distribution system · L.Network + closely coupled

Information → abstract data → physical signal conventions (length, delay)
propagation delay/speed (copper 2/3 c) data error rate
simplex / TDD-Time Division (time slots synchronized / on-demand (canceled logic))
half duplex / 2 cables / fast enough multiplex
full duplex CN: V.V.-scalability ⇒ structure end ↔ routers
circuit switching + simple no queuing − R. dedicated path: needed flow − setup in pause
packet switching: chop into packets with administrative info
store and forward; Multiplexing needs Demultiplexing
downward mean imitate / Dll-division Mux mainly wireless
up: Mux over 1 time, frequency, code, space
orthogonal Spreizcodes · L wavelength (Optic)
Frequency Spectrum aufgeteilt
Broadcast medium needs exclusive access
Mux v MAC (central only speak on asked z.B. backoff)
Layer Architecture − overhead / + exchangeability simplicity
Protocol format, order messages ≈ ent.
action rules m. sequence chart
packet unit of transportation
datagram CL send packet
frame with find envelope
cell small packet fixed size
(N)-PDU = (N)PCI + (N)-SDU
(N)SDU = (N+1)-PDU

1. Physical Non secure signal stream adjacent systems bit activation lines P: USB
2. Data Link Error recovering stream frames adjacent sys. sequence no. boundary detection checksum LAN 2 Sublayers: Logic Link Control − error secure; Medium Access Control undeferred P: Ethernet
3. Network packet stream end systems routing, mux error corr. Flow/Congestion Control
4. Transport logic con. processes mux, Packet assembly, adress handling error flow congestion fragmentation hide details P: TCP shorter + pipelin.
5. Session manage establish/release connection; Tokens, checkpoints P: PTT telematic services
6. Presentation harmonize data representation P: Abstract Syntax Notation ASN.1 (structs)
7. Application Service elements for common tasks P: Commitment, Concurrency, Recovery, file transfer P: FTP
OSI 7 Internet 5 (6+7, 3-4)
CO − telephony + better service − RTT overhead; Flow C. reliability
phases: establishment, data transfer, teardown connection: distributed state
CL − postal + stateless scalability − no reliability
Info → Algorithm → Routing table (fills updates / uses) control plane; control plane
⇒ Forwarding data plane
CONS: Routing at setup, forward circuit ID
CLNS: periodically
Goals: Optimality, Correctness, Simplicity, Robustness, Stability, Fairness
Metrics: average delay, individual throughput; practice hop count
Algorithms
1. Non-adaptive: don't base on current state
1.1. Flooding: Excessive load; sequence and spr.
reduce: sequence no., TTL (1/4)
+ multicast to many; dyn. database, crash, mobile
+ fast changing networks
1.2 static · pre configure all per dest. + predictable − network/load change
Static entries · over provisioning
2. adaptive use network state
traffic adaption only with care, changes fast, overhead
2.1. centralized periodically routers → (forward link + state) RCC − Router Control Center (computes, dispatches)
− Aging − bottleneck (drop by router) scalability, vulnerability, traffic
2.2 isolated · 2 Hot potato: Port with shortest queue − efficient + inefficient
· 2 Backward Learning: starts random, source address and # hop in header
keep soft state with min # hop from source of link
2.3 distributed graph path + link cost
static ↔ dynamic: periodic update / respond to events, changes
router changes view
Global ↔ decentralized − all their links, iterative comp. exchange
Sell routers full topology
2.3.1 DVR − Distance Vector R. (distributed, self-terminating, wait, recompute, notify)
$D^x(Y,Z) = \text{dis}(X \overset{\text{via}}{\rightarrow} Y) \Rightarrow$
$= c(X,Z) + \min_w \{D^Z(Y,w)\}$
+ good news − errors propagate: travel fast − bad news slow (problem: count to ∞)
(Split Horizon with) Poisoned Reverse
if $D^Z(X,Y)$ min → Z tells Y ∞ as dist to X
no fix for ≥3 · Split Horizon · notify update
don't advertise to neighbor that sent
2.3.2 LSR − Link State routing (global)
Broadcast LSP Packets, then router compute
− O(nE) send − n × O(n²) algorithm
Hierarchical Routing to solve scalability
Administrative autonomy of intern routing
AS − Autonomous System ~60.000; Gateway Router
Stub Lunique ID, need know path to
− Multihomed (corporations)
− Transit diff relations equals ↔ provider/customers
Finden wo gateway, intra AS routing system

RIP z.B. Routing Information Protokoll
1983, metric # hops, distance vektor; Poison reverse
Open Shortest Path First Link State, publicly available
routing info through TCP, integrated multicast, best effort, metric
Interior Gateway Routing Protocol · realtime, cisco proprietary
inter-AS policy dominated; often hot potato = min intra cost
also needs to propagate info reachability to router in AS
BGP − Border Gateway Protocol · Path vector + allows loop avoidance · de-facto standard
Peers have session (semi permanent TCP)
not 1:1 links, eBGP − between AS's, iBGP between
advertising = Promise to forward routers of AS session
route = Prefix + attributes − AS Path, − NEXT Hop gateway router
accept decline after import policy; z.B. dual homed may don't forward between providers
Potential Problems
Route sub-optimal, instability, oscillation, flapping
Security: blackholing, redirection
Mobile Routing Dynamic Source Routing DSR IETF RFC 4728 designed for sensor nodes
step towards realtime routing; sender determines path
discovery if needed an unknown: flooding with id, hops add their address
dest sends it back (→ bidirectional ⇒ flooding)
maintainance − delete unused errors − detection − layer 2 ack (listen if station forwards; only directed antennas) − Carrier Sense, explicit ack
inform sender, look up new path ↩ (may be globally suboptimal)
Overlay Routing z.B. P2P, shared memory
virtual network topology over real network
Stretch factor = #hops Overlay / #Underlay (of shortest path) $= \frac{N_{sh}^{o}}{N_{sh}^{u}}$
metric: z.B. stretch, #overlay hops
existing (properly build ⇒ fast rules)
ICMP − Error reporting, router signaling
IP − addressing conventions, datagram format, Packet handling
IP header 4 version, 4 head length (32 bit words), 8 type of service (rarely used)
16 Length (total in bytes) · same for fragments
for Fragmentation = 16 identifier, 4 Flags, 8 offset (original pos in 8 octets)
8 TTL (in hops), 8 upper layer protocol, Don't frag, More Frag (all but last 1)
16 internet checksum (header) · ones' complement of ones' comp. sum with carry added · sum of header 16 bit words
data · 32 source, 32 dest., options if any,
Namespace: flat − conflicts, inflexible, central; hierarchical Partitioning + delegate + decentralized, recursion
IP classfull
A: 0 7 Net 24 Host · network host 0 · loopback
B: 10 14 Net, 16 Host · broadcast 1 · 127.x
C: 110 21 Net, 8 Host · private: 10/8
D: 1110 Multicast · 172.16/12 · 192.168/16
E: 1111 Reserved · Link local: 169.254.0.0/16
Prefix: IP & Net Mask ⇒ IP if 1 in Mask
one IP per host and router interface ⇒ con between router and subnet
Subnet: device interfaces with same subnet Prefix can work without router z.B. organisation, departments
class A, B can have 2, but 1 subnet (+ more hierarchy)
⇒ variable length subnet mask VLSM
classless interdomain routing CIDR
discard class boundaries, aggregate when advertising if ISP assigns continuous block
Longest match routing
Link with longest matching Prefix (mask)
Internet Corporation for assigned Names and Numbers
L non profit, LA, carries Internet assign. Numbers Authority function
allocate Adresses, DNS, −
↳ ISP → Organisation → Host · static: hard coded / reverse ARP from server
Dynamic Host Config. P. + plug and play
L IP, default gateway
IPv4 shortage
Network Address Translation
+ only one extern IP address, intern use local
router replaces IP and Port, needs NAT translation table
only 16 bit port ⇒ 65 k · private
+ intern IP independent · simultan connections
− violates e2e principle − OSI was layer 4
+ security, device not explicit addressable − no server − IPv6 better
IP ⇒ MAC: Address Resolution Protocol + plug & play − ARP cache poisoning
1. broadcast IP 2. unicast back 3. cache (soft state)
IPv6 128 bit addresses, same routing switching principals
new Addressing, Protocols · get rid of NAT
+ traffic class, flow label may better Quality of Service
+ flexible hierarchy levels + serverless plug and play
+ authentication, encryption + triangle mobile routing
header: 4 version, 8 traffic class (like type of service in IPv4)
20 Flow Label, 16 payload length, 8 next header, 8 hop limit
128 source, dest. · TCP, UDP or extension header
addresses written as 8 Hex mit : getrennt, keine leading zeros, one :: für 0 bis 128 bit
global Unicast: top level aggregation IANA
001, 13 TLA ID, 8 Reserved (for extending neighbor fields)
24 NLA ID, 16 SLA ID, 64 interface ID
Next level, site level · may base on MAC (FFFE inserted, 7th bit inverted, but privacy concern)
Transition from IPv4
Dual stacking, interoperative
tunneling: IPv6 packet carried as IPv4 payload
IPv6 router usually can IPv4
Other effects: NDP or ND ersetzt ARP (uses ICMP Neighbor Discovery P.)
Link local mandatory: FE80::/10, Prefix MAC
SLAAC but DHCPv6 exists · DAD, allow + IP multicast
Stateless Address Autoconfiguration
router announces /64 network (RA) · interface ID explained above
remaining via EUI-64 / SHA1(NTP-time | MAC)
DNS: AAAA records
ICMPv6 Errors: dest. unreachable, timeout, Packet too big, Parameter Problem
Informative: Echo request/reply, NDP, discovery, multicast Listener
NDP messages: router/neighbor advertisment/solicitation
usage: discovery, router, prefixes, net param, Autoconfig, addr. resolution
duplicate addr. detection, neighbor unreachable detect.

Layer communication between processes
Segmentation if N-SDU > N-PDU, Service Access Point, Connection End Point
Service Primitives Layer − name.type(param)
L Name: Connect, Disconnect, Data, ...
L Type: request, indication, response, confirm, ...
L Parameters: z.B. header fields, payload
n tiny SDUs → 1 PDU · blocking/unblocking · ensure boundary detect.
Addressing OSI: Local CEP (socket), but how to initially?
→ OS Process ID? but arbitrary, changes on restart, only 1 per process
⇒ abstract service name · well known ports 1-1023
Internet: Ports Protokol · L 0 für arbitrary port
Processes can register them · echo 7/udp · ftp 20/tcp data, 21 control · https 443
OS queues packets for them · time 37 · name 42 · ssh 22
socket = (IP, port) · dns 53/udp, tcp · smtp 25 · http 80
socket-pair (local, remote socket) · diff to OSI ports
L only pair fully qualifies connection ⇒ OSI violation · port only unique with IP! · what if IP replaced
receive IP datagram → TCP inspects socket pair
⇒ Types: CO, CL
Connection Control: 1. Idle, 2. Establishing out going, Establish incoming, Connected (D.ind, DReq, CR, T-Data)
T-connect.Request (dest addr., source addr.) · .ind (dest, source addr.) · .response (resp addr.) · .confirm (resp. addr.) CR / CC · responding
may be addr., afterwards CEP
T-Disconnect .req (userdata) .ind (cause, userdata) · only at ind, lack of resources, unknown, requested, below minimal quality, provider error, user unreachable · lauch zum Connect ablehnen
Errors Establishment
Loss of CR ⇒ timeout resend
duplication ⇒ wrong timeout · application shouldn't be involved
other Error Loss CC (Initialisierung verwirrt über Data)
⇒ 3WHS
Problem duplications / replay attacks
↳ sequence numbers (reuse only after packet lifetime)
in bbps connections large timestamp in addition
CR · CC (may be ack) · Ack or DT
critical examples: CR(seq=x) · Ack(seq=y, Ack=x) · Data(seq=x Ack=y) · CR(seq=x) · Ack(seq=y, Ack=x) · Reject(Ack=y) · CR(seq=x) · Ack(seq=y ack=x) · DATA(seq=z, ack=z) · Reject(Ack=y)
handled correctly y ≠ z
Connection Release
Graceful (explicit) − mit DR
Abrupt (implicit) · Teardown of Network layer con. · risks loss
good Praxis wait for all acks
unsolvable ⇒ two army Problem
Praxis 3WHS · lost 2. DR · lost ACK · lost retrans.
DR → DR ← ACK · solved by retransmission · timeout · N timeouts ⇒ release
solved: timeout by optimistic

Congestion Control
bottleneck in network?
Scenario 1: 2 sender, receiver, ∞ buffer, no retransmit (λ_{in}, λ_{out}, delay, C/2)
Scenario 2 finite buffers $\lambda'_{in} = \lambda_{in}$ + retransmitted
goodput $\lambda_{in} = \lambda_{out}$ · Perfect: only lost retransmitted
congestion ⇒ more work for given goodput
only send when room in router · retransmit only lost · retransmit after time out
(graph: delivered vs packets sent: perfect, max desired, congested)
scenario 3 4 sender multihop paths, timeout · upstream capacity for dropped wasted
L snowball effects ⇒ Congestion Control · Global issue · if drop − design wrong
Goals: most data, best delay, avoid collapse, fairness
Detection − open loop no runtime corrections − at source Admission control (check before con Establish) − at dest. ACK
closed loop feedback allows sender to adapt
explicit feedback: point of congestion informs sender
implicit: sender deduces from network behaviour
Actions increase capacity usually only longterm
reservations & admission control don't allow additional traffic
reduce load without terminating on-going · usually requires closed loop
Feedback about state rare, usually circuit switched or similar
Taxonomy router vs. host centric (Internet except dropping packets)
window vs. rate vs. credit based
Router Actions − dropping packets · when buffer full · which? − drop-tail old more important z.B. for Go-Back-N − old new for multimedia more important; maybe even drop from same flow but overhead
Feedback
implizit assumes loss only from congestion · reduces rate · L wrong for wireless (securing L2 helps)
proactive (explicit) warning state if exceeding lower threshold
choke packets − delay − more packets
warning bits router sets dest copys in ack
Random Early Detection (implicit) dropping probability ~ Congestion (P(drop), MaxP, minTH, maxTH, Avg Length queue)
User Datagram Protokoll RFC 768
CL, best effort · reliability can be build Application specific in L7
+ multimedia · loss tolerant · rate sensitive · wichtigsten ≤ 16 bit integers · check + sum == 0xFFFF
Demultiplexing: sockets source port = return addr.
socket identified by dest. · header 16 source 16 dest port, 16 length, 16 checksum · Application layer
Transmission Control Protocol Point to Point (no multicast) pipelined
reliable in-order byte stream, Full Duplex (MSS − maximum segment size)
send & receiver buffer (old TCP used Go-Back-N) · counting bytes of data
header 16 source, 16 dest port, 32 sequence number, 32 ack number
5 head length (in 32 bit words), 5 Reserved, 6 Flags (Urgent Pointer, ACK valid, Push (causes sender to push all unsent data), Reset (the connection)
SYN (sequence), FIN (end of data)), 16 window (# bytes receiver willing to accept), 16 checksum
16 Urg data pointer (offset last urgent data byte), Options mainly Maximum Segment Size, app data
Connection Establishment · Piggyback ACKs · Cumulative
(active mode (connect) · passive " (listen/accept)) 3WHS
< incomming ⇒ new socket
SYN seq=17 · SYN seq=39 Ack=18 · seq=18, ACK=40
Data handling out-of-order packets up to implementation
Ack nächsten erwarteten byte
timeouts > RTT sample from last few acks
Flow & Congestion Control · buffer, rate, MSS
Nagle's Algorithm
1 full paket ∧ un-acked data in flight · gegen overload durch kleine packets
confirmed release FIN ACK → ← close traffic FIN ← ACK → client timewait state if ACK lost
Self lock-clocking: implicit feedback from (missing) acks
AIMD − additive increase − would need to track RTT so Näherung: Increment = MSS · MSS / CongWindow (per ack)
(avoidance phase) multiplicative decrease ⇒ Sawtooth pattern + drives to fairness
Slow start = multiplicative increase until Threshold · start by MSS initially and after timeout
to avoid burst after cumulative ACK
Fast retransmit bei triple duplicate ACK · Threshold = CongW = CongW/2 · CongW = 1
Flow Control protect slower receiver
on T-layer: PDU sizes differ widely
processes need time to answer, parallel connections
↳ needs buffer · can't assume receiver fast enough enough buffer
rate-based: source/node get assigned average rate
window-based: rate depends on window size and rate of feedback · better for high-speed
Alternating-Bit-Protokoll assumes free buffer
Only ack if processed, only send if last acked · after receiving · needs only Bit 2 numbers
overhead (Timeout) late reaction
+ reliable $T_{prop} = \frac{l}{v}$ $T_{trans} = \frac{size}{capacity}$
− stop & go ⇒ slow · number of packets, transfer time
assumes T_{trans} packet = ack · $= 2 \cdot n \cdot (T_{trans} + T_{prop})$
if only one ack: $T_{prop} + n T_{trans} + T_{prop} + T_{trans}$
Stop & continue messages z.B. XON/XOFF Protokoll
→ arrives stop before overflow?
rate based: receiver indicates max rate · + low overhead
− delay can mess rate up − no conf or error handling
credit based: sender only with credit; receiver indicates max packets
absolut credit − might get received at diff time
credit/window credit in relation to acked packets
sliding · ack before able to process more
problem if ack from process ⇒ combine window
retained, otherwise timeouts, resends · size in Permits
indicates − Ack (ack: stop timeout; flow: advance; delete copy)
Error Control
reliable transfer
Detection: Wrong checksum ⇒ drop packet
übernächste seq · error could be in header
L NACK? But what's in routers?
⇒ timeout @sender ⇒ resend only · reliable scheme
ARQ − Automatic Repeat reQuest
type 1: Stop and wait (siehe Alternating Bit)
type 2: Go-Back-N consider current and consecutive lost
Selective repeat only retransmit lost

today mostly TCP but Growth UDP (QUIC, streaming, Real time Applications)
fewer ports and protocols, Growth of Content Distribution network
stream Control Transmission Protocol (user mode, kernel proposal)
message oriented, Flow, congestion, error Control, in-sequence · multiplex message streams
enabling/disable − receiver: Ordering reliability · Multi streaming: into 1 connection
L sender: fragmentation, Nagle · Multi homing: both endpoints: multiple IPs for failure
Datagram Congestion Control Protokol unrealible, timely (CO) congestion control
optional ACKs of ACKs, seperate header/data checksums, congestion control scheme not fixed, instead CCID
Multi-homing (mainly for mobile) will probably die out
Real-time Transfer Protokol on top of UDP, scaleable for many senders, receivers · RTP mixer
+ sequence # and timestamps (to sync, hardly used), encryption, payload info (audio encoding)
RT Control P lost seq received, observed loss rates, jitter, member info, control algorithm
QUIC − Quick UDP Internet Connections − proposed by Google
interwines with Security and HTTP/2 · adjust ratio to RTT, forward error correction
faster with known secret · efficient multiple streams
crypto block = packet boundaries
on UDP + probably not firewall blocked

Appl	HTTP/2	HTTP/2 slim
Security	TLS	Quik
Transport	TCP	UDP
Network	IP	

⇒ sec threats require Quik change
⇒ fast in user space

Time mismatch → problems temperature, radiation
Difficulties Quartz oszillators sensitive
accurate to 1 part per million (1 sec every 2 weeks)
⇒ Synchronization UTC by atomic clocks
Internet-variant best effort
Christians Algorithm standarized Network Time Protocol
1 request T1, response T2, T3, T4
estimate $\delta_{resp} \approx \delta_{req}$ $= (t_4 - T_1) - (T_3 - T_2)$
clock := $T_3 + \frac{1}{2}\delta$ At best 100 of many milli sec
error probability small but never 0

Logical/Lamport Clock - orders events
$a \to b$ = a before b, all $\Rightarrow$ concurrent, order transitive
$C(a) <$ clock time event a
$a \to b \Rightarrow C(a) < C(b)$
in same process: physical time, different: order if direkt effekt
every Process P local clock C_i:
before executing event C_i: $C_i = C_i + 1$
send C_i with every Message m
when receiving $C_j := \max(C_j, C(m)) + 1$
ties append Process ID $C(a)..i$ $< C(b)..j$ $\Leftrightarrow a < b$
totally-Ordered Multicast client $\xrightarrow{m}$ n Replicas
receive update from client → broadcast (including self)
from replica → add request serial queue broadcast ack, if head
all ack'd → mark ack'd
all ack'd → remove & process, send ack for new head
-Scalability - slowest is bottle neck - node fail? test m ?

Replicating Database Operations
write any, read any + speeds load - consistency
write only to Primary, read all + consistency - write not speed
Primary-Backup Replication replica: state machine update: trans mission
primary receives update → order / log
backups → execute log, ack
primary if all ack'd → execute log
- fails? deterministic, All-or-nothing atomicity
Recent-iguration fresh node needs logs backup
Leader Election Consensus Protokoll primary fails

CAP Theorem C Consistency A Availability P Partition Tolerance
only 2 properties at same time AP CP
90s Paxos Protocols: Crash Fault Tolerance (CFT) consensus
00s Zookeeper: wildely used CFT meta-data-store
2013 Raft: CFT RSM solution, easy to understand
Broadlayer: 90s PBFT Practical Byzantine Fault Tolerance
2018 Hot Stuff: higher Throughput

Hardware data grows: bottlenecks
Entwicklung: faster links (NICs), CPU limits scalability
1990-2000 CPU frequency scaling
~2015 Commodity in Cloud (CPU, GPU, TPU, FPGA, NICs, Switches, Smart Devices, Motherboards w. special chips)
now Specialized Hardware
more logical cores
flexible CPU ×20 FPGA ×20 ASIC efficient
Application specific integrated circuit
programmed with P4
e.g. Protocol Independent Switch Architecture
Parser → Match-Action Pipeline → Deparser
Re-programmable Specialized Hardware
Field Programmable Gate Array
+ Free choice of Architecture
+ Fine-grained pipelining
- code occupies space
- only ~ 100s MHz
Code → Synthesized Circuit → Placed & Routed
Coordination Bottleneck dependent tasks
network access traversing layers & overhead
⇒ 1. Operation on network level z.B. Net Chain
Consensus with programmable switches at line rate
small dataset in switch, Replication by manipulating packets
+ low nearly constant latency - small memory
- hard to program & integrate - not e2e
2. remove CPU from critical path z.B. DARE
Remote Direct Memory Access of NIC
+ performance - many failure options
Alternative: Specialise part of server
no change in Infrastructure & Protocols
= PCIe on host with specialized optimized TCP stack
Pipeline: Network → Consensus → Application
All specialized way faster than not

Multicast 1 sender, n receiver
examples: Communication (IPTV, live streams, video conferenc, shared rides), synchronization, publish/subscribe systems - push still pull
advertisment, discovery, data center
Motivation Performance bandwith, processing
Comfort (know all receivers?), (bugtrees)
Unicast 1:1 ≈ Broadcast 1:n in some special cases
Terminology defined well, arbitrary
source: senders
open → closed groups / multicast
Internet

Implementations - only syntactic sugar + need to know
Multicast via Unicast - source duplicates receivers
Network multicast - in-network duplication
Application-Layer " - application duplication
receivers copy & forward amongst themselves

IP Multicast only host to host, same only 1 receiver per host
host groups := multicast groups, anyone can join/send
Internet Group Management Protocol Maintenance
Addresses: has reserved 1-255 for routing/group
IPv4 class D addr. 224.0.0.0 - 239.255.255.255 negative NACK
only us dest., no ICMP error Messages Dos-attacks
IPv6: FF00::/8 well known/transient/scope/node/embedded
1111 1111 flgs step 112 group ID
Transmission Interface local: loopback (1), link local, site local (5), global (E)
in local subnet
NIC maps multi IP to corresponding layer 2 mult. addr. ethernet Lower 23 bit embedded
5 bits not used → collisions receiver also checks IP in packet
through Internet: see multicast routing

Joining group local tell router - IP_ADD_MEMBERSHIP socket option, e.g. if no crash recognition
Host: IGMP Report at join soft state
Router: regular IGMP Query (may be group specific explicit leave)
member host must answer (last host may send explicit leave)

Multicast Routing Flooding + no routing tables - bandwith - state for non-packets
Spanning Trees (similar to Minimum ST problem)
1. Group-shared trees - group uses one tree
e.g. Steiner Trees - MST - NP-hard
- info about entire Network + heuristics
- Monolithic: rerun at any change
1.2 Core Based Tree one router as core
on join may receive intermediate routers process it, where it hits tree we branch of
- less optimal tree + less complex token path
- bottlenecks near core
1 tree per source
2. Source-based Trees (may replicate each Link State Packet in AS, or one)
2.1 Shortest Path Tree - through Dijkstra
tree of shortest path to all receivers
2.2 Reverse Path forwarding
flood if meant from shortest path to sender
Optimization „pruning" cut of subtrees through soft-state "Prune" messages
unicast routing table - asymmetric routes

Application layer Distributed Applications
= systems of combined communicating components
Inter Process Communication
Point to Point model
- adverse properties: delay packet loss jitter faulty transmissions
+ remote resources
Data sources/users, administrative regions
Computation multiple machines, external Data specialized Hardware = Links, Sensors
User
Open: needs protocols (smart integrated KPIs)
specification available
interoperability between different Implementations
Proprietary by vendor without intent to open Skype MS Exchange

Client/Server model - dominant in 90s
request / response Remote Procedure Call
further model z.B. Push
Server processing iterative: process one request at a time
Concurrent: Main task accepts requests, new sub process handels request
queued by OS
Server is like underlying transport duplexes
reliable: streams, IO, atomic request, atoms requests
unreliable: datagrams, IO

DNS - Domain Name System
hierarchic distributed database
divide et impera
IN-Internet
Resource Record: Name, TTL, class, type, rd length, rdata
A-IPv4, AAAA-IPv6, NS-Name Server, MX-mail server, CName
delegation: iterative = recursive query
Name: 127 chars but compress 1-100 chars
IPs: historic
Zone: Start of Authority, versatile Domain, excl. subzones
Pointers to child zones (zone cuts)
Root
com org
Google
Zonedata in RR
alias for canonical name

Web/HTTP before Web 1970s FTP 1980s Emails, for linking documents
1989/90 at CERN Web: HTTP, HTML, Web server/browser
91: 0.9 HTML only 94: HTTPS: TLS 96: 1.0 GET, Post, multimedia 22: HTTP/3
97: 1.1 virtual hosting, security 15: HTTP/2 multiplexed stream via TCP / QUIC
HTTP host name path
www.example.de/path/info.html
base html file references objekte HyperText Markuplang.
IETF → Protocols, W3C → Web standards, Companies → own Protocols
ap to HTTP/1.1 ASCII 404 Not found
Request: GET <path> HTTP/1.1 HTTP/1.1
Host: <host name> Response: Date: In Server:
User-Agent: z.B. Mozilla/5.0 Last-Modified: In
Connection: close Content-Length:
Accept-Language: fr Content-Type: text/html
Leerzeile indicates end In data...
Status code phrase
200 Ok
400 Bad Request
301 Moved permanently
505 HTTP version not supported

HTTP/1.0 non-persistend
max 1 object over TCP con.
- Overhead
- RTT dominates
HTTP/1.1 persistend
without pipelining
with pipelining
+ efficient
- cornered problems
HTTP/2 Stream priority
+ speed
may binary
Multiplexing in TIP
Stream priority
Push
HTTP/3 QUIC
always uses TLS
0-RTT support, TLS handshake
security and data
abbreviated

User-Server State: Cookies
components:
header line in request/response
file managed by browser
Back-end database
examples
authorization
shoping carts, recommended items
user site: z.B. Email
Tracking

Peer-to-Peer - Buzzword, flat, anarchistic = Decentralised distributed System
no structure, no identifiers, only names
can be introduced algorithmically
assumptions nodes, resources anywhere
nodes cooperate
degrees of freedom
arrive, depart
random node identifier
neighbor selection
Discovery of Resources known name or solution classes
Organisation: flat, random, hierarchical structured with application
Delegation: recursive, iterative

Applications Blockchain, Video Streaming, Dynamic Load balancing, Routing traditionally File Sharing
Napster Centralized Directory Server
Gnutella floods Query, Query Hit over reverse Path for scalability
fully distributed overlay network of TCP connections limited steps flooding
many implementations usually < 10 neighbors only n hops
all peers server → highly scalable

Extended Client/Server Model
delegate sub tasks, message chains
Client Request → Response Server 1 delegation → Response Server 2

Queuing Systems
A(t) = P[time between arrivals ≤ t]
B(t) = P[service time ≤ x]
places in queue, # servers
disciplines (strategies)
FCFS first come first serve
shortest job
deadline
priority
Metrics
waiting time
time in system
backlog
initialization

Utilization factor ρ
normally $0 \le \rho \le 1$
for single server $\rho = \frac{\lambda}{\mu} = \lambda\bar{x}$

Stochastic Processes
family of random variables
X(t) indexed by time
Classifications
State space - X(t) discrete ? Then called chain
Time parameter - discrete $X(t) = X_n$
Dependence X(t)'s (dependent on previous) conditional probabilities scene?
order of dependence $P[X(t_{n+1}) \le x_{n+1} \mid X(t_n) = x_n, X(t_{n-1}) = x_{n-1}]$
$= P[X(t_{n+1}) \le x_{n+1} \mid X(t_n) = x_n]$

The Marker Process
discrete state space
first order of dependence
homogenous transition independent of observation time
memoryless x_{n+1} independent on time in x_n
exponentially distributed

Poisson Process
$\mu_k = 0 \Rightarrow \frac{dP_k(t)}{dt} = -\lambda P_k(t) + \lambda P_{k-1}(t)$
$\lambda_k = \lambda$
solving $\Rightarrow P_k(t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}$ if $k = 0$: $P_0(t) = e^{-\lambda t}$
average # arrivals in t $E[k] = \lambda t$
Merging $\lambda' = \sum_{i=0}^{n} \lambda_i$
Variance $= \lambda t$

Queuing Systems
Kendalls Notation: A/S/m [IN/K/SD]
A, S: processes (M = Markovian, D = Deterministic, G = General)
m: # servers
K: places in system (queue + served)
N: population size
SD: Queuing Discipline usually FCFS
L: # servers
service process
arrival

M/M/1
$0 \underset{\mu}{\overset{\lambda}{\rightleftharpoons}} 1 \underset{\mu}{\overset{\lambda}{\rightleftharpoons}} 2 \rightleftharpoons$ $p_0 = 1 - \rho$ requires $\rho = \frac{\lambda}{\mu} < 1$
steady states $p_k = p_0 \prod_{i=0}^{k-1} \frac{\lambda}{\mu} = p_0 \left(\frac{\lambda}{\mu}\right)^k = p_0 \rho^k = (1-\rho)\rho^k$ geometrically distributed
average customers in system $\bar{N} = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu - \lambda}$
average time spent in system $T = \frac{\bar{N}_\mu}{1-\rho} = \frac{1}{1-\rho} = \frac{1}{\mu - \lambda}$
server idle average $1-\rho$ of time
bei $\rho = 1$ variability of arrivals is the problem
utilization ρ

M/M/m Queues
$\mu_k = \begin{cases} k\mu & k \le m \\ m\mu & k \ge m \end{cases}$ $\rho = \frac{\lambda}{m\mu}$
$\bar{N} = m\rho + \frac{\rho S}{1-\rho}$
$T = \frac{1}{\mu}\left(1 + \frac{S}{m(1-\rho)}\right)$
Erlang's Formula
$S = P[\text{queuing}] = p_0 \frac{(m\rho)^m}{m!(1-\rho)}$
$p_0 = \left[\left(\sum_{i=0}^{m-1} \frac{m^i \rho^i}{i!}\right) + \frac{m^m \rho^m}{m!} \cdot \frac{1}{1-\rho}\right]^{-1}$

M/M/1/N
$p_k = \begin{cases} \frac{1-\frac{\lambda}{\mu}}{1-\left(\frac{\lambda}{\mu}\right)^{N+1}} \left(\frac{\lambda}{\mu}\right)^k & k \le N \\ 0 & \text{otherwise} \end{cases}$
Effective: $\lambda' = \lambda \sum_{k=0}^{N-1} p_k$
$\bar{N} = \frac{\rho}{1-\rho} - \frac{(N+1)\rho^{N+1}}{1-\rho^{N+1}}$
$T = \frac{\bar{N}}{\lambda}$

M/M/m better than m × M/M/1
apart from travel time
humans automatically adapt

Birth death Process special case of M
transitions only to neighbor states
λ_k Birth rate in population size k
μ_k Death rate
$k \to k-1$ Death
$k \to k+1$ Birth
$0 \underset{\mu_1}{\overset{\lambda_0}{\rightleftharpoons}} 1 \underset{\mu_2}{\overset{\lambda_1}{\rightleftharpoons}} 2 \underset{\mu_3}{\overset{\lambda_2}{\rightleftharpoons}}$
homogenous - Birth and Death independent
$P[\text{1 birth in } (t, t+\Delta t) \mid k \text{ in population}] = \lambda_k \Delta t$
death $= \mu_k \Delta t$
no $= 1 - (\lambda_k + \mu_k)\Delta t$
bei $k=0$ $P_k(t+\Delta t) = P_k(t) P_{k|k}(\Delta t) + P_{k-1}(t) P_{k-1,k}(\Delta t) + P_{k+1}(t) P_{k+1,k}(\Delta t) + o(\Delta t)$
$\mu_0 = 0, \lambda_{-1} = 0$
$\lim_{\Delta t \to 0} \frac{dP_k(t)}{dt} = -(\lambda_k + \mu_k) P_k(t) + \lambda_{k-1} P_{k-1}(t) + \mu_{k+1} P_{k+1}(t)$
Flow rate out = in if Equilibrium

Customers
T(1), T(2), T(3)
time
total time customers in sys
$Y(t) = \sum_{i=1}^{\alpha(t)} T(i) = \int_0^t N(\tau) d\tau$
$N_t = \frac{Y(t)}{t}$ average # customer
$\Rightarrow N_t \simeq \lambda_t T_t$ mean $\lim_{t \to \infty} T_t = T$
average per Customer $T = \frac{Y(t)}{\alpha(t)}$
ratio jobs enter: done
Little's Law
$N = \lambda T$

Classes
stationary Process invariante shifts in time
$F_x(x, t+\tau) = F_x(x, t)$
Independent Process white noise
Random Walk new Pos = old Pos + random Var
Renewal process Random walk but interest in counting transitions

Notation
C_n - n^{th} Customer
$\alpha(t)$ - # arrivals in $[0, t]$
$S(t)$ - # departures
$N(t) = \alpha(t) - S(t)$ # in system
τ_n - arrival time C_n
$t_n = \tau_n - \tau_{n-1}$ interarrival time
x_n - service time
w_n - waiting time in queue
$s_n = w_n + x_n$ - system time
arrival rate $\lambda = \lim_{t \to \infty} \frac{\alpha(t)}{t}$
Service rate μ
$A(t) = P[t_n < t]$
$B(x) = P[x_n \le x]$
if independent n
$\bar{T}$ - average interarrival time
$\bar{T} = \frac{1}{\lambda}$ $\bar{w} = W$ $\bar{s} = T$ $\bar{x} = \frac{1}{\mu}$
$0 \underset{\mu_1}{\overset{\lambda_0}{\rightleftharpoons}} 1 \underset{\mu_2}{\overset{\lambda_1}{\rightleftharpoons}} 2 \underset{\mu_3}{\overset{\lambda_2}{\rightleftharpoons}}$
$\lambda_k \simeq \lambda$, $\mu_k \simeq \mu$
$\lim_{\Delta t \to 0} o(\Delta t) = 0$
$+ o(\Delta t)$

CNwS - quite grand, somewhat disorganized

distributed system - one specified system by closely cooperating computers
fließende Grenze 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 reciever $d = \frac{distance}{v}$

propagation speed v - (Elektromagnetic wave in copper $\frac{2}{3}c$)

data rate r ($\frac{bits}{second}$) $\frac{Data\ size}{r}$ = End-Start or Transmission

Error rate (also error patterns)

Simplex - only one party transmits

Half duplex - parties alternate as sender/receivers ≒ TDD - Time derision duplex
on-demand: each sender presumances length, end break symbol?

Full duplex - both parties (may) send all times ⫴ two cables expensive overhead
short distance (also faster or wireless) Frequency Dirisen Duplex FDD

TDD - time derision duplex - if digital + data transmission rate ≥ 2 user data rate

Connect multiple computers: 8

each with each - way too much, does not scale

structure of end systems & and switching elements/routers

- circuit switching + simple, garanteed data flow, no queuing delay
  - Resource dedicated, even through pause, setup needed
- packet switching
  - chop data up into pacheges with administrative information
  - continues data flow → occasional package
  - typically implemented on higher layer
  - store-and-forward: receive → store → find dest → decide next hop → forward

Multiplexing - regulate access to shared recource of multiple users
- downward - multiple connections to imitate one
- upward - enable many connections over one
  - demultiplexer needed
- TDM - time derision multiplexing: one package after another
- FDM - frequency division multiplexing:
- mainly wireless: CDM - code divisen multiplexing
- mainly wireless: SDM - space divisen multiplexing

Broadcast medium and multiple access
common characteristic
- only one sender

achieved through: multiplexer

Medium Access Control (MAC)

needs common rules

BRUNNEN

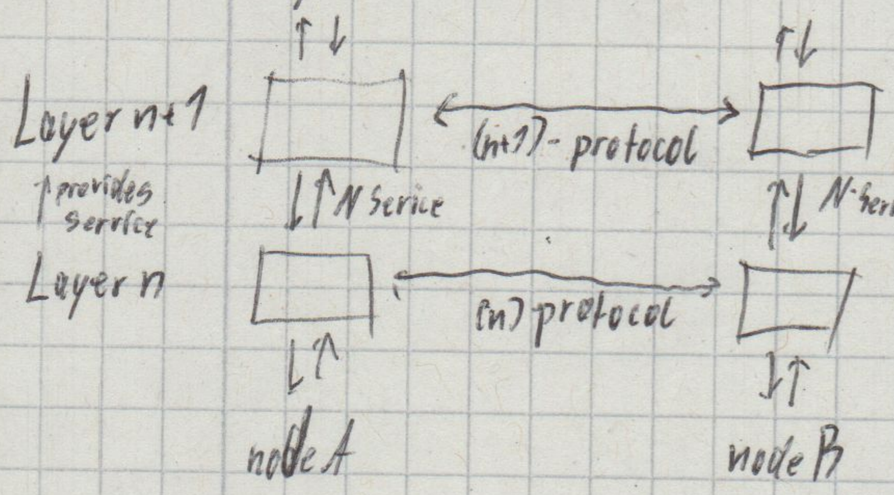
Abstraction

Network Layer Models
- ISO's OSI model (Open Systems Interconnection)
- Internet Layer model (basically subset

makes some problems „transparent"

„Distributed Machine"
- AS - autonome systems
- CSS - communication subsystems

Layered Architecture



Layer 0 has to perform actions itself

\+ independently exchangeable, interoperation A,B
\- Overhead

Tradeoff: overhead ⇔ 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: plugs, cables
- Protocols: z.B. RS232-C = ITU-TV.24

2. Data Link Layer

Error-recovering stream of frames between adjacent systems
- boundaries detected (of frames?)
- recognices, recovers transmission errors
  - through sequence no's, checksums
  - acknowledgement or timeout and retransmission
- severe errors deferred to higher layers
- may <u>flow control</u> - goal: protect slow receiver
  - z.B. sliding window protocoll, something to avoid stop and go for satelite

Broadcast Networks LAN 2 Sublayers

LLC Logic Link Contal - error detection

MAC Medium Access Control - ordered access to single medium

Protocol z.B. Ethernet

3. Network Layer

Packet stream between end systems
- routing (store&forward switching ⇔ 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)

→ Overhead

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

Protocol

defines format and order of messages exchanged between ≥2 comuni. entities and actions taken on diffrn messages (action rules)

dargestellt in message sequence chart

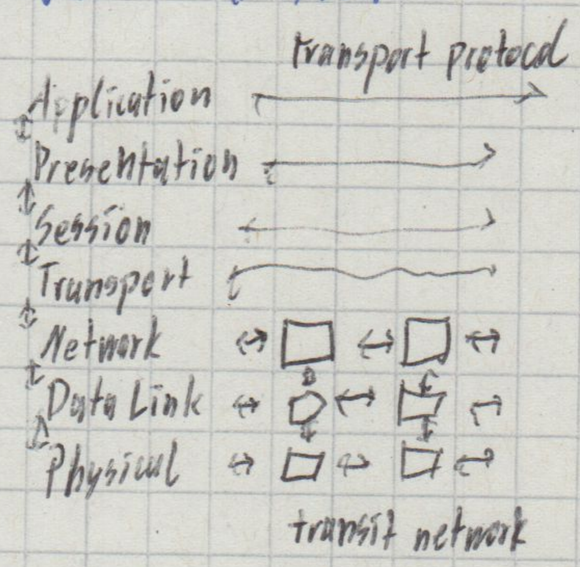
OSI terminology

PDU - Protocol Data Unit (Message) — may contain fragments
- (N)-PDU - semantics understood by peer entieties 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 final envelope, ready to send (deepest)
cell - small packet of fixed size



4. Transport Layer

Logical connections between individual processes (ports, sockets)
- Multiplexing, Packet assembly, Adress handling, error correction, flow control

Protocol: z.B. TCP — more hide network details, quality dif.
- shorter packets allow parallel (pipelining)

5. Session Layer

manages consecutive connections
- tokens (right to request)
- Communicate about checkpoints (for recovery/rollback)

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

6. Presentation Layer

harmonize data representations in diffrent end systems
- OSI standards: ASN.1 (Abstract Syntax Notation) Compare: C structs
  - BER (Basic Encoding Rules): common representation between end systems
- may encryption
- Internet: in Application Layer more pragmatic standards cf. IDLs protobf

7. Application Layer

service elements (SE) for common tasks
- association control
- CCR - Commitment Concurrency and Recovery
- ROSE - Remote Operation Service Element
- ...

for domains
- File trasfer (ftp)
- Electronic mail (X400)

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

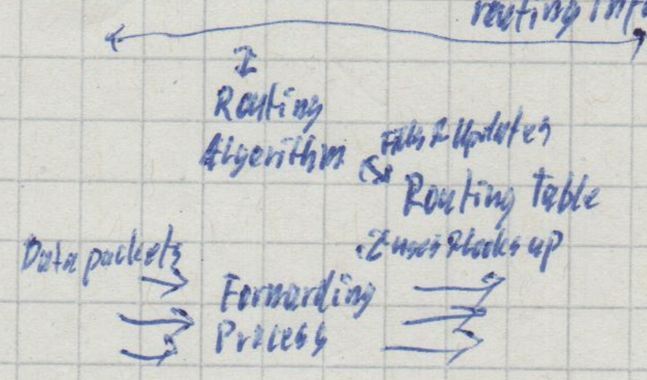
Routing: determine route
Forwarding: move packets from routers input to apropiate output

"cheap" "good"

find route in network of endsystems & routers

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

SDN controller



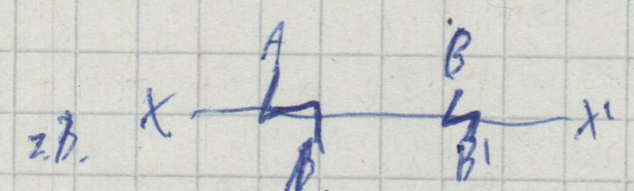
CO vs CL on Network Layer Services (NS) - CONS every intermediate system knows connection

- CL NS - datagram service: routing at setup, forwarding through (short) 'circuit ID'
- CONS - 'virtual circuit': routing algorithm usually periodicly or at first for destination

Goals:

- Correctness
- Simplicity
- Robustness
- Stability
- Fairness
- Optimality

z.B.



optimal, aber nicht für

often using metrics:
- average package delay
- total throughput
- individual delay

practice:
- mostly minimize hops

Routing Algorithms

Non-adaptive routing algorithms: dont base on current state

Flooding: sent to everyone but source and endsystems
reducing packages:
- TTL: in max number hops (needs to be long enough to find dest) (erstaunlich wenig nötig)
- Sequence numbers: only forward once

+ fast changing networks z.B. mobile
+ multicast to many anycasts z.B. Crisis Application (if hops geödert)
z.B. distributed database

Static Routing: preconfigure all possible dest

+ in static, predictable environments
- networks changes frequently
- if traffic varies → huge over provisioning

→ Adaptive Routing Algorithms: use network state (traffic-adaption 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
- Bottleneck: 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 sources
start with random routing: hot potato or flooding

"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 (z.B. peering

Distributed adaptive routing

delay
$

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

Decentralized:
router knows it links and cost
iterative computing, exchange
„distance vector" algorithms: z.B. RIP protocol, BGP potocol (path vector)

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 potocol

DVR

Distance Vector Routing (distributed, self-terminating)

$D^x(y,z) = dist\ x \xrightarrow{via\ z} y$
$= c(x,z) + \min_w \{D^z(y,w)\}$

data structure: Row for each possible destination
Column for each direct neighbour

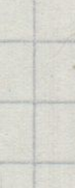
distance matrix D^x: dest | links
routing table: dest | link, cost
distance vector: dest, cost

wait
recompute
notify

Init ∞

changes: good news travel fast
bad news slow - "count to infinity"-problem

if Z routes through Y to X, Z tells Y its distance to X is ∞



Split Horizon / ⇒ Poisoned reverse: ~~if something gets worse share it fast~~
with poisoned reverse sends it with ∞
no fix in bigger network

doesn't even send updates ⇒ Split Horizon: don't advertize to neighbour that send update
no full fix

errors propagate through network

Link State Routing LSR (global)

input for algorithm
graph (V, E)
nodes, links
mapping c(v,w) = cost

2. B. Dijkstra

exchange: link state broadcast

links
O(n·E) send each time

each router builds, link state packets and floods them
cost to all neighbors
sequence number, age field

Speed of convergence n times $O(n^2)$ algorithm.

Potential prob

Hierarchical Routing

Growth of net maybe fast for flat network

Administrative autonomy: each network admin may want to control routing

increasing number is a problem

Autonomous Systems (AS) (~60.000 stück)
- Stub AS: small corporation (only one link to internet)
- Multihomed AS: large corporation (multiple links, but no transit)
- Transit AS: provider

everyone unique number
every AS must know a route to every network
differen 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

policy may dominate

inter-AS routing
BGP - Border Gateway Protocol: Path Vector (allows loop avoidance)

if subnet and gateway

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

de facto standard
BGP peers have BGP session (semi permanent TCP connection) (not 1:1 to physical links)
advertising premises to forward
- route = Prefix + attributes
  - AS-PATH
  - NEXT-HOP (specific gateway router)

gateways accept/decline after import policy

eBGP - between AS's
iBGP - between routers of AS

session

advertising after policy
z.B. a dual homed customer may not want to forward between it's providers only to/from it's customers

Potential Problems
- Sub-optimal routing
- Route instability
- Route oscillation
- Route flapping

Security
- blackholing
- redirecting

Mobile Routing

Dynamic Source Routing - DSR (IETF RFC 4728)

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 it own address
- destination returns packet to sender

(without bidirectional paths flooding back)

path maintainance

only if used, unused paths get eventually deleted

or

Layer 2 acknowledgement (z.B. WLAN)
Carrier sense: listen if other station forwards (doesn't work with directed antennes)
Request explicit acknowledgement

in case of problems in intermediate node
inform sender
or lookup 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 then through 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 identifies, neighbor selection

Discovery of resources
know either name or location

solution classes (strategies to adopt to variation of assumptions)

Organisation:
flat random, hierarchical, structured (with mapping)

Delegation
recursive, ~~iterative~~ iterative

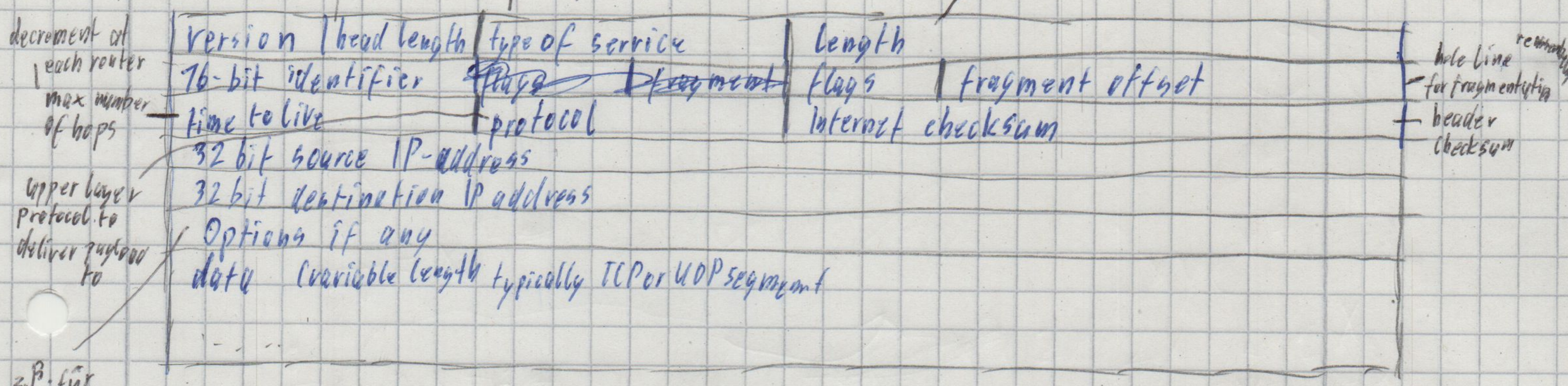
Internet Network Layer

Internet: Connectionless
forwarden nach dest. host address

Routing Protocols → Forwarding Table

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)



Fragmentation
all fragments same 16-bit identifier
Flags: DF: Don't Fragment
MF: More Fragments (set by all but last fragment)
Fragment Offset = original position in 8octets (unit)

ARP 1 IP → MAC
DNS IP → Host name

flat, permanent
topological (mostly)
mmo.... hierarchical

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

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

IP address classes (two layers of hierachie + subnets)

classfull:
A | 0 | Net (8 bits) | Host (24 bit)
B | 10 | Net (16 bits) | Host (16 bits)
C | 110 | Net (24 bit) | Host (8 bits)
D | 1110 | Multicast address
E | 1111 | Reserved

unfaire verteilung einige früh Unternehmen Westen z.B. Ohio deswegen sehen mehr IPv6

Specials Network Address: all host bits 0
Broadcast Address: all host bits 1

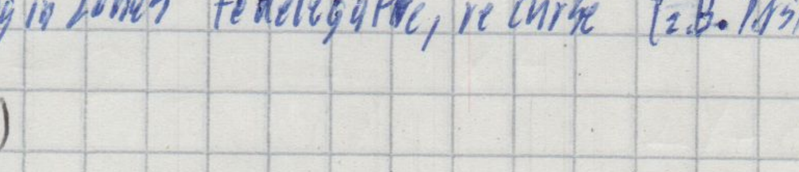
Loopback 127.anything for loop back testing
processed locally

Private IP addresses 10/8
172.16/12
192.168/16

Link local 169.254.0.0/16

Network Prefix = IP & Net Mask ⇒ ~~CIDR~~ IP/#1 in Net Mask (immer führend)

Routers usually multiple interfaces each with own IP
⇒ IP for host and router interface
⇒ connection between two routers is own subnet!!



Subnet: device interfaces with same subnet Prefix
can physically reach each other without router
all detached islands from router

+ organisation may want to split departments each admin own address space
+ ~~one~~ some departments may be smaller

class A can have only 2 subnet ⇒ VLSM (variable length subnet mask)
B 1 multiple subnetwork hierarchies possible

CIDR: Class less Inter Domain Routing

CIDR supernetting (discard class boundaries
"longest match routing"

advertise many access with smaller Prefix

ISP assigns contigues group of class C blocks

ICANN → ISP → Organisation → Host

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

Host
static: hard coded by sys admin
or retrieved from local server RARP
reverse Address Resol. P.

DHCP: Dynamic Host Configuration P.
configures IP address
default gateway
(+ plug and play <-)

example forwarding table

Prefix Match	Link Interface
11001001...	0
00010	1
00011000	2
00011	3
otherwise	

NAT - Network Address Translation - Motivation: Shortage of IP addresses
router übersetzt interne addressen nach außen in einen durch Aufsetzung ports
+ only one (extern) IP for network
+ IP address extern und intern funktionsabhängig
+ Devices not explicitly addressable (security)

private IP addressen nur noch in lokalem Netz eindeutig

16 bit port-number field
⇒ 65k simultaneous connections

incomming
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 e2e principle (z.B. complicated for P2P applications)
- no servers behind NAT (server classes reached by port)
← IPv6 better solution

IP → MAC
Address Resolution Protocol
ARP: Broadcast IP asking for MAC
cache
+ plug and play

1. broadcast DEST MAC = ?
2. unicast back
3. caches in IP-to-MAC pair in ARP table with timeout

- ARP cache poisoning

BRUNNEN

IPv6

128 bit addresses
Routing und switching works in same way
new addressing architecture
new Protocols

reason: more IP addresses
(get rid of NAT ;))
IPv4 practical limit of 250M
technical reason:
"traffic class" & "flow labels" may
enable (better actually used) QoS
flexible number of hierarchy levels
serverless plug and play possible
end-to-end, IP-layer authentication & encryption
Better for mobile IP "triangle routing" ;) possible

wie Tos in v4 / identify connections bit / forbidden word

Version	~~Flow label~~ Traffic Class	Flow Label
Payload length	Next header	Hop Limit
Source address		
Dest. address		

without header

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

IPv6 Addresses
128 bits: written 8 16 bit hex numbers — one or allowed filled with 0s bis 128 bit
leading zeros required

Aggregate Global Unicast Addresses
~~0041TLA ID~~

001	TLA ID	Res	NLA ID	SLA ID	Interface ID
3	13	8	24	16	64

Next Level / site-level / may be based on MAC (FFFE inserted 24 bit (inverted) but that would be a big privacy concern

Top Level aggregation (IANA assigns global ISP)

reserved for extension of TLA and/or NLA

Transition from IPv4 to IPv6
nicht alles auf einen Schlag umgestellt
⇒ interoperation

tunneling
IPv6 packets carried as payloads in IPv4 Packet

IPv6 router usually also can IPv4

BRUNNEN

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 advertisment
- RS - router solicitation
- RA - router advertisment

affects related protocols

ARP existiert durch NDP oder ND des ICMPv6

Link-local addresses werden gebraucht
FE80::/10 prefix MAC, DAD, allows IP multicast

DHCPv6 existiert but SLAAC recommended
Stateless Address Autoconfiguration
Router announces /64 network (RA)
remaining 64 bit determined
a) automatically via EUI-64
b) privacy extensions SHA1 (current NTP time + MAC) + 64 bit

DNS: new AAAA records
more new functions

Transport Layer

Logical communication between processes (run in end-systems)

Segmentation if N-SDU > (N+1)PDU

Service primitives to access service SAP

SAP - Service Access Point
CEP - Connection EndPoint

- Name {CONNECT, DISCONNECT, DATA, ...}
- Type {request, indication, response, confirm}
- Parameters {Parameters (header field(s)), payload}

blocking

Multiplexing / Demultiplexing

<prim>.type characterizes primitive
Usually described in Time Sequence Diagrams (Message Sequence Chart)
.req, .conf only for confirmed services
Prefix for Layer

blocking / unblocking

Addressing

OSI ~~suggest~~ suggests identify remote process via local CEP
- Problem: How to initially identify?

idea operating systems PID
- Problem: arbitrary; ~~en~~changes with restart, only one service per process

⇒ abstract service name or address (z.B. fileserver)

Internet: abstract service addresses = Protocol Ports
Process can register ports. OS queues packets for ports

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

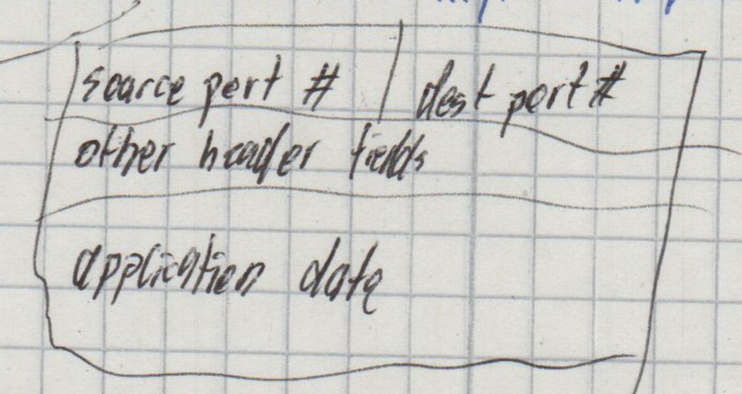
(<local socket>, <remote socket>) fully identifies connection ⇒ violates OSI

well known ports 0-1023 IANA (requires root)

echo 7/udp	ftp	20/tcp	file transfer
time 37/udp		21/tcp	control
name 42/udp	ssh	22/tcp	
dns 53/udp	smtp	25/tcp	
	dns	53/tcp	
short lived, ephemeral ports	http	80/tcp	
(unregistered)	https	443/tcp	

Demultiplexing in Internet

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



Connection Control OSI Terminology

Establishment

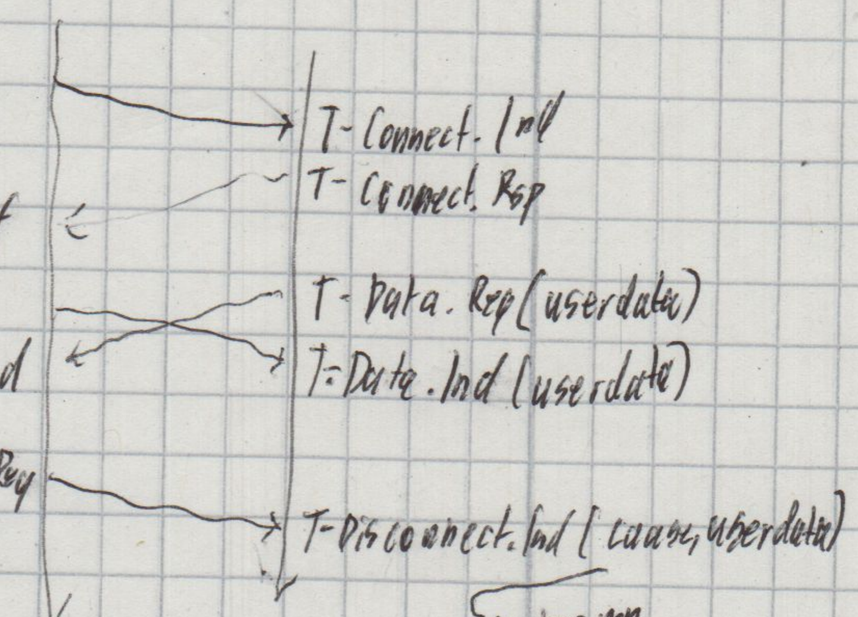
called transport service user / calling

T-Connect.Request (Destination addr, source addr)
- Indication (dest. addr., source addr.)
- Response (response addr)
- Confirmation (response addr)

responding service user

may be address, afterwards CEP

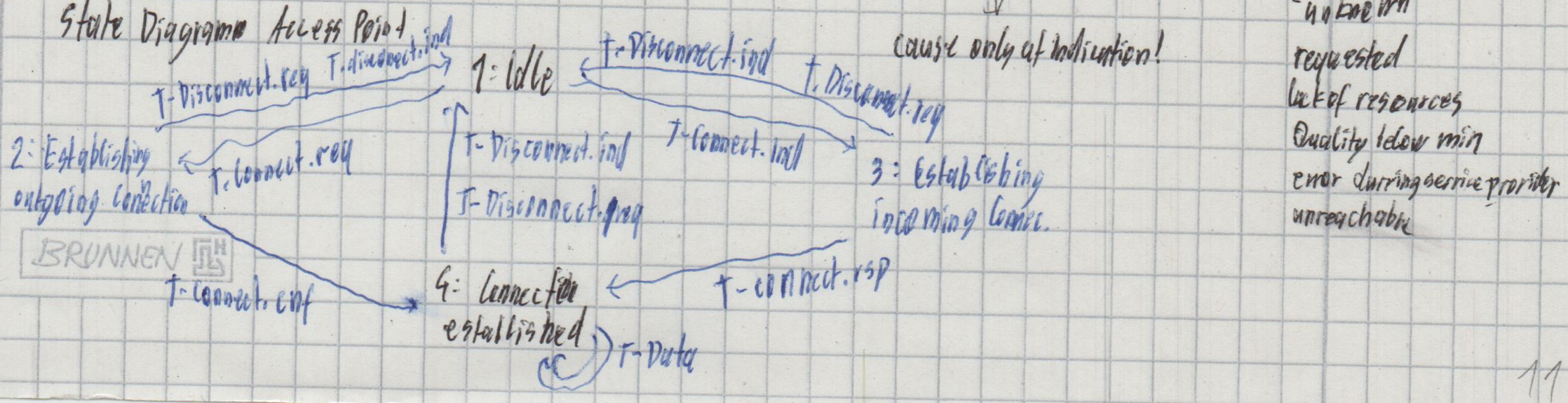
Example



cause only at indication!

unknown
requested
lack of resources
Quality below min
error during service provider
unreachable

State Diagram Access Point

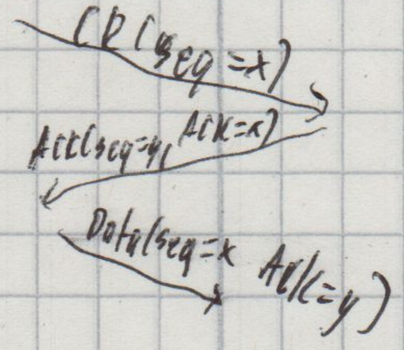


Errors during Establishment

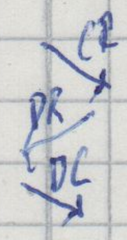
Loss of CR => Timeout resend but Duplication

Loss of CC => Three-Way Handshake (ACK or Data(DT) acknowledges CC)

but delays, duplication or replays => sequence numbers
reuse after packet_lifetime
or timestamps
only establish with right sequence



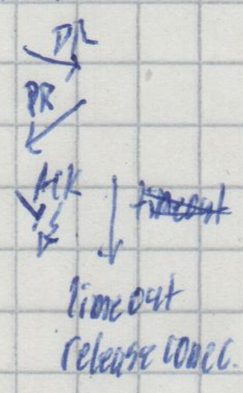
Connection Rejection / Connection Release



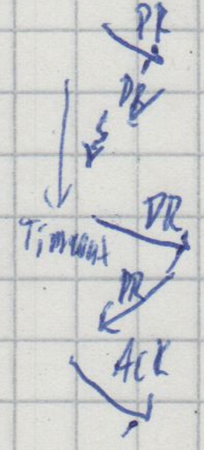
Teardown before data can cause data loss
Implizit: Teardown of network layer connection
Explicit: connection release with Disconnect-TPDUs (slides 04 p. 36)

but how to be sure the ~~other~~ both aggree on teardown ? => two army problem impossible

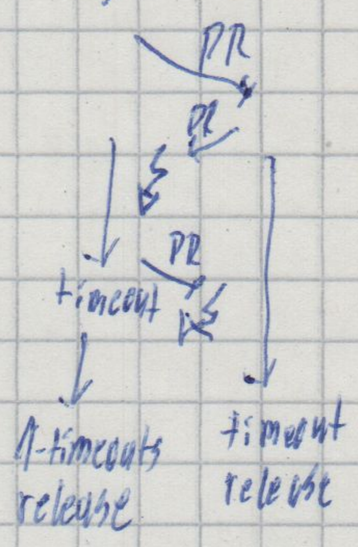
Praxis: 3WHS scenarios lost ACK



lost 2. CR



many lossed



possible errors
dropped packets (biterrors, congestion, ...)
Erroneous
receiver slower than sender → Flow Control - protect slower receiver

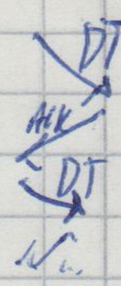
On NT-Layer: PDUs size differs widely
processes might take time to answer => T-layer needs to buffer
parallel connection => dynamically

can't assume receiver is fast enough or has enough buffer

manage sending rate
- slow when error
- ~~sender rejects~~
- receiver gives credit

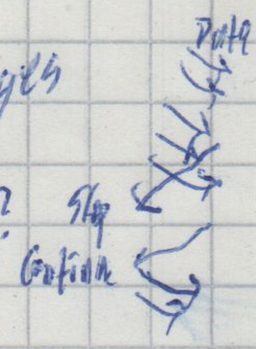
Alternating-Bit-Protocol
- reliable
- assumes free buffer after receiving

step and go!
- only ack if processed
- only 2 packet numbers



short Timeout? long
overhead ← → late reaction
- slow

step and Continue Messages
z.B. XON/XOFF Protocol
arrives stop before overflow???



T_{prop} = ~~distance~~ / v (Lange Weg / Geschwindigkeit)

T_{trans} = size / capacity (Bit / sec)

rate-based - receiver indicates max rate
- + low overhead
- - rate can get messed up by delay
- - no confirmation
- - no error handling

⇓

credit-based - receiver indicates max packets
sender only sends with credit
- absolut credit
  - - might get received at different time
- credit window (= sliding window)
  - credit in relation to acknowledged packets

credit based

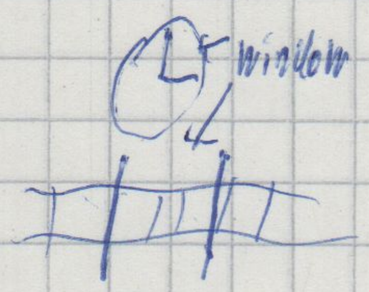
primitives?

Permits & ACK indicates → can be combined with window size ⟵ problem if ack retained to ack before possibility to process more
sender timeouts and resends

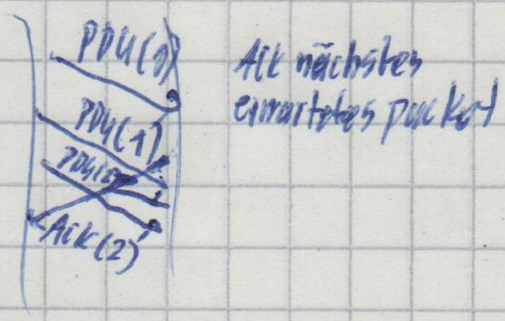
ack ⇒ stop timeout, delete local copy

flow_advance ⇒ advance window, possible send more data

Sliding Window



example with explicit permits



Alt nächstes erwartetes packet

Error Control - reliable transfer

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

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

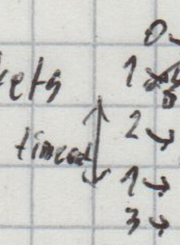
sequence numbers

⇒ only reliable scheme timeout @ sender → resend

ARQ - Automatic Repeat reQuest

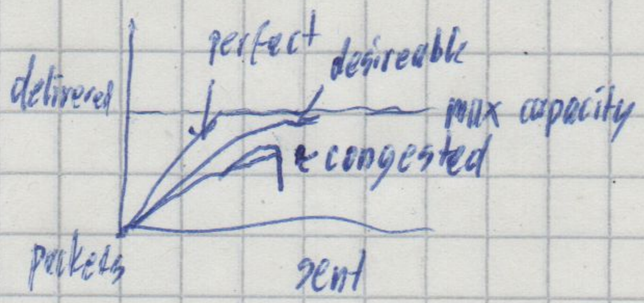
- type 1: Stop-And-Wait (siehe Alternating-Bit-Protocol)
- type 2: Go-Back-N
  - considers current and consecutive packets lost

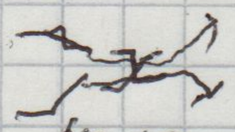
- selective Repeat
  - only retransmit lost packets



Congestion Control

what with bottleneck in network?



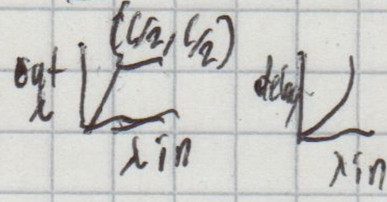


scenario 1

2 sender, 2 empfänger

infinite buffer

no retransmission

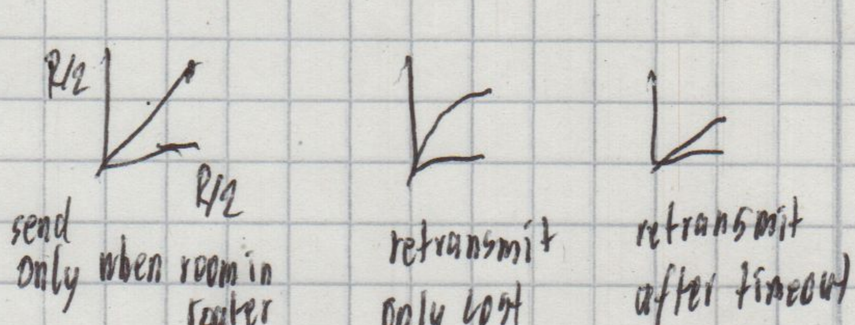


2) finite buffers ↪ lost packets

λ'_{in} = original data + retransmitted

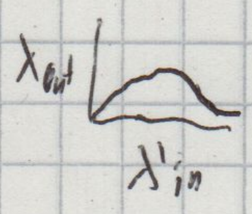
goodput $\lambda_{in} = \lambda_{out}$

perfect: only lost packets retransmitted



congestion ⇒ 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

↓

sending rate adapt to networks actual capacity

Global issue ↪ Flow Control local

depends on all routers, routing, ...

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)
- Closed Loop: feedback allows sender to adapt
  - explicit feedback: point of congestion informs sender
  - implizit feedback: sender deduces congestion from network behaviour (e.g. missing ACK)

dropping packets
- ~~congestion~~ should never happen
else reservation or design wrong

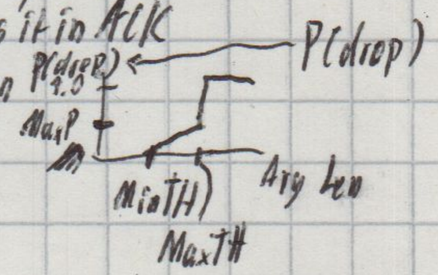
Actions:
- increase capacity: usually not practical (maybe long term)
- reservations and admission control: dont allow additional traffic if near limit
  Usually when feedback of state rare, ~~or~~ e.g. for circuit-switched or similar
- reduce load: without terminating on-going connections
  usually requires closed loop

Taxonomy (← Internet (except for 'dropping packets'))
- router centric vs. host centric
- window based vs. rate based vs. ... (z.B. credit based)

Router Actions - dropping packets when buffer full

which?
- drop-tail queue (old packets important for go-back-n transport protocol)
- old packets (new for multi-media traffic more important) (maybe even drop from same flow but overhead)

Feedback
- implizit: assumes loss only caused by congestion → reduce rate ↯ for wireless (securing L2 helps)
- explicit
  - 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 copys it in ACK
  - RED - Random Early Detection: dropping propability relative to congestion



UDP - User Datagram Protocol [RFC 768]

connectionless (each ~~part~~segment independently)
best effort

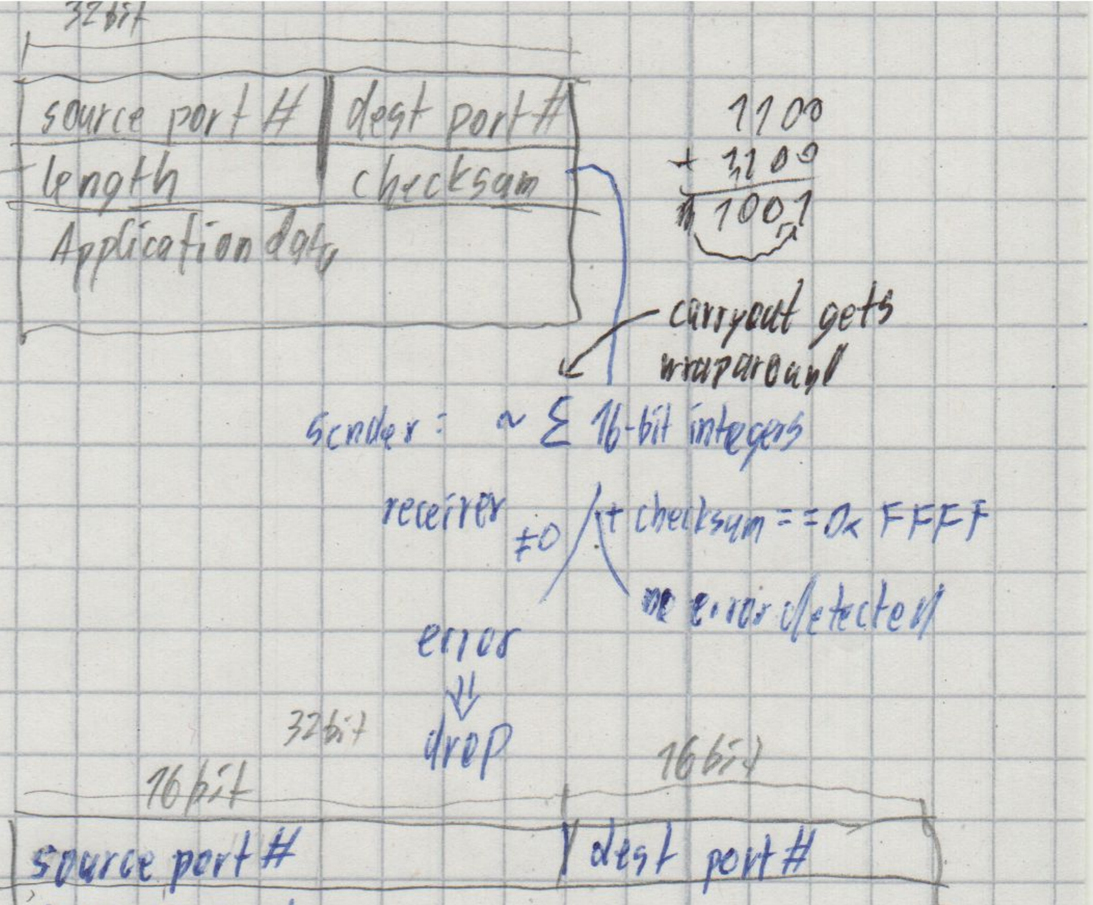
+ multimedia apps
  - loss tolerant
  - rate sensitiv

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

(register frame an wichtigsten)

Demultiplexing: sockets
source port = return "address"

socket identified by dest



TCP - Transmission Control Protocol (TCP)

Point-to-point No multicast

connection oriented
Flow, congestion control

Reliable in-order byte stream

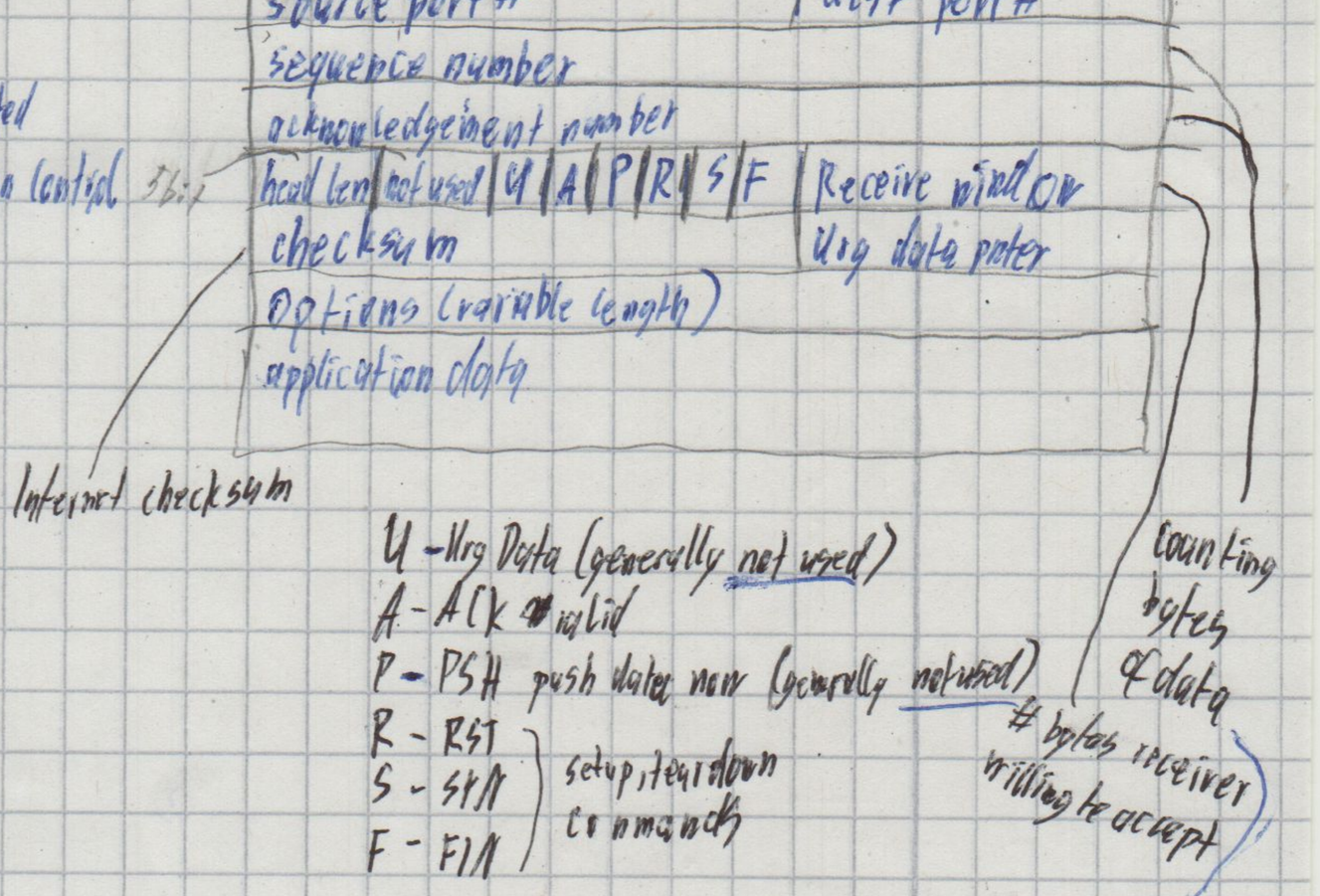
Pipelined

Full duplex (MSS - maximum segment size)

send & receive buffers (old TCP used GoBack N)

socket identified by source, dest pair

(connection Establishment)



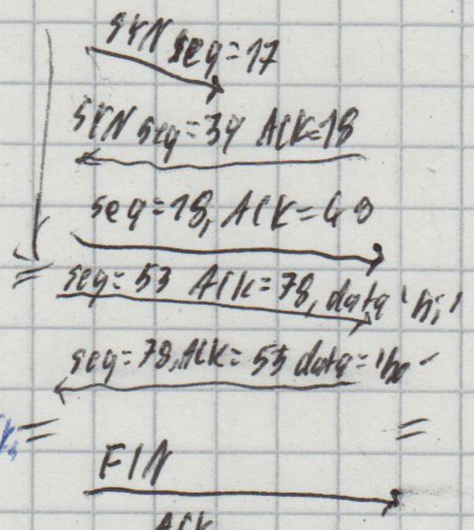
U - Urg Data (generally not used)
A - ACK valid
P - PSH push data now (generally not used)
R - RST
S - SYN
F - FIN
} setup, teardown commands

Connection Establishment

active mode (connect)
request connection with transport service user (IP+port)

passive mode (listen/accept)
tells TCP port or all ports
upon incomming connection ⇒ new socket created

3-Way-Handshake
Negotiation window size, sequence numbers



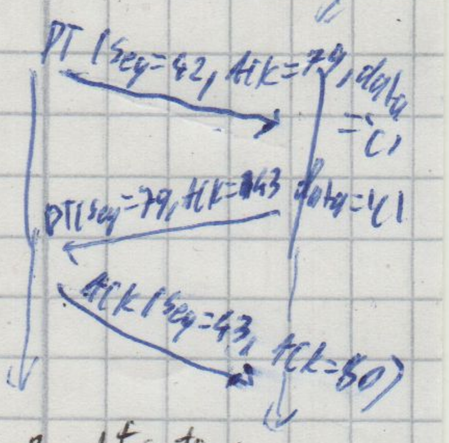
Piggybacking of ACKs

Last thing gets ACK
solves premature timeout, allows cumulative ACKs
confirmed release

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

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

ACK means expected



timeouts: 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 ← Nagle's Algorithm

congestion window

→ self-clocking if implicit feedback from dropped packets
ACK-clocking

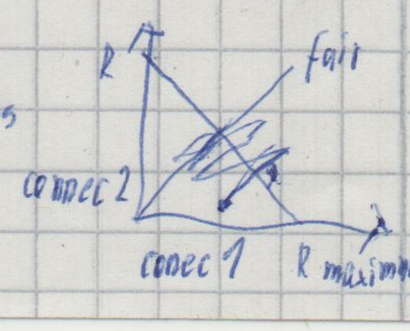
⇓

congestion window: ~~bad~~ multiplicative decrease (50%)
additive increase
(would need to track RTT so Näherung
increment = MSS · (MSS/congestion window)
with each ACK

AIMD
additive increase
multiplicative decrease

⇓

Sawtooth pattern
+ drives to fairness



Slow Start (hysteric name) until ~~first~~ loss
double congestion window

start fast retransmit if not triple duplicate ACK

Problem packet bursts even if many ACKs lost ⇒ reset window to 1
Slow start ←
with previous window as congestion threshold

Internet transport today

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

↳ fewer portal protocols

Growth of ~~CA~~ CDN - Content Distribution network

~~[redacted]~~

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 messag streams into 1 connection

Multi-homing: both multiple IPs for failover

DCCP - Datagram Congestion Control Protocol

~~[redacted]~~ unreliable, timely

connection oriented, congestion contrd, optional Acks of Acks

seperate header/data checksums

CC 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 (redsync, hardly used)
- encryption
- payload info (audio encoding, ...

RTCP last sequence received
observed loss rates
jitter
member info
control algorithm
} from various senders

QUIC ~~(Quick UDP Int~~ (Quick UDP Internet (connections) - proposed by Google

used

integrates with Security and HTTP/2 efficient multiple streams

- faster with known server
- Crypto block = packet boundaries

adjust rate to RTT

forward error correction

Application HTTP/2 | HTTP/2 shim
Security TLS | QUIC
Transport TCP | UDP
Network IP

On UDP
+ firewalls block unknown protocols

Security threats require QUIC changes
↓
fast in user-space

Queuing Systems

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

places in queue
servers
disciplines (strategies)
- FCFS - first come first serve
- shortest job
- deadline
- priority

Metrics:
- waiting time
- # in system
- backlog undone work
- utilization

notation $\lambda_t = \frac{\alpha(t)}{t}$

- arrival rate $\lambda = \lim_{t\to\infty} \frac{\alpha(t)}{t}$
- service rate μ

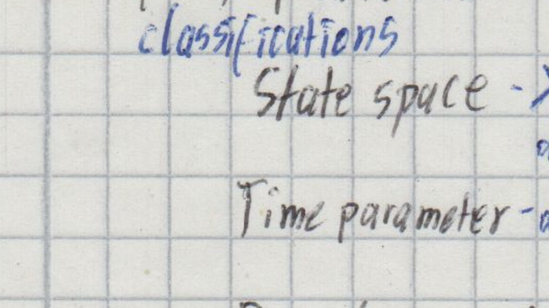
C_n - n^{th} customer
$\alpha(t)$ - # arrivals in $[0,t)$
$\delta(t)$ - departures
$N(t) = \alpha(t) - \delta(t)$ - # in system

τ_n - arrival time C_n
$t_n = \tau_n - \tau_{n-1}$ - interarrival time
x_n - service time C_n
w_n - waiting time in queue
$s_n = w_n + x_n$ - system time

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

~~für random t_n independent of n~~

$\bar{t}$ - average interarrival time if independent n
$\bar{t} = \frac{1}{\lambda}$ $\bar{w} = W$
$\bar{x} = \frac{1}{\mu}$ $\bar{s} = T$



total time customers in system $Y(t) = \sum_{i=1}^{\alpha(t)} T(i) = \int_0^t N(\tau)\,d\tau$

average per customer $T_t = \frac{Y(t)}{\alpha(t)}$

average # customer $N_t = \frac{Y(t)}{t}$ $\Rightarrow$ $N_t = \lambda_t T_t$ wenn $\lim_{t\to\infty} T_t = T$ $\Rightarrow$ $N = \lambda T$ Little's Law

Utilization factor ρ ratio jobs enter : jobs done
normally $0 \leq \rho < 1$
for single server: $\rho = \frac{\lambda}{\mu} = \lambda\bar{x}$

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 scene)
  - order of dependence $P[X(t_{n+1}) \leq x_{n+1} \mid X(t_n) = x_n, \ldots X(t_0) = x_0]$

Classes:
- Stationary process invariant to shifts in time $F_X(x, t+\tau) = F_X(x, t)$
- Independent process (white noise)
- Random walk
  - new Pos = old Pos + random variable
- Renewal process
  - Random walk but interest in counting transitions

The Markov Process
- discrete state space
- first order of dependence $P[X(t_{n+1}) \leq x_{n+1} \mid X(t_n) = x_n)$
- homogenous transitions independent of observation time
- memoryless X_{n+1} independent on time in X_n
- exponentially distributed

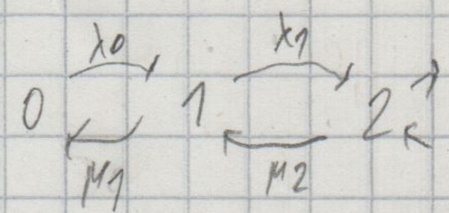
← Birth-Death-Process
- special case Markov transitions only to neighbor states
- λ_k Birth rate in population size k
- μ_k Death rate

Homogenous
Birth and Deaths independent

$P[1 \text{ birth in } (t, t+\Delta t) \mid k \text{ in population}] = \lambda_k \Delta t + o(\Delta t)$
death $\mu_k \Delta t + o(\Delta t)$
no $= 1 - (\lambda_k + \mu_k)\Delta t + o(\Delta t)$

keine simultanen Übergänge für $\Delta t \to 0$ $o(\Delta t) = 0$

$k \to k-1$ Death
$k \to k+1$ Birth



bei $k=0$ $\mu_0 = 0$ $\lambda_{-1} = 0$

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

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

Flow rate out = Flow rate in (falls Equilibrium)

Poisson Process

Deathrate $\mu_k = 0$
Birth $\lambda_k = \lambda \ \forall k$

$\Rightarrow \frac{dP_k(t)}{dt} = \begin{cases} -\lambda P_k(t) + \lambda P_{k-1}(t) & k \geq 1 \\ -\lambda P_0(t) & k = 0 \end{cases}$

solving $\Rightarrow P_k(t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}$

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

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

Equilibrium Birth-Death Systems

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

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

flow conservation over any closed boundary

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

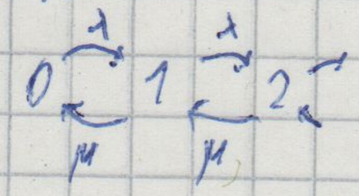
$\Rightarrow p_k = p_0 \prod_{i=0}^{k-1} \frac{\lambda_i}{\mu_{i+1}}$ $k = 0, 1, 2, \ldots$, da $\sum p_k = 1 \Rightarrow p_0 = \frac{1}{1 + \sum_{k=1}^{\infty} \prod_{i=0}^{k-1} \frac{\lambda_i}{\mu_{i+1}}}$

Queuing Systems

Kendall's Notation: $A/S/m[/N/K/SD]$

- places: # in system, assumed ∞ if not given
- Population size
- queue discipline, usually FCFS
- m: # servers
- S: service process
- A: arrival process
  - M = Markovian
  - D = Deterministic
  - G = General
  - ...

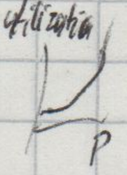
M/M/1



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

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

Server idle on average $1-p$ of time



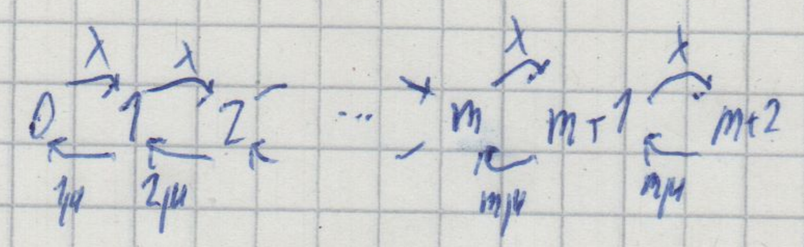
bei $p \to 1$ variability in arrivals is the problem

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

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

M/M/m Queues

$\mu_k = \begin{cases} k\mu & k \leq m \\ m\mu & k \geq m \end{cases}$



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

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

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

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

M/M/1/N

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

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

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

$T = \frac{\overline{N}}{\lambda}$

M/M/m better than m × M/M/1
apart from travel time
humans automatically adapt

Time

mismatch → problems

Difficulties:

Quartz Oszilators sensitive

⇓

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

⇓

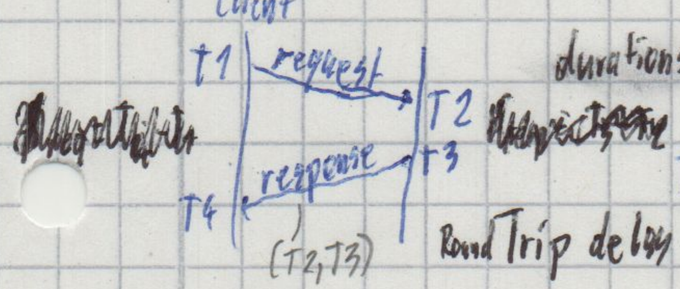
Synchronization

Internet

bat asynchronos
besteffort

⇓

Cristian's Algorithm



durations comparable

Transmission delay

Round Trip delay $\delta = \delta_{req} + \delta_{resp}$

$= (T_4 - T_1) - (T_3 - T_2)$

estimation:

$\delta_{req} \approx \delta_{resp}$

⇓

$clock := T_3 + \frac{1}{2}\delta$

⇓

Logical Clocks / Lamport Clock - orders events

$C(a)$ - clock time wen event a

$\forall a,b: a \to b \Rightarrow C(a) < C(b)$

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 $(C(a), i) < (C(b), j) \Leftrightarrow C(a) < C(b) \vee (C(a) = C(b) \wedge i < j)$

⇓

total order

UTC - by atomic clocks radio broadcast

- needs radio (to milliseconds)
  or GPS (to microseconds)

⇒ At best 100s of micro/milli seconds

standardized NTP - Network Time Protocol

error probabillity small

but never zero

$a \to b$ a happens before b

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

order transitive

in same process use physical time

diffrent - order if direct effect

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

receive update from client → broadcast to all (including self)

from replica → add to queue (sorted by Lamport time)

if head broadcast ack

On ack → mark ack'd

all ack'd → remove & process

send ack for new head

- scalability
- performance of slowest
- no protection against
  node fail
  lost messages

BRUNNEN

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

Primary-Backup Replication

replica = state machine, update = transission
determistic - same order ⇒ same result

All-or-nothing atomicity

One Primary communication with backups
- Reliable Atomic Broadcast

- Primary receives update → order / log
  replicate log to backups
  backups → execute log
  ack
  Primary if all ack'd → execute log
- fails ?
  - Reconfiguration - fresh node if backup fails (needs logs)
  - Leader election - if primary fails can use Consensus Protocol

Further reading:
- 80s Paxos protocols for Crash Fault Tolerant (CFT) consenses
- 00s Zookeeper widely-used CFT meta-data-store
- 2013 Raft RSM easy to understand
- Grundlagen 90s PBFT ~~BFT~~ Practical Byzantine Fault Tolerance
- 2018 Hotstuff higher throughput

CAP Theorem

only two properties at same time

Consistency
CA CP
Availability AP Partition Tolerance

Hardware

Re-programmable Specialized Hardware

FPGA - Field Programmable Gate Array
+ Free choice of architecture
+ Fine-grained pipelining, distributed memory, communicating
- code occupies space
- only in 100s MHz range

- Code → Synthesized Circuit → Placed & Routed (map onto specific FPGA)

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

In-network with Network-attached without Host CPU

Bottlenecks

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

Switching
faster links
but CPU limits scalability
needed data gets more

1990-2000 CPU Frequency Scaling
2000-2015 Commodity in Cloud
2015- Specialized Hardware (more logical cores ↓)
- GPU, TPU, FPGA
- NICs, Switches
- Smart Drives
- Motherboards with spez. Chips

more flexible
CPU ×10→ FPGA ×10→ ASIC
more efficient

Coordination bottleneck

z.B. Zookeeper broadcast, ack, commit (1 Leader, n Followers)
TCP

network access + traverses layers → overhead

parallelism? can only pipeline

operation on network level - NetChain Consensus with programmable switches at line rate
- 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

remove CPU from critical path
RDMA - Remote direct memory access of NIC z.B. DARE
+ performance - many failure options z.B. detect CPU fail?

z.B. PISA - Protocol Independent Switch Architecture
programmed with P4

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

Alternative: Specialise part of server no change in Infrastructure, Protocols
FPGA on host with special optimized TCP-stack Pipeline: Network → Consensus → Application

BRUNNEN

All specialised faster than not
important for bottlenecks

Multicast data to multiple receivers
1 sender

examples: IPTV
communication (video conference, shared live stream, notes ~~whit~~)
synchronization
publish/subscribe systems - push/pull
advertisment, discovery
data center

motivation
+ Performance (bandwith, processing)
+ Comfort (know all receivers ?)
(+ Gurantees)

↔ Unicast 1:1 standard internet connection
↔ Broadcast 1:all Radio

in theory special cases

Terminology
sources := senders defined role ⇔ arbitrary
open ⇔ closed group
open ⇔ closed multicast
Internet

Implementations

Multicast via unicast - only „syntactic sugar"
source duplication - need to know all receivers

Network multicast
In-network duplication

Application-layer multicast receivers copy and forward
application duplication amongst themselfes

IP Multicast only host to host (assumes one receiver per host)

host groups := multicast groups
anyone can join/send

IGMP - Internet Group Management Protocol

Addresses

IPv4: class D addresses 224.0.0.0 - 239.255.255.255
reserved
↓
224.0.0.0 - 224.0.0.255 for routing and group maintainance
only as dest
no ICMP error messages
↳ DoS Gefahr

IPv6: FF00::/8 (8 bits: 1111 1111) flgs (4) scop (4) group ID (112)
flgs: well-known / transient / embed rendezvous point
scop: interface local = loopback (1) / link-local / site-local (5) / global (E)

Transmission
local subnet: NIC maps multicast IP addresses to corresponding layer 2 multicast address
through internet: receivers notify their IP / see multicast routing
lower 23 bits embedded to ethernet multicast; 5 bits not used ⇒ collisions

Joining group (cont)
Host: IGMP Report at join (IP_ADD_MEMBERSHIP socket option)
~ soft state (no need for unjoin z.B. if crash)

Router regular IGMP Query (may be group specific)
member host must answer (last host may send explicit leave)

BRUNNEN

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 - info about entire network
+ heuristics - Monolithic: rerun ect on change

Core Based Trees
- one router
- less optimal tree
+ less complex
- bottlenecks near core

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

Source-based trees

- may profit at from link state routing in IS

Shortest Path tree - with Dijkstra
tree of shortest path to all receivers

Reverse Path forwarding
relys on unicast routingtable (or own one)

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

optimization
"Grafting": cut of subtrees through soft-state "Prune" messages

Application Layer

Distributed Applications - system of combined communicating components

IPC - Inter Process Communication
Point-to-Point model

Process — OS-kernel — Transfer System — OS-kernel — Process

+ remote recources
- delay, jitter, can't be hidden
faulty transmissives
packet loss

Data: sources/users, administrative reasons
Computation: multiple maschines, specialized hardware (diff. sensors), exclusive data
User

needs protocols
- Open: specification available, interoperability between implementations; most Internet in RFCs
- Proprietary: z.B. Skype, MS Exchange

client/server model - dominant in 90s — further Push
requests / response
RPC - Remote Procedure call

server processing
- iterative
- concurrent: sub process for each request

transport
- reliable: streams, CO, atomic requests, dialogues
- unreliable: datagrams, CL, atomic requests

DNS - Domain Name System
hierarchic distributed database
IP ~ Name
divide et impera

Resource Record:
name | ttl | class | type | rdLength | rdata

class: IN - Internet
type: A, AAAA, NS, MX, CName

delegation
- iterative
- recursive
+ query

root → . → com, org → google

zonedata in Resource Records
13 root servers but anycast → 1600 instances
Redundancy for availability

WEB / HTTP — before Web: FTP, Emails, Gopher - for linking documents (1970s, 1980s, 1990s)
1989/90 at CERN
Web
HTTP 0.9 documented 1991
HTML
Web server
browser

~~HTTP - stateless~~
~~www~~ www.example.de / path / info.~~Bsp~~html
host name | path

base html file references objects

HTTP stateless but TCP holds state
client/server
traditionally client initiates
TCP mit ~~port~~ server port 80

HTTP
- ~~v1.0~~ / v1.0: nonpersistent — -overhead, RTT dominates
- v1.1: persistent — multiple objects over a single TCP connection
  - with pipelining: ersetzt ⇒ superseded by multiplexing HTTP/2
  - without pipelining: +efficient, -caused problems

91	0.9	HTML only
94	HTTPS	TLS
96	1.0	GET, POST, multimedia
97	HTTP/2	multiplexed stream via 1 TCP
15	1.1	virtual Hosting, security
22	HTTP/3	QUIC via UDP

IETF defines protocols z.B. HTTP
W3C defines Web standards z.B. HTML
Companies: create own protocols

Up to HTTP/1.1
ASCII

Request
GET <path> HTTP/1.1
Host: <host name>
User-agent: z.B. Mozilla/4.0
Connection: close
Accept-language: fr
+ Leerzeile indicates end

Response
HTTP/1.1 200 OK
connection close
Date:
Server:
Last-Modified:
Content-Length:
Content-Type: text/html

data data ...

301 Moved Permanently
400 Bad Request
404 Not Found
505 HTTP version not supported

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

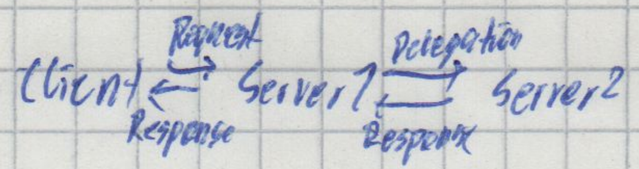
HTTP/3
QUIC
0-RTT (TLS handshake abbreviated)
always uses TLS

User-Server State = Cookies
components
- header line in request/response
- file managet by browser
- Back-end database

examples
- authorization
- shoping carts
- 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 => So still one requesting, one responding at an instance

no structure
no identifiers
only names
) can be introduced algorithmically

Applications
- Blockchain
- Video Streaming
- Dynamic load balancing
- Routing

traditionally File Sharing all peers server → highly scalable

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

Query Hit over reverse path

for scalability: limited scope flooding
only n hops