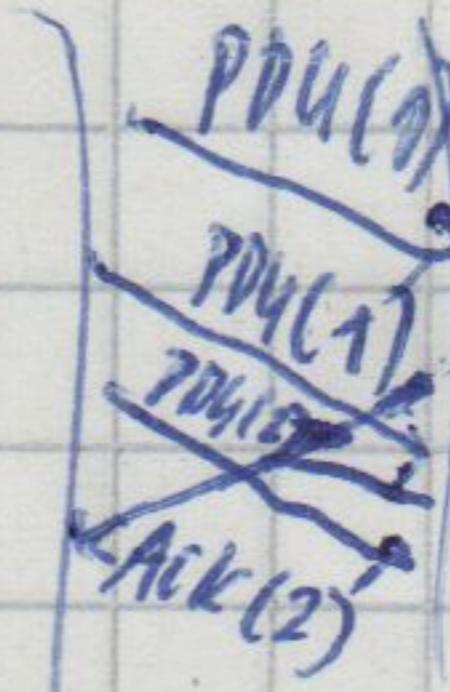
sender timeouts and resends

Sliding Window



example

with explicit permits



für nächstes
erwartetes Paket

Error Control - reliable transfer

Detection: wrong checksum \Rightarrow drop packet (error could be in header)

receiver indication ~~would be nice but what is still in routers?~~

sequence numbers

\Rightarrow only reliable scheme timeout @ sender \rightarrow resend

ARQ - Automatic Repeat reQuest

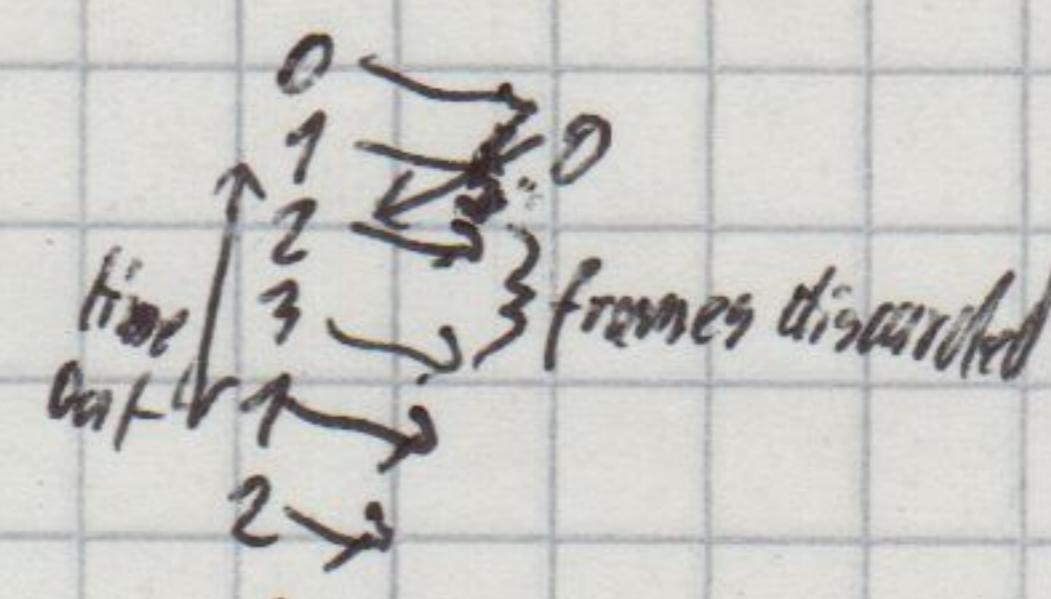
{ type 1 = Stop-And-Wait (siehe Alternating-Bit-Protocol)

type 2 = Go-Back-N

↳ considers current and consecutive packets lost

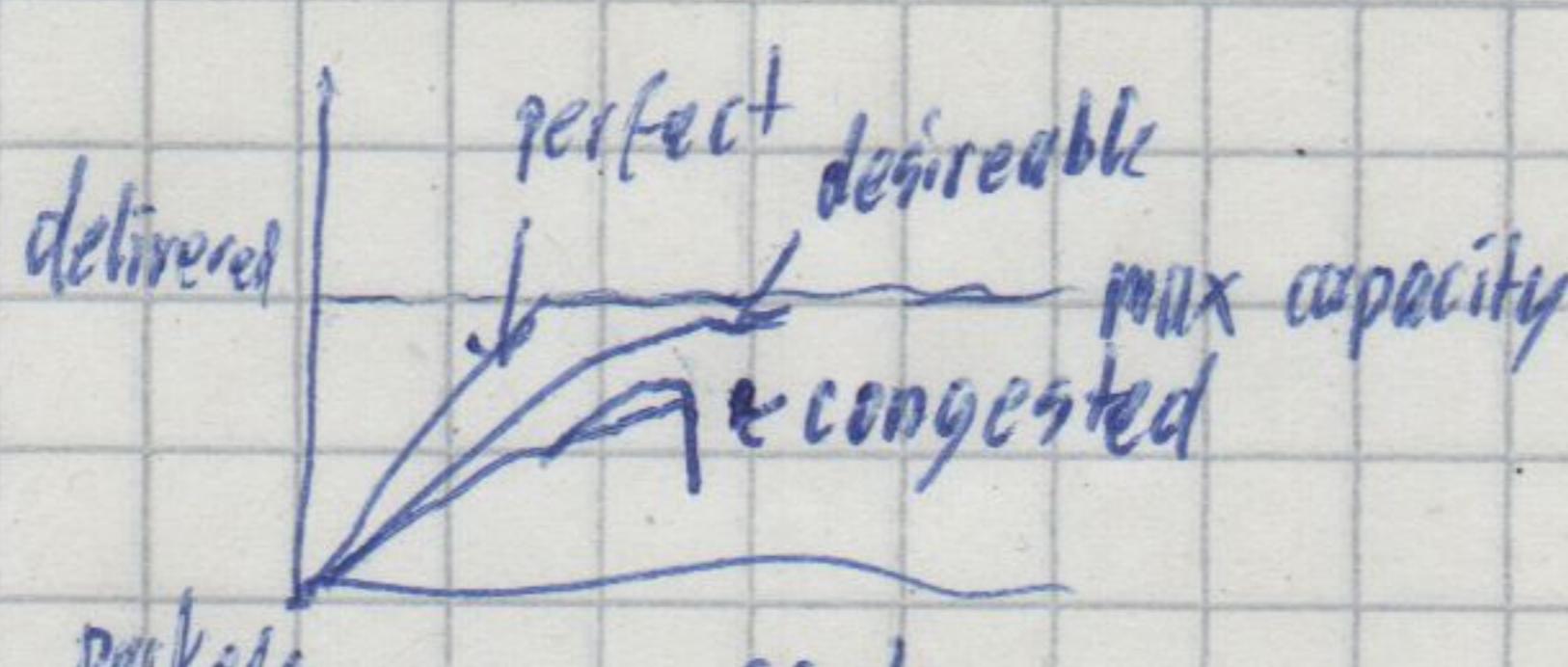
selective Repeat

↳ only retransmit lost packets



Congestion control

what with bottleneck in network?

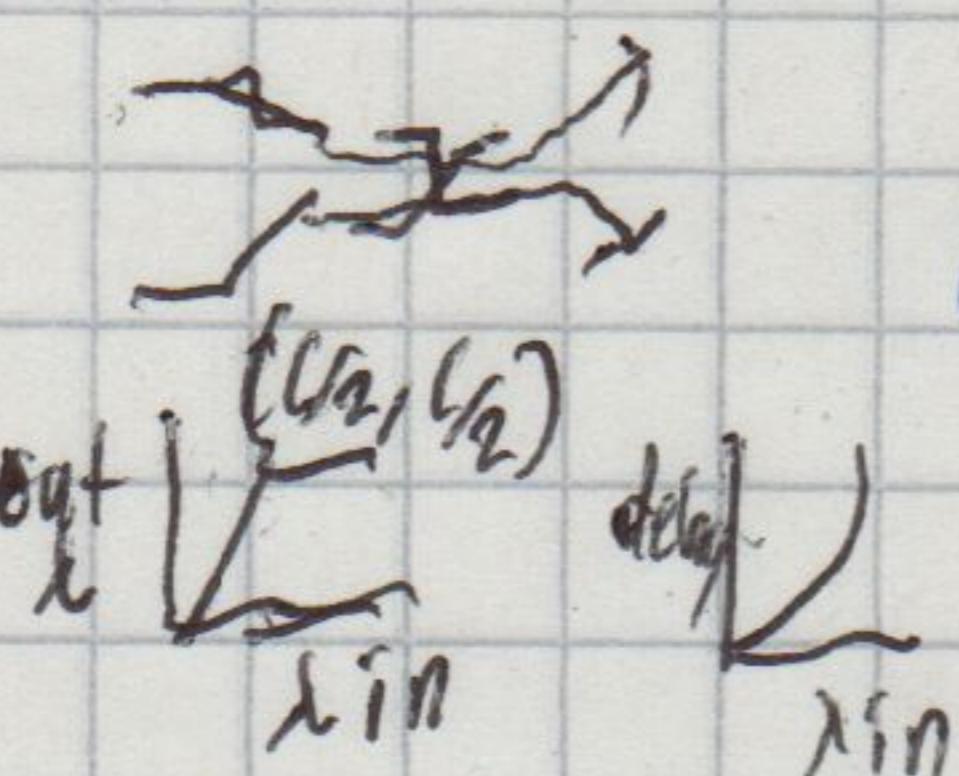


scenario 1

2 sender, 2 empfänger

infinite buffer

no retransmission



2)

finite buffers

\hookrightarrow lost packets

$\lambda_{in}^1 = \text{original data + retransmitted}$

goodput $\lambda_{in}^1 = \lambda_{out}^1$

perfect: only lost packets retransmitted

$\lambda_{in}^2 = \text{original data + retransmitted}$

send only when room in router

retransmit only lost

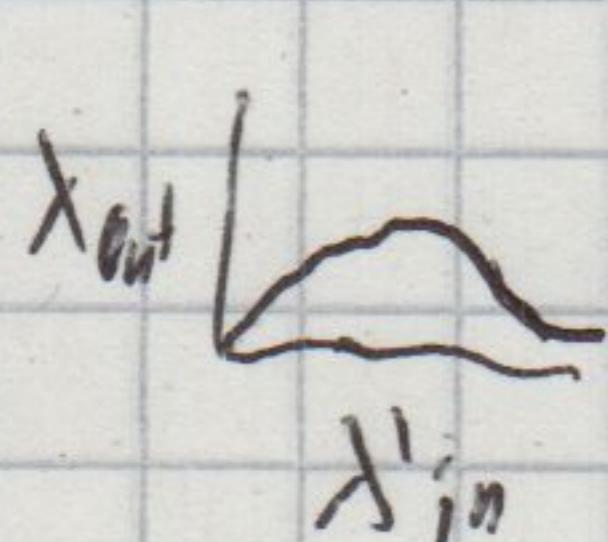
$\lambda_{in}^2 = \text{original data + retransmitted}$

retransmit after timeout

Congestion \Rightarrow more work for given goodput

3)

4 sender
multihop paths
timeout/retransmit



- any upstream capacity
for dropped packet wasted

control essential to avoid snowball effects

\downarrow
sending rate adapt to networks actual capacity

global issue \Rightarrow Flow Control local

'depends on all routers, ranking, ...'

Congestion Control

Goals:

- most possible data (and short delay)
- prevent congestive collapse
- Fairness

detection

Open loop: no corrections at runtime

- Act at source (Admission control) (check before connection)
in Establishment
- Act at destination (ACK)

dropping packets

- Congestion should never happen
else reservation or design wrong

Closed loop: feedback allows sender to adapt

Explicit feedback: point of congestion informs sender

Implicit feedback: sender deduces congestion from network behavior
(e.g. missing ACK)

Actions:

increase capacity = usually not practical (long term)
maybe

reservations and admission control: don't allow additional traffic if near limit

Usually when feedback of state rare, e.g. for circuit-switched or similar

reduce load: without terminating on-going connections
usually requires closed loop

taxonomy

router centric vs. host centric

window based vs. rate based vs. ... (z.B. credit based)

Internet (except for 'dropping packets')

Router actions - dropping packets when buffer full

which?

drop-tail queue (old packets important for go-back-n transport protocol)

- old packets (newer for multi-media traffic more important) (maybe even drop from same flow but overhead)

Feedback

explicit

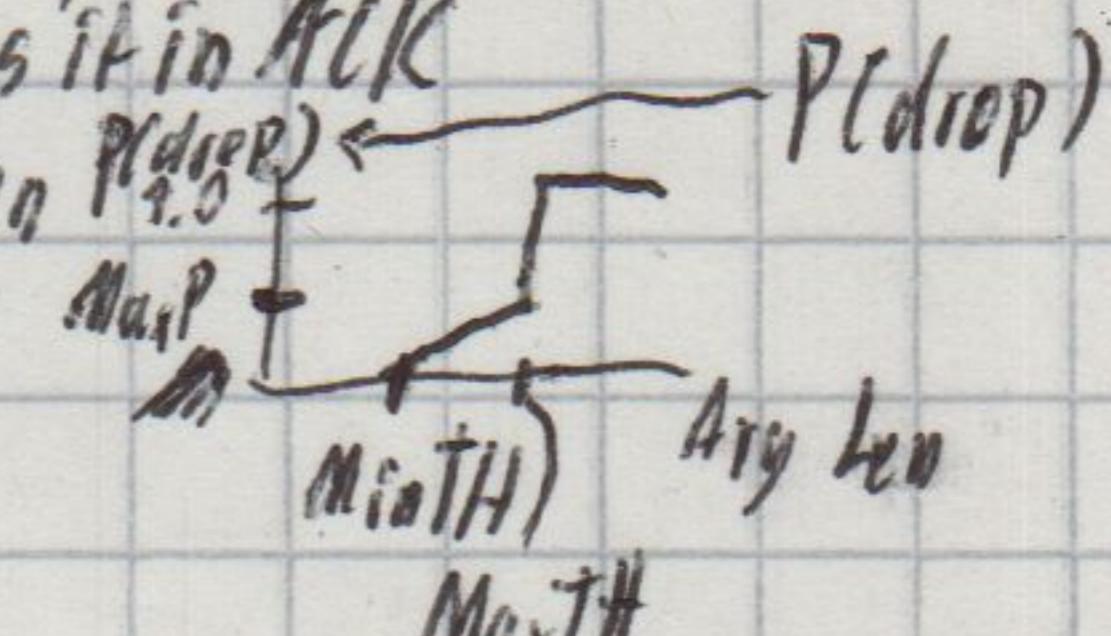
implicit assumes loss only caused by congestion + reduce rate \rightarrow for wireless (occuring L2 layer)

proactive warning state if exceeding a lower threshold

choke packets (tells source of packet to slow down) - delay - more packets

Warning Bits: router sets warning bit in packet destination copies it in ACK

RED - Random Early Detection: dropping probability relative to congestion



UDP - User Datagram Protocol [RFC 768]

connectionless (each part segment independently)
best effort

reliability can be implemented in Application Layer (application-specific)

+ multimedia apps

loss tolerant (neuter frame rate sensitive am wichtigsten)

Demultiplexing = sockets

source port = return "address"

socket identified by ident

TCP - Transmission Control Protocol (TCP)

Point-to-point No multicast

Reliable in-order byte stream

Pipelined

Full duplex (MSS = maximum segment size)

send & receive buffers (old TCP used GoBack N)

socket identified by source, dest pair

Connection Establishment

active mode (connect)

request connection with transport service user (Pt+port)

passive mode (listen/accept)

tells TCP port or all ports

upon incoming connection \Rightarrow new socket created

3-Way Handshake

Negotiation window size, sequence numbers

Piggybacking of ACKs

last thing gets ACK

solves premature timeout, allows cumulative ACKs

confirmed release

Fast Retransmit (receiver ACK lost)
if 3 ACKs for same data ordered if receiving
unordered

Slow Start (hysteresis window) until less than double congestion window

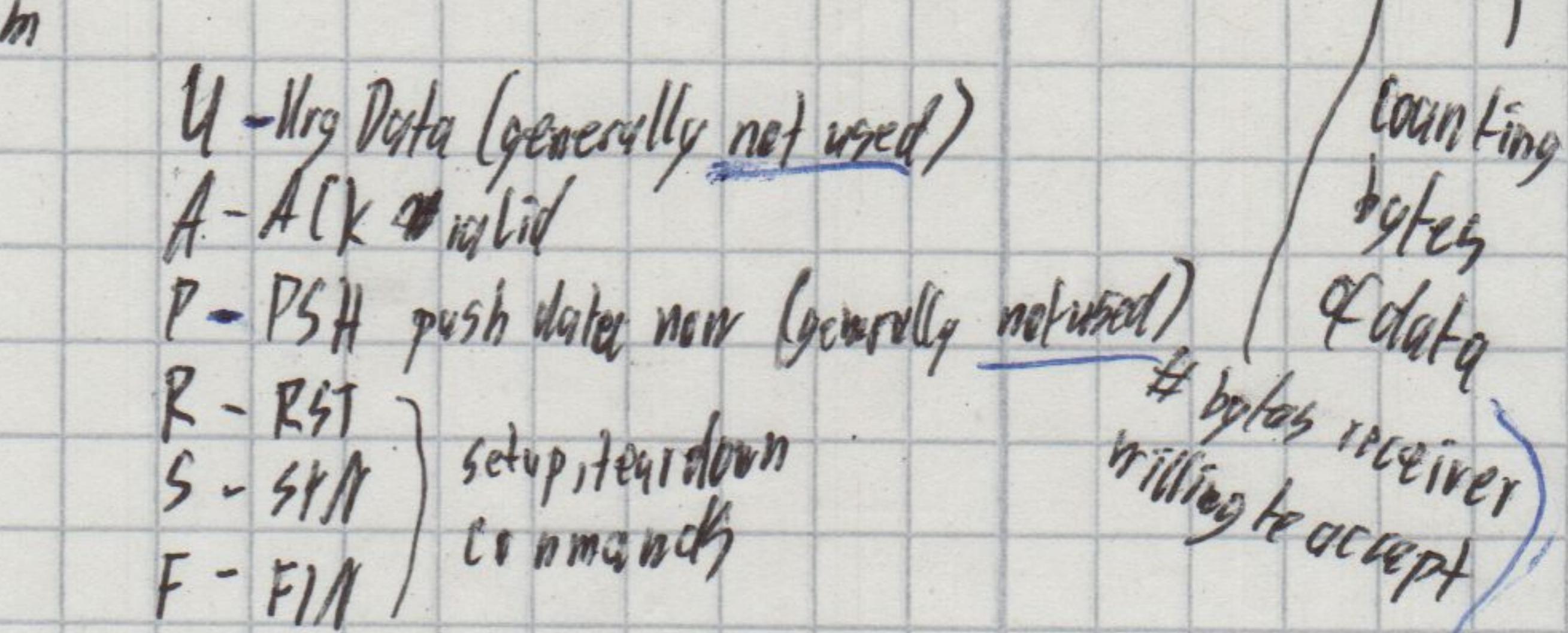
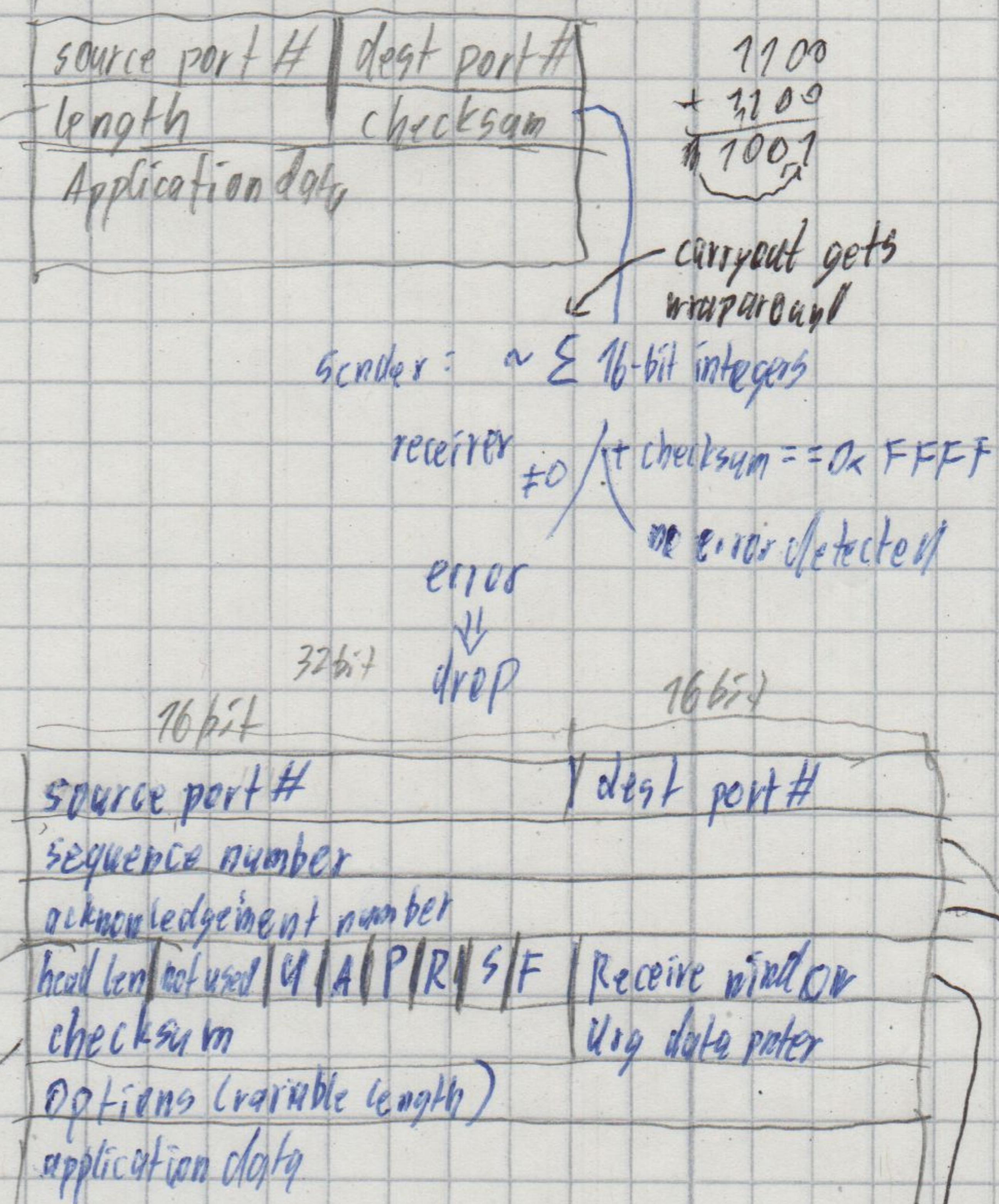
sent fast retransmit
if not triple duplicate ACK

Problem: packet bursts even if many ACKs lost \Rightarrow reset window to 1

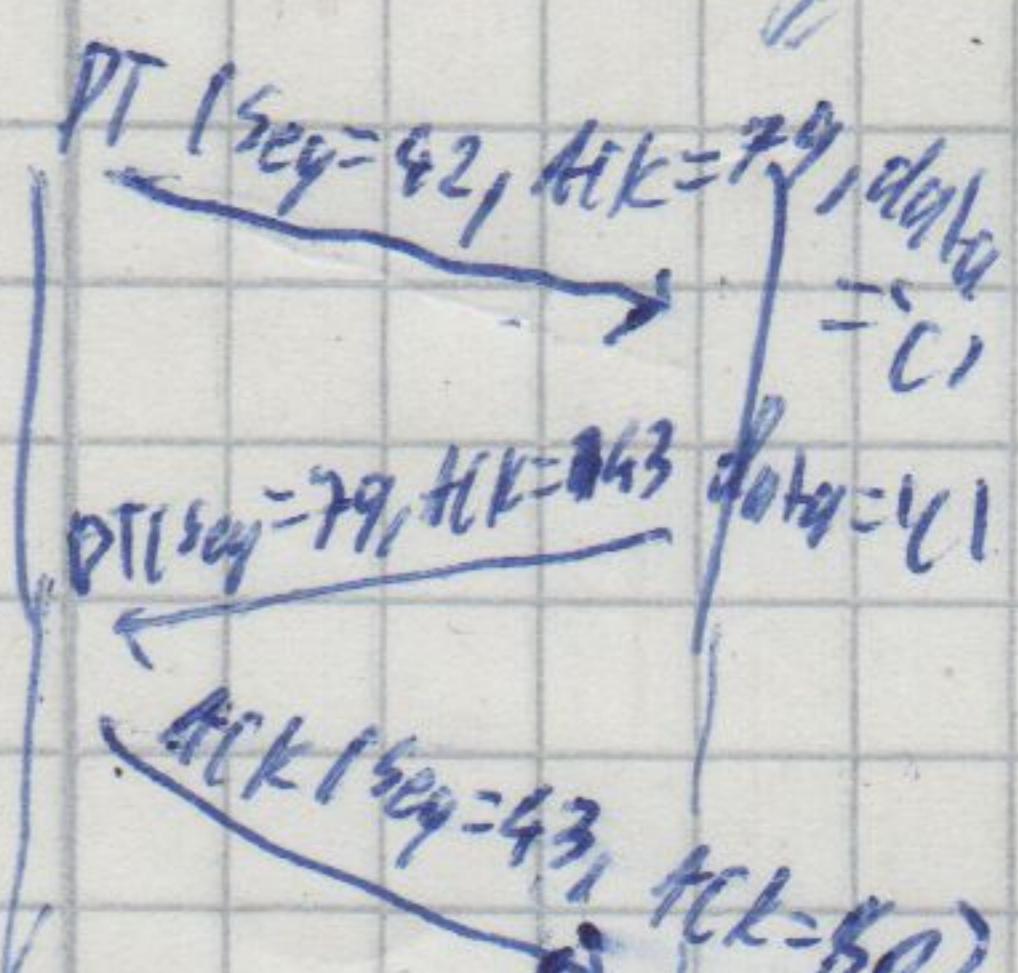
slow start

with previous window as congestion threshold

BRUNNEN



handling of out-of-order packets?
up to implementation



ACK nichts erwartet

timeout: longer than RTT - Round Trip Time
sample RTT from last few ACK

Flow & Congestion Control

receiver can advertise buffer available

if can't send full packet and un-acked data in flight
buffer new data until MSS full \Rightarrow Nagle's algorithm

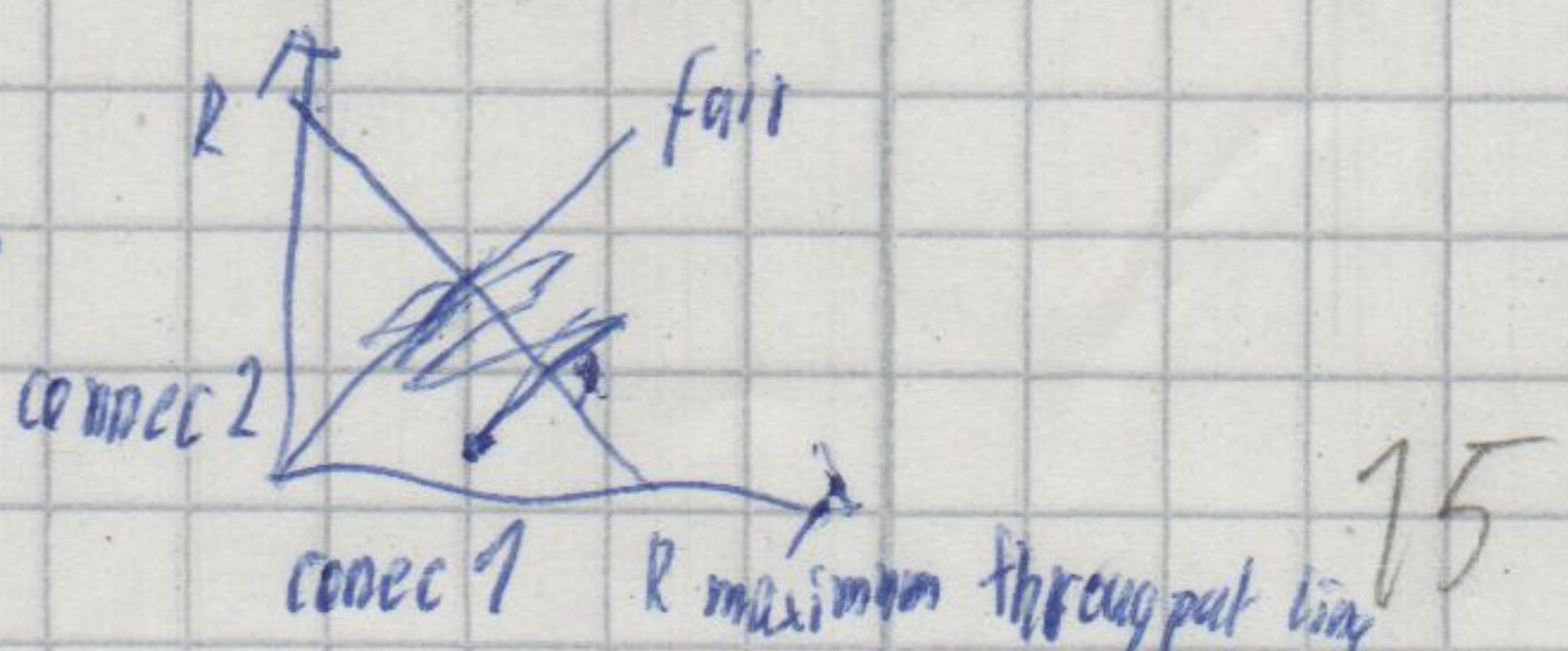
Congestion avoidance phase

self-clocking if implicit feedback from dropped packets
Ack-clocking

congestion window: add multiplicative decrease (MSD)
(additive increase increase)

MSD
additive increase
would need to track RTT so N/A
multiplicative decrease
increment = MSS * (MSS / congestion window)
with each ACK

Sawtooth pattern
+ drives fairness



Internet transport today

mostly TCP but Growth of UDP (QUIC, streaming, Real time Application)

fewer port protocols

Growth of ~~CDN~~ Content Distribution Network

~~SDP~~ SCTP - Stream Control Transmission Protocol (user mode, (kernel mode proposed))

message oriented, Flow & congestion & error control, in-sequence

enabling/disable [receiver: ordering, reliability
sender: fragmentation, Nagle]

Multi-streaming: multiplex message streams into 1 connection

Multi-homing: both multiple IPs for failover

DCCP - Datagram Congestion Control Protocol

unreliable, timely

connection oriented, congestion control, optional ACKs of ACKS

separate header / data checksums

(scheme not fixed, instead congestion control ID)

Multi-homing (mainly for mobile)

will probably die out

Transfer Transfer control

RTP / RTCP real time multimedia protocol ontop of UDP

RTP

- sequence # and timestamps (televis, hardly used)
- encryption
- payload info (audio encoding, ...)

RTCP last sequence received
observed loss rates
jitter
member info
control algorithm

) from various senders

QUIC (~~Quick-HTTP~~ (Quick Internet Connections) - proposed by Google

intervenes with security

faster with known secret

- crypto block = packet boundaries

adjust rate to RTT

forward error correction

and HTTP/2 efficient multiple streams

used

Application HTTP/2

Security TLS

Transport TCP

Network IP

HTTP/2 shim

QUICK

UDP

on UDP

+ firewalls block
unknown protocols

Security threats
require QIC changes
VJ

fast in user-space

Queuing Systems

$$A(f) = P[\text{time between arrivals} \leq f]$$

$$B(x) = P[\text{service time} \leq x]$$

places in queue

servers

disciplines (strategies)

↳ FCFSS - first come first serve

shortest job

deadline

priority

notation

$$\lambda = \frac{\alpha(t)}{t}$$

arrival rate

$$\lambda = \lim_{t \rightarrow \infty} \frac{\alpha(t)}{t}$$

service rate

$$\mu$$

C_n - n^{th} customer

$\alpha(t)$ - # arrivals in $[0, t]$

$S(t)$ - departures

$N(t) = \alpha(t) - S(t)$ - # in system

T_n - arrival time C_n

$t_n = T_n - T_{n-1}$ - interarrival time

x_n - service time C_n

w_n - waiting time in queue

$s_n = w_n + x_n$ - system time

$$A(f) = P[t_n \leq f]$$

$$B(x) = P[x_n \leq x]$$

Metrics:

waiting time

in system

backlog undone work

utilization ρ

für random t_n independent of n

\bar{T} - average interarrival time if independent n

$$\bar{T} = \frac{1}{\lambda}$$

$$\bar{x} = \frac{1}{\mu}$$

$$\bar{s} = \bar{T}$$

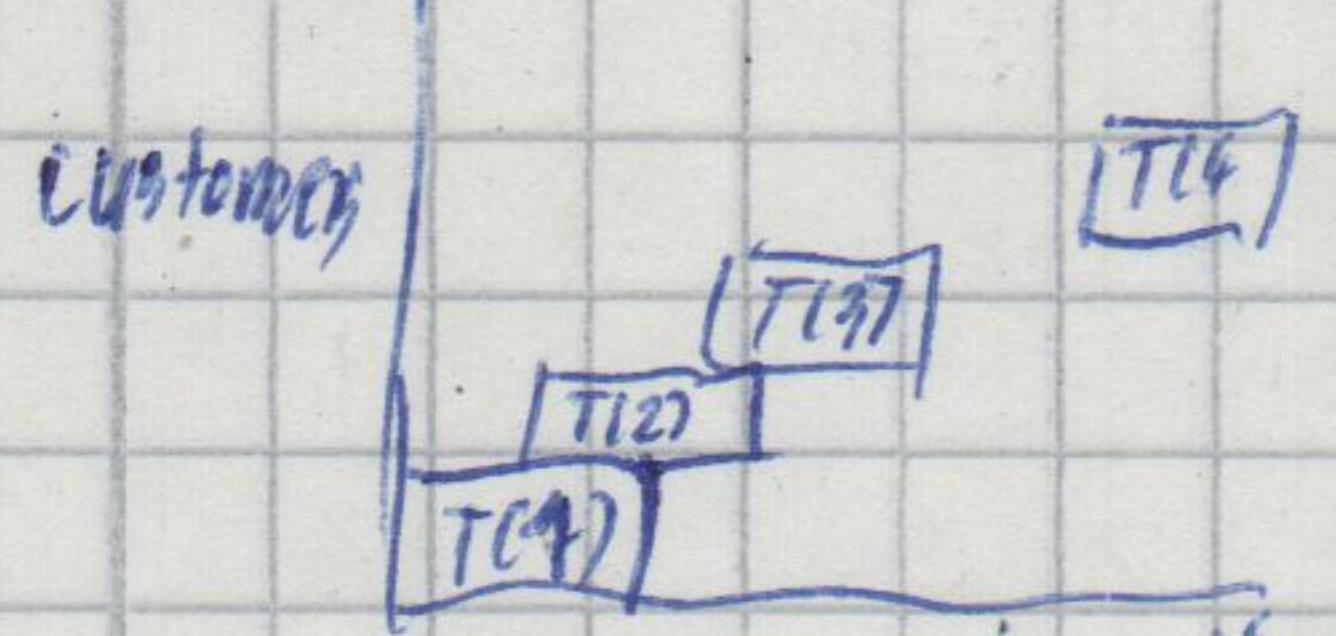
$$\text{average per customer } T_f = \frac{r(f)}{\alpha(f)}$$

total time customers in system

$$r(f) = \sum_{i=1}^{\alpha(f)} T(i) = \int_0^f N(\tau) d\tau$$

$$N_f = \frac{1}{T_f} \int_0^{T_f} \frac{r(\tau)}{\tau} d\tau \text{ average # customer} \Rightarrow N_f = \lambda_f T_f \text{ wenn } \lim_{f \rightarrow \infty} T_f = T \Rightarrow N = \lambda T$$

little's law



Utilization factor ρ ratio jobs enter : jobs done

normally $0 \leq \rho < 1$

$$\text{for single server: } \rho = \frac{\lambda}{\mu} = \lambda \bar{x}$$

Classes:

Stationary process invariant to shifts in time

$$F_x(x, t+T) = F_x(x, t)$$

Independent process (white noise)

Random walk

new Pos = old Pos + random variable

Renewal process

(s) Random walk but interest in counting transitions

Stochastic Process

family of random variables $X(t)$ indexed by time

classifications

State space - $X(t)$ discrete?

discrete state also called chain

Time parameter - discrete: $X(t) = X_n$

Dependence $X(t)$'s [conditional probabilities (dependent on previous state)]

Order of dependence $P[X(t_{n+1}) \leq x_{n+1} | X(t_n) = x_n, \dots, X(t_1) = x_1]$

The Markov Process

discrete state space

first order of dependence $P[X(t_{n+1}) \leq x_{n+1} | X(t_n) = x_n]$

homogeneous transitions independent of observation time

memoryless X_{n+1} independent on time in X_n

exponentially distributed

← Birth-Death-Process

special case Markov transitions only to neighbor states

λ_k Birth rate in population size k

μ_k Death rate

↓

Homogenous

Birth and Deaths independent

keine simultanen
Übergänge für $k \rightarrow k+1$ oder $k \rightarrow k-1$

$$P[1 \text{ birth in } (t, t+\Delta t) | k \text{ in population}] = \lambda_k \Delta t$$

$$+ o(\Delta t)$$

$$\mu_k \Delta t$$

$$+ o(\Delta t)$$

$$= 1 - (\lambda_k + \mu_k) \Delta t + o(\Delta t)$$

$$\text{bei } k=0$$

$$\mu_0 = 0$$

$$P_k(t+\Delta t) = P_k(t) p_{k,k}(\Delta t) + P_{k-1} P_{k-1,k}(\Delta t) + P_{k+1} P_{k+1,k}(\Delta t) + o(\Delta t)$$

$$\lim_{\Delta t \rightarrow 0}$$

$$\frac{dP_k(t)}{dt} = -(\lambda_k + \mu_k) P_k(t) + \lambda_{k-1} P_{k-1}(t) + \mu_{k+1} P_{k+1}(t) \quad k \geq 0$$

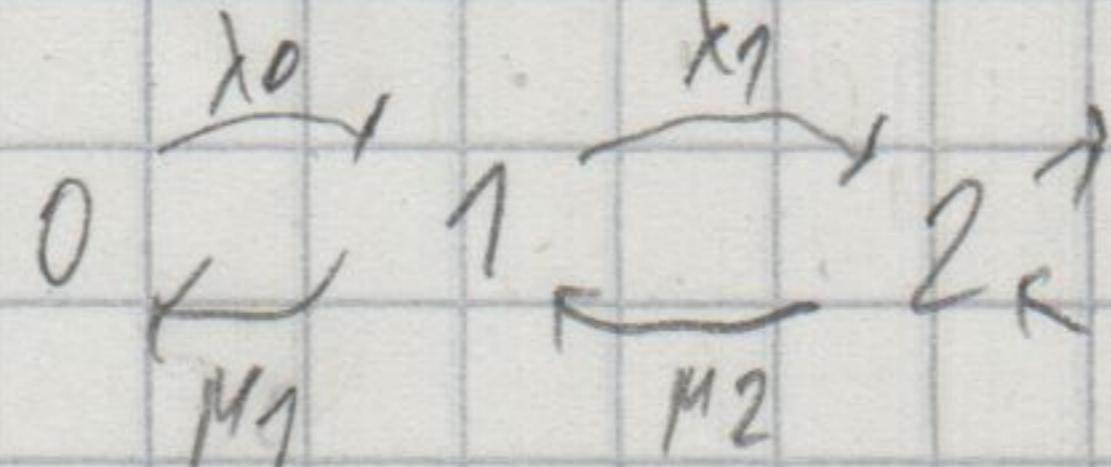
BRUNNEN

Flow rate out

Flow rate in

Falls

Equilibrium



Poisson Process

$$\begin{array}{l} \text{Death rate } \mu_k = 0 \\ \text{Birth rate } \lambda_k = \lambda \end{array} \Rightarrow \frac{dP_k(t)}{dt} = [-\lambda P_k(t) + \lambda P_{k-1}(t)]_{k=1} \Rightarrow P_k(t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}$$

average # arrivals in t $E[k] = \lambda t$

Variance = λt

$$\text{Merging } \lambda' = \sum_{i=0}^n \lambda_i$$

Equilibrium Birth-Death Systems

$$\text{Assumption } \lim_{t \rightarrow \infty} P_k(t) = p_k$$

flow conservation over any closed boundary

$$\text{equilibrium } \lambda_{k+1} p_{k+1}^\top \mu_{k+1} p_{k+1} = (\lambda_k + \mu_k) p_k$$

$$\lambda_{k+1} p_{k+1} = \mu_k p_k$$

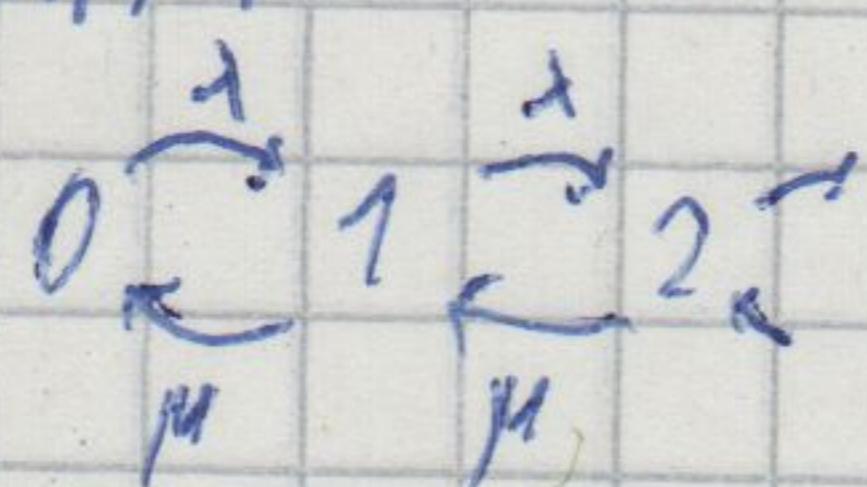
$$\Rightarrow p_k = p_0 \prod_{i=0}^{k-1} \frac{\lambda_i}{\mu_{i+1}} \quad k=0, 1, 2, \dots \quad \text{da } \sum p_k = 1 \Rightarrow p_0 = \frac{1}{1 + \sum_{k=1}^{\infty} \prod_{i=0}^{k-1} \frac{\lambda_i}{\mu_{i+1}}}$$

Queuing Systems

$$\text{Kendall's Notation: } A/S/m/N/K/SD$$

- A : arrival process
- S : service process
- m : number of servers
- N : population size
- K : queue discipline
- SD : usually FCFS

M/M/1

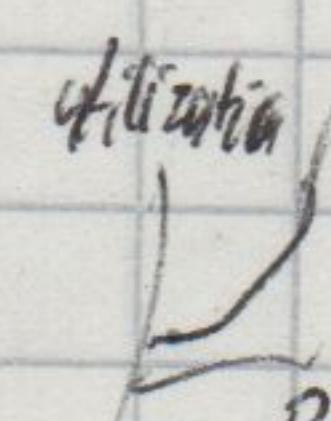


geometrically distributed

$$\text{steady states } p_k = p_0 \prod_{i=0}^{k-1} \frac{\lambda}{\mu} = p_0 \left(\frac{\lambda}{\mu} \right)^k = p_0 p^k = \frac{p}{1-p} p^k$$

$$\text{requires } p = \frac{\lambda}{\mu} < 1 \quad p_0 = 1-p$$

Server idle on average $1-p$ of time

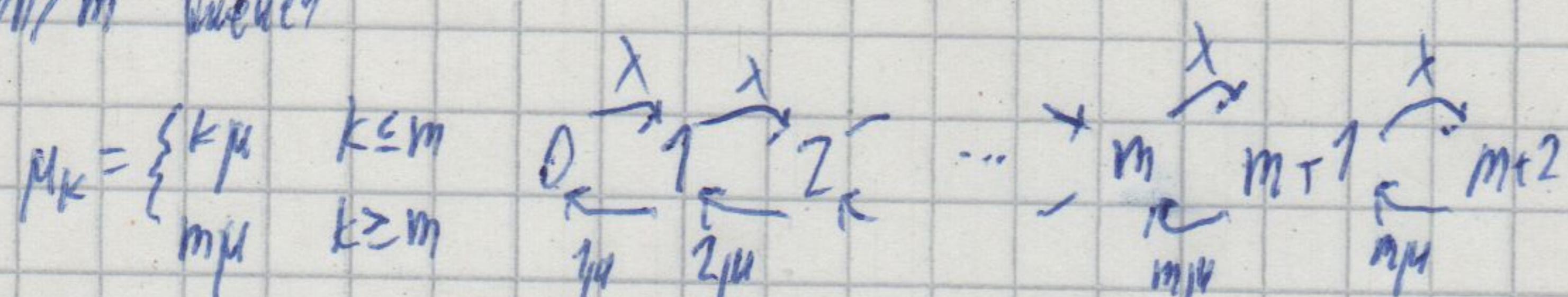


$$\text{average customers in system } \bar{N} = \frac{p}{1-p} = \frac{\lambda}{\mu - \lambda}$$

bei $p=1$ variability in arrivals is the problem

$$\text{average time spent in system } T = \frac{1/\mu}{1-p} = \frac{1}{\mu - \lambda}$$

M/M/m Queues



$$\bar{N} = mp + \frac{pS}{1-p}$$

$$T = \frac{1}{\mu} \left(1 + \frac{S}{m(1-p)} \right)$$

$$p = \frac{\lambda}{m\mu}$$

$$\text{Erlang's formula } S = P \sum \text{queuing J} = p_0 \frac{(mp)^m}{m! (1-p)}$$

M/M/1/N

$$p_k = \begin{cases} \frac{1 - \frac{\lambda}{\mu}}{1 - (\frac{\lambda}{\mu})^{N+1}} \left(\frac{\lambda}{\mu} \right)^k & k \leq N \\ 0 & \text{otherwise} \end{cases}$$

Effective: $N-1$

$$\lambda' = \lambda \sum_{k=0}^{N-1} p_k$$

$$\bar{N} = \frac{p}{1-p} - \frac{(1-\lambda p)^{N+1}}{1-p^{N+1}}$$

M/M/m better than $m \times M/M/1$

apart from travel time

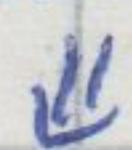
humans automatically adapt

Time

mismatch \rightarrow problems

Difficulties:

Quartz Oscillators sensitive



accurate 1 part per million (1 sec every 2 weeks)



Synchronization

Internet

but asynchronous
best effort

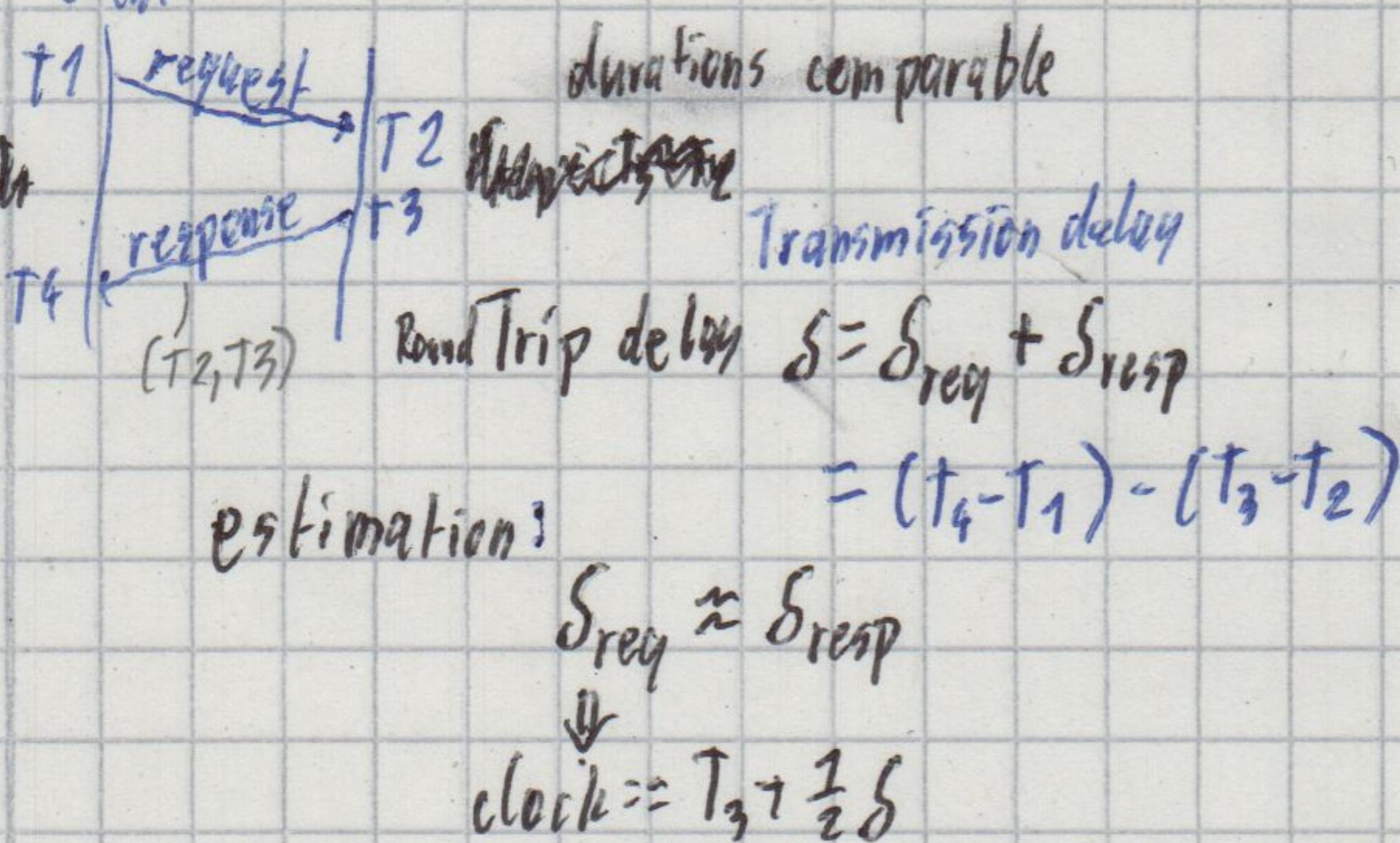
UTC - by atomic clocks radio broadcast

- needs radio (to milliseconds)

or GPS (to microseconds)

Cristian's Algorithm

Client



\Rightarrow At best 100s of micro/milli seconds

standardized NTP - Network Time Protocol

{ error probability small
but never zero }

Logical clocks / Lamport Clock - orders events

$C(a)$ - clock time when event a
rel: $a \rightarrow b \Leftrightarrow C(a) < C(b)$

every Process P local clock C_i
before executing event $C_i := C_i + 1$

Send C_i with every Message m
When receiving $C_j := \max(C_j, C(m)) + 1$

on ties append process ID $C(a).i \leq C(b).j \Rightarrow C(a) < C(b) \vee (C(a) = C(b) \wedge i < j)$

\downarrow
total order

$a \rightarrow b$: a happens before b

$a \parallel b$ concurrent (we can't order them)

order transition

t in ~~same~~ process use physical time
 t different order if direct effect

Totally-Ordered Multicast - client \xrightarrow{m} n Replicas

receive update from client \rightarrow broadcast to all (including self)
from replica \rightarrow add to queue (sorted by Lamport time)
if head broadcast ack

on ack \rightarrow mark ack'd

all ack'd \rightarrow remove & process

send ack for new head

- scalability

- performance of slowest

- no protection against
node fail

lost messages

Replicating Database Operations

write any, read any
+ spreads load
- consistency issues

write any, read all
+ write spreads load
- read requires talking to all nodes

write only to "Primary Node", read any
+ consistency, backups for reading
- write load not spread

In Primary receives update → order / log
replicate log to backups
backups → execute log
ack
Primary if all ack'd → execute log

- fails?
↳ needs logs

Reconfiguration - fresh node if backup fails

Leader election - if primary fails can use Consensus Protocol

Further readings:

80s Paxos protocols for Crash Fault Tolerant (CFT) consensus
00s Zookeeper widely-used CFT meta-data-store
2013 Raft RSM easy to understand

Grundlagen 90s PBFT Practical Byzantine Fault Tolerance
2015 Hotstuff higher throughput

Primary-Backup Replication

replic = state machine, update = transaction
deterministic - same order → same result

All-or-nothing atomicity

One Primary communication with backups

- Reliable Atomic Broadcast

CAP Theorem

only two properties at same time

Consistency
CA

CP

Availability AP

Partition
Tolerance

Hardware

Bottlenecks

Data Movement (Bandwidth) ← modern NICs mostly solve this
Coordination (Latency)

Entwicklung

faster links
but CPU limits scalability needed data
gets more

more logical cores
1990-2000 CPU Frequency Scaling
2000-2015 commodity in cloud
2015- Specialized Hardware
GPU, TPU, FPGAs
NICs, Switches
Smart Drives
Motherboards with spec. Chips
more flexible

CPU → FPGAs → ASIC
more efficient

z.B. PISA - Protocol Independent Switch Architecture
programmed with PG

Programmable Parser → Programmable Match-Action → Prepr. Pipeline → Deparser

In-network with Host CPU
Network-attached without

+ Free choice of architecture
+ fine-grained pipelining, distributed memory, communication
- code occupies space
- only in 100s MHz range
- code → synthesized → placed & routed
map onto specific FPGA

Blocks:
I/O
CLB - Configurable Logic Block
Embedded Memory
DSP

Coordination bottleneck

z.B. Zookeeper broadcast, ack, commit Leader n followers parallelism? can only pipeline
TCP

network access + traverses layers → overhead

operation on network level - Net Chain with programmable switches at line rate

consensus

small dataset in switch

Replication by manipulating ethernet packets

+ low nearly constant latency

- not end-to-end switch becomes node

- small memory

- hard to program & integrate

Alternative: Specialize part of server no change in infrastructure, protocols

FPGA on host with special optimized TCP-stack

Pipeline: Network → Consensus → Application

Multicast Routing

Flooding

- + no routing tables
- bandwidth
- state to remember known packets

Spanning Trees (similar to Minimum ST problem)

Group-shared trees

Steiner Trees - MST

- * NP-hard - if fo about entire network
- + heuristics - Monolithic: rerun w/o change

Core Based Trees

+ one rooter - less optimal tree - bottlenecks

+ less complex near core

on join msg to core where it hits tree → new branch

Source-based trees

Shortest Path tree - through Dijkstra's algorithm - may profit from link state routing in fls
tree of shortest path to all receivers

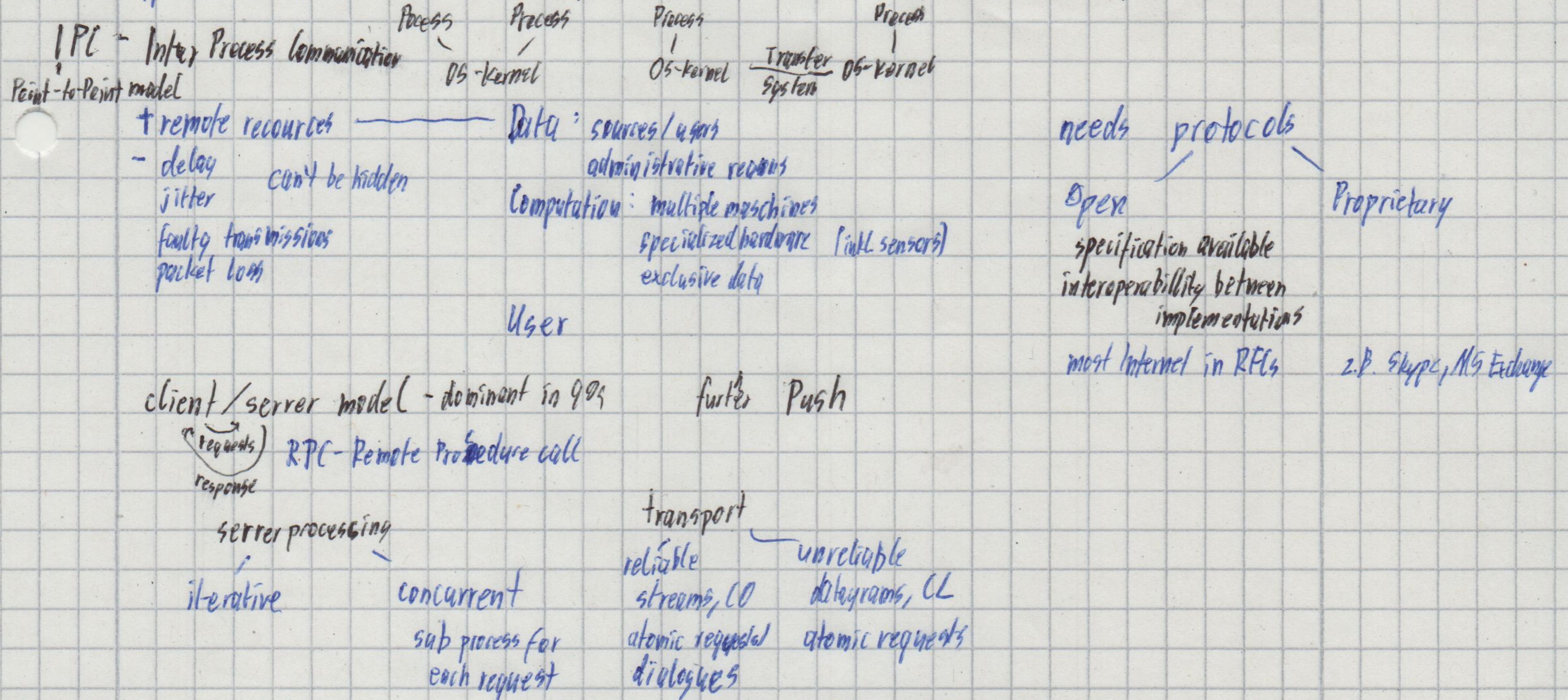
Reverse Path Forwarding , relies on unicast routing table (or own one)

flood if multicast datagram from shortest path to sender may be bad with asymmetric links

optimization

"grafting": cut of subtrees through soft-state "Prune" messages

Application Layer



needs protocols

Open

specification available

interoperability between implementations

most Internet in RFCs

z.B. Skype, MS Exchange

DNS - Domain Name System

hierarchic distributed database IP → Name

Resource Record:

name	TTL	class	type	rdLength	rdata
IN-			A		delegation
internet			AAAA		iterative
			NS		query
			MX		recursive
			CNAME		

root . com org google

zone data in Resource Records

13 root servers but request → 1600 instances
Redundancy for availability

WEB / HTTP

before Web: 1970s
HTTP stateless
<http://www.example.de/path/info.html>

host name path

1970s 1980s

FTP Emails

1980s

Gopher - for linking documents

base html file

references objects

1989/90 at CERN

Web
HTTP 0.9 documented 1991
HTML
Web server
browser

HTTP stateless but TCP holds state
client/server

traditionally client initiates
TCP mit port server port 80

v1.0, v1.1

nonpersistent
overhead
RTT dominates

v.1.1

persistent
multiple objects
over a single TCP connection

9.1 0.9 HTML only

9.4 HTTP/1.1 TLS

9.6 1.0 GET, POST, multimedia

9.7 HTTP/1.1 multiplexed stream via 1 TCP

15 1.1 virtual hosting security

22 HTTP/1.1 SSL in UDP

IETF defines protocols z.B. HTTP

W3C defines Web standards z.B. HTML

Companies create own protocols

Request: ASCII

GET <path> HTTP/1.1
Host: <host name>
User-Agent: z.B. Mozilla/4.0
Connection: close
Accept-Language: fr
+ leerzeile indicates end

status code phrase
with without pipelining
reset - efficient
→ superseded by
multiplexing HTTP/2

Response: HTTP/1.1 200 OK
connection close 301 Moved Permanently
Date: 400 Bad Request
Server: 404 Not Found
Last-Modified:
Content-Length:
Content-Type: text/html
data: data --

HTTP/2
+ speed
messages binary
Multiplexing in TCP
Push
stream priority

HTTP/3

QUIC

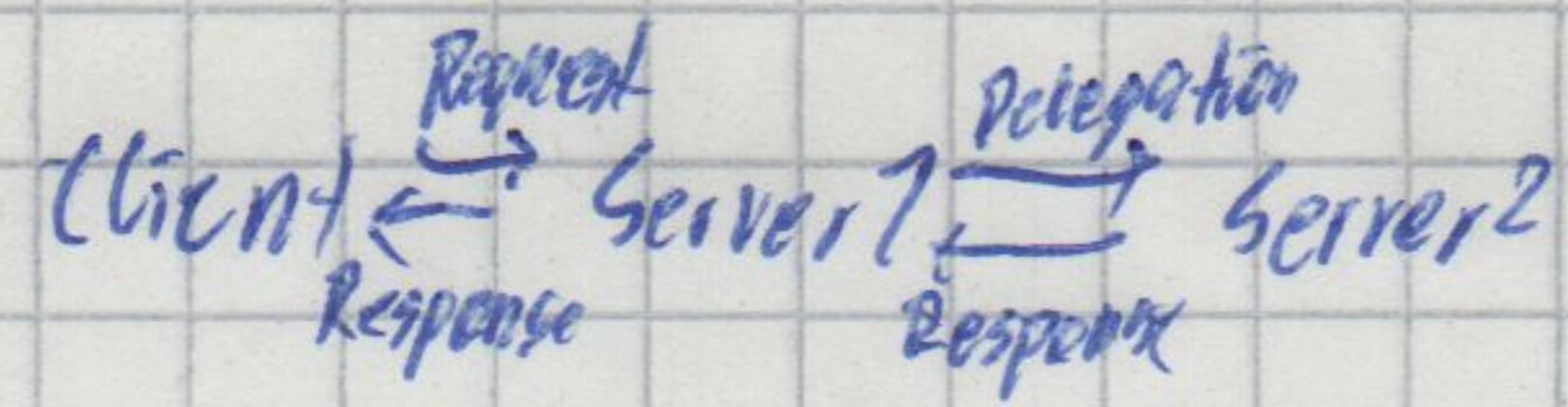
O-RTT (TLS handshake abbreviated)
always uses TLS

User-Server State - cookies
components
header line in request/response
file manager by browser
Back-end database

examples
authorization
shopping cart
recommendations

user state z.B. Email
Tracking

Extended Client/Server Model
↳ delegate sub-tasks
message chains



Peer to Peer - Buzzword, flat anarchistic = Decentralized distributed system

Clients act as servers \Rightarrow So still one requesting, one responding at an instance

no structure) can be introduced algorithmically
no identifiers
only names

traditionally File Sharing all peers server \rightarrow highly scalable

Applications
Blockchain
Video Streaming
Dynamic load balancing
Routing

Napster Centralized Directory Server

~~public domain protocol~~
~~many implementations~~
~~overlay network of TCP connections~~
~~usually <10 neighbors~~

Gnutella Query Flooding

fully distributed
many implementations
overlay network of TCP connections
usually <10 neighbors

flood query

for scalability: limited scope flooding

Query Hit over reverse path

only n hops