Computer Networks

Oversubscription

- More users then capacity
- Reservation:
  - Reserve bandwidth in advance
  - One link fails -> connection fails -> not dynamic
  - Waste bandwidth when $\frac{\text{Reserved Bandwidth}}{\text{Used Bandwidth}}$ is big
  - Used for phone-lines
  - Resource reservation protocol
    - Time based multiplexing } TV Radio Cisco
    - Frequency based multiplexing }
  - ↳ Circuit switching
  - Low latency
    - ↳ Because of buffers
- On demand:
  - Only when you need it
  - Used for Internet
  - ↳ Packet switching
  - ↳ Video is booming -> maybe reservation ?

Layers

- L5: Application (HTTP) (messages)
- L4: Transport (TCP/UDP) (segments)
- L3: Network (IP) (packets)
- L2: Link (Ethernet) (frames)
- L1: Physical (bits) (coax / twisted pair)

} Laptop
} Router
} Switch

Delay

- Transmission delay + propagation delay + processing delay + queing delay
  - Transmission delay: Packet size / Bandwidth
  - Propagation delay: Length of link / speed of light
  - Processing delay: tend to be tiny, depends on router, switch
  - ↳ 1 Gbit/s doesn't make a huge difference for tiny packets
    - ↳ Propagation delay dominates
      - -> Direct routes / better routing / CDN

DNS

- One name can point to several IP-addresses for redundancy and load balancing
- DNS uses UDP
- CNAME: aliaz to some other hostname
- PTR: reverse DNS-lookup
- Recursive query: · Offload query to DNS-server
  · Can't be done on root-servers
- Iterative query: · Each nameserver sends back only the information it knows
  · Not as fast as a recursive query
- Caching: · save resolution for TTL time on DNS-server

HTTP

- stateless → Need coockies
- Blocking: When browser has to wait for other objects to load because of dependencies
  ↳ Load time dependencies
  · Critical path analysis 1. Topological sort
  2. Max. Path
- Types of loadtime: · Last resource loaded
  · Last visual change (whole page)
  · Last visible change (only part I see)
- Enchant speed: · Compress
  · Use one file (inline in HTML)
  · Use multiple TCP-connections
  · Flatten dependency graph
  · small objects can be packed in one request

- Forward proxy: · Cache server at ISP
- Reverse proxy: · Cache server at hoster of the server
  ↳ · Pull cashing by user · No need to compute things again and again, just cache it (wordpress)
  · Push replication by high access expectation

Statistics

- $P(X = x) \sim Bin(X; \underbrace{P_{online}}_{\text{Online probability}}, \underbrace{N}_{\text{shared link by N users}})$

X: users online

Domain name vs. host name

- Domain name: · Registered
  · Used to reach network externally
  ↳ main-number of a company
- Host name: · name given to a particular machine
  ↳ Internal numbers inside a company

HTTP Post / Get

- Get: · Can be cached
  · In URL
  · In browser history
  · Requesting data
  · Never use it as authentification, because proxy can log it
- Post: · In body request
  · Can't be cached
  · More secure for authentification
  · No length restriction

Addressing vs. naming

- Addressing hosts via IP
- Naming hosts via domain names

Content delivery networks

- Overlay network: · To keep data in sync with all
  · Set up additional servers at the important internet nodes
  · Maintain persistant TCP connection between servers, because not everything is mirrored
- BGP-routing: · Advertise same IP from different locations
- DNS-routing: 1. Client requests to local DNS-server
  2. Local DNS server asks website-DNS-server and gets CNAME to CDN-DNS
  - DNS response depends on local DNS resolvers's guessed location
  - Not that good to use large DNS-services
- Most of the time: · HTML by own server
  · Media by CDN
- TTL-time is small to be able to switch server

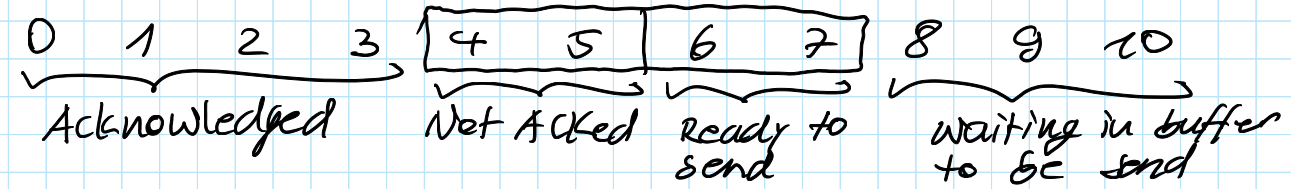
Video-streaming

1. Encode video in multiple bitrates
2. Replicate using a CDN-network
3. Video player picks bitrate according to bitrate-ladder which is computed by complexity analysis of video and buffer based adaption

Not good for startup! 1. Buffer full => higher bitrate
2. Buffer allmost empty -> low bitrate

Transport

- Correctness: Data is received by the application in the right order without gaps
- Timeliness: Deliver as fast as possible
- Efficiency: Don't send too many packets
- Timeliness vs. efficiency:
  - Small timer for ACKs → many retransmissions
  - Large timer for ACKs → slow transmission
- Sending window: min (Congestion Window; Receiver window) → Not overloading the network
- Receiving window: Flow control / Not overloading receiver
  - → Sending window ⊆ receiving window



Acknowledgement

- Individual ACK: Ack each packet → simple; No problem with reordering; Loss of Ack → retransmission
- Cummulative ACK: Ack the highest seq. number you received for which all previous packets have been received. → recovers from loss of Ack; confused by reordering
- Full information ACK: List all packets that have been received highest cumulative ACK, plus any additional packets → complete information; introduces overhead

Fairness

- Give users with small demands what they need and evenly distribute the rest
  - Max-min-fair-allocation
    1. Start with all flows at rate 0
    2. Increase flows until there is a bottleneck
    3. Hold fixed rate of flows that are bottlenecked
    4. Go to step 2 for remaining flows
- Implementation of max-min-fair-allocation:
- Progressively increase sending-window-size
- Whenever a loss is detected, decrease window size and repeat it
  - Causes warm-ups in downloading files

ACKing	full information ACK	TCP
retransmission	after timeout	
	after *k* subsequent ACKs	
window management	additive increase upon successful delivery	
	multiplicative decrease when timeouts	

Sliding-Windows-protocol

- Mechanism to detect loss and corruption
- Go-back-N
  - Retransmits all the frames that lie after the lost/corrupted frame
  - Use only one timer, reset at each ACK
  - Cumulative Ack

- Selective repeat
  - Only retransmits the missing frame
  - Uses per packet timer
  - Individual packet ack

UDP

- No handshake needed
- Avoids overheads for reliable delivery
- Not reliable
- Stateless
- Gaming, VoIP & DNS

TCP

- TCP sessions don't survive IP changes
- Earlier, sliding-window was the limiting factor, now it is the network itself

Congestion Control (=congestion Window)

- Network is bottleneck
- How much can I send without overflowing the router
- How can congestion be detected?
  - Measure packet delay → Varies ✓
  - Measure packet loss ✓ (already in TCP)
  - Network marker → needs router support ✓
- Duplicated Acks: Packets are still making it through
- Timeouted Acks: Dropping packets
- Slow start:
  - Start with window-size 1
  - Each ack you receive, increase by 1
  - Bandwidth estimation

- Multiple Increase/Decrease: cwnd = a * cwnd
- Additive Increase/Decrease: cwnd = a + cwnd } Bandwidth adaptation
- TCP: AIMD (=Additive Increase, multiple decrease)
- Threshhold for switching estimation => adaptation
- Fast retransmit: 3 same Acks -> AIMD (weak congestion ← Fast recovery
- Timeout: switch to slow start (Heavy congestion)

If the bandwidth is 20Mbps and the round trip time (rtt) is 40ms, the BDP is 20000000/8 * .04 = 100k. So 100kB is the maximum amount of data that can be in transit in the network at one time.

↳ Bandwidth delay product

↳ Jede 40ms werden 100kB von neuen Daten in die Leitung gegeben.

Socket vs. Port

- **Socket**:
  - Represents single connection between two network applications
  - OS Layer
  - Consists of:
    - IP address
    - Port
    - Protocol
  - Entry to an application
- **Port**:
  - "Channel" for network communication
  - Allow different applications on the same computer to utilize network resources without interfering
  - Network layer

- OS stores mapping between port <=> socket
- separate port space for TCP/UDP
- Well known ports for servers
- Ephemeral ports for clients

Best effort vs. reliable transport

- **Best effort:** ↳ Network Layer
  - Not in right order
  - Not correctly
  - Or even at all
    ↳ But tries to
  - + Lightweight implementation
  - + No three-way handshake
  - + Faster, suitable for "live" applications
  - + No setup time, start sending immediatly
  - − Dropped, Corrupted, Delayed, Duplicated, Reordered packets

  } UDP

  - Lightweight implementation
  - Can send immediatly
  - No handshake
  - Better performance

- **reliable transport:** ↳ Transport layer if needed
  - Correctness
  - Timeliness
  - Efficiency
  - Fairness

  } TCP

TCP 3-way-handshake

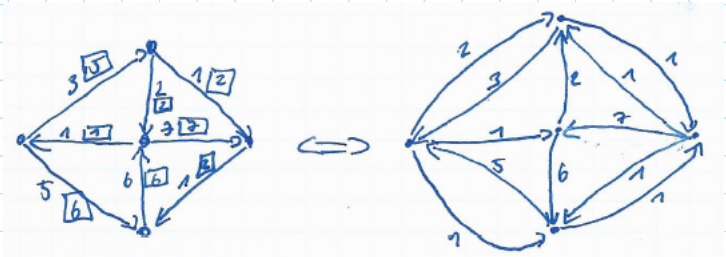
```
Alice ---> Bob    SYNchronize with my Initial Sequence Number of X
Alice <--- Bob    I received your syn, I ACKnowledge that I am ready for [X+1]
Alice <--- Bob    SYNchronize with my Initial Sequence Number of Y
Alice ---> Bob    I received your syn, I ACKnowledge that I am ready for [Y+1]
```

- Initial sequence number is choosen, because old packets with same IP and port could be still in transit

Algorithmen

Max-flow-min-cut

- Der grösstmöglichster Fluss von s->t ist gleich dem kleinstmöglichen Schnitt, da dieser das Maximum vorgibt
- Restnetzwerk:



- Ford-Fulkerson-Algorithmus: 1. Suche mit Restnetzwerk augmentierenden Pfad
  2. Mach das, solange es einen s->T Pfad im Restnetzwerk gibt
- Bipartide graph matching: 1. Add s and t
  2. Give all flows 1
  3. Do Ford-Fulkerson

Linear Programming

- Variables
- Objectives: Our goal
- Constraints: rules

Max-flow-problem

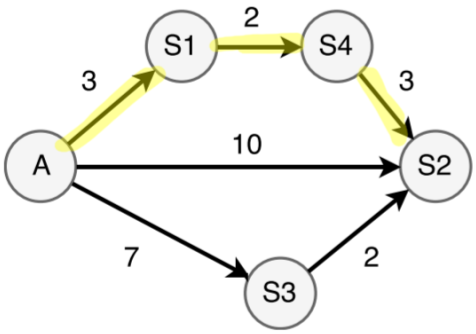
Flow start -> target: f
Flow on edge $u \to v$: f_{uv}

maximize f ← was rausfliesst, muss auch reinfliessen

$\Sigma_{u \to v} f_{uv} = \Sigma_{w \to u} f_{wu}$ (except at s,t)

$\Sigma_{s \to v} f_{sv} = f$
$\Sigma_{v \to t} f_{vt} = f$
} flow in = flow out

$f_{uv} \geq 0$ } No negative flows



- Shortest path
  - Goal: find shortest path A → sz
  - Variables: d_x (= distance to A)
  - Constraints: $-d_A = 0$
    $-d_{s1} \leq d_A + 3$ $-d_{s2} \leq d_A + 10$
    $-d_{s3} \leq d_A + 7$ $-d_{s2} \leq d_{s3} + 2$
    $-d_{s4} \leq d_{s1} + 2$ $-d_{s2} \leq d_{s4} + 3$

Load balancing

- Pick a server random for each request
- How often does imbalance occurs? (= Balls into bins)
  - Probability ball i and j land in same bin: $1/n$ ← servers
  - Expected number of collisions: $1/n$ ← m users
- Power of two: 1. pick two servers at random
  2. use the less used one
- Stick to one server per session
  - server = hash(user-id) % (n-1)

    ↳ Dadurch verraten wir nichts über unsere Infrastruktur

Logging object requests

- Log objects requested in a bloom filter
- Check filter: yes → cache it
  No → don't cache it, log it
- 80% of objects are accessed only once

Routing vs. forewarding

- Routing is process of deciding in which direction to send traffic
- Forewarding is sending packets to next hop

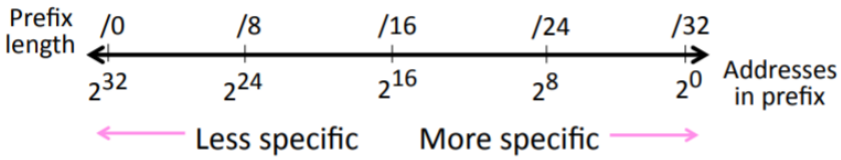
Datagram Model

- Each router has a forwarding table
- Gives next hop for each destination address

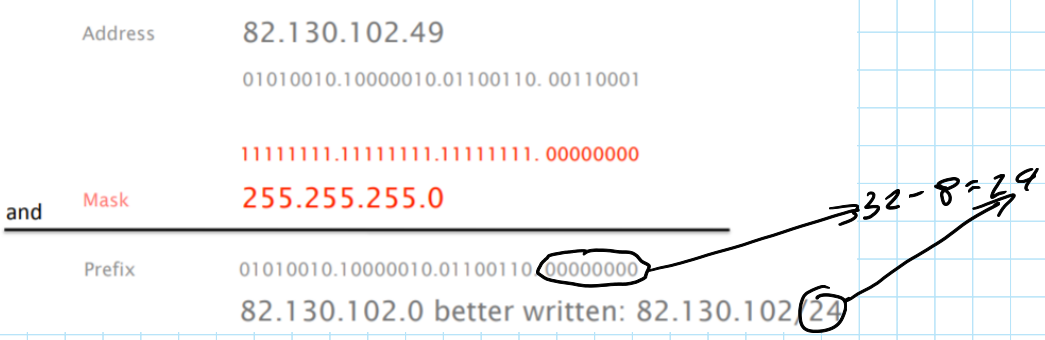
IPv4

- 32 bit addresses
- 8 Bit. 8 Bit. 8 Bit. 8 Bit

- 128.13.0.0/16 is 128.13.0.0 (Prefix | Host) to 128.13.255.255 (Prefix | Host)
- So a /24 is 256 addresses, and a /32 is one



- A /24 prefix means, that we have 32-24=8 bits left for host addresses
- In practice, the first/last IP address of a prefix is not used: Host part: 0000....0 ≙ Network identifier
  111....1 ≙ Broadcast address



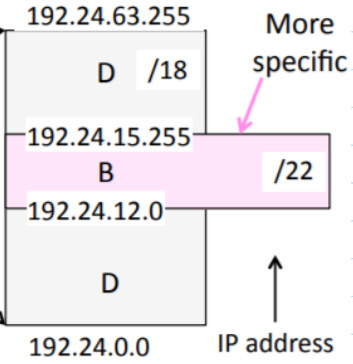
- 82.130.0.0/17 ≙ 10100.100000 10.1XXXXXXX.XXXXXXXX
- Addressable hosts: $2^{15}-2 = 32'766$
- Mask: 11111111.11111111.10000000.00000000
  255 . 255 . 128 . 0

- Network address: 82.130.0.0
- First host address: 82.130.0.1
- Last host address: 82.130.127.254
- Broadcast address: 82.130.127.255

Longest matching prefix

- For each packet, find the longest (most specific) prefix that contains the destination address

Prefix	Next Hop
192.24.0.0/18	D
192.24.12.0/22	B



192.24.6.0 → D
192.24.14.32 → B
192.24.54.0 → D

DHCP

- New computer sends broadcast messages to all other network devices in order to reach the DHCP-server

---

Packet size (MTU (= Maximal transmission unit))

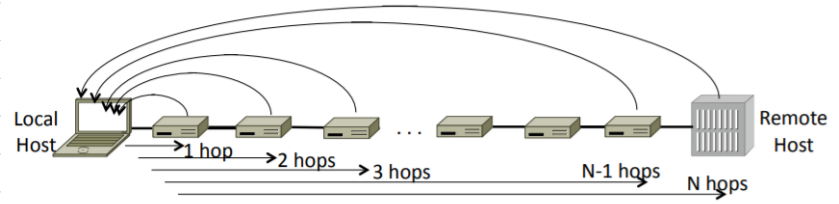
- The greater the size, the better
- Fragmentation:
  - Split up large packets at sending router
  - Reassemble if possible at receiving router
- Discovery:
  - Find largest packet that fits on path
  - IP uses this approach

- If packet is to big for some router it uses ICMP to report it

---

ICMP (=Internet Control Message Protocol)

- Used for Ping and Traceroute



- Traceroute:
  - IP has TTL field, starts for example at 30, and at every router it is decreased by 1. If TTL is 0, the router discards packet and informs sender via ICMP
  - If TTL wouldn't be used, packet would travel endlessly
  - Traceroute sets TTL first to 1, then 2, then 3, and so on, to get feedback from each node

---

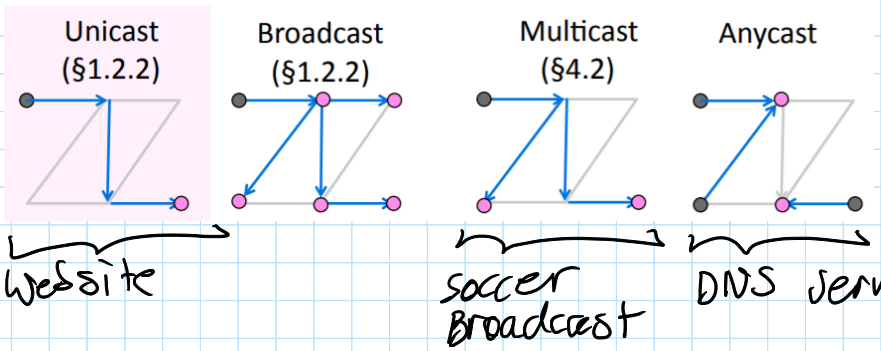
IPv6

- No checksum, because we have lower layer and higher layer checksum
- No fragmentation field in header
- Deployment:
  - IPv4 and IPv6 support
  - IPv6 packets in IPv4 packets
  - Translate IPv6 packets to IPv4

NAT

- Connect internal network to external (Lan → Wan)
- Many hosts are connected to only one external IP-address
- Has a translation-table
  ↳ Internal IP:port → external IP:port
- Difficult for UDP (no state is saved)
- Difficult if both ends are behind NAT (Skype)
  1. A wants to speak to B
  2. A sends a packet to B and waits for a response (B never gets this packet because of NAT)
  3. A notifies Rendez-vous-server with IP/port combination on which its waiting for a response
  4. Rendez-vous-server notifies B to send a response to A
  5. A receives response
  6. connection established

Delivery Models



Intra-domain routing

- Every node is alike (no controller)
- Nodes only know what they learn by exchanging messages with neighbours
- Sink trees: All shortest path from all starts to one particular destination

Distance vector routing

- Simple, but slow convergence
- Nodes only know the cost to their neighbours
- Vector is table with destinations and costs

1. Initialize vector with: 0 to myself
   ∞ to all other destinations
2. periodically send own vector to neighbours
3. Update vector for each destination by selecting the shortest route

- Use the best neighbour for forwarding
- Count-to-infinity-problem

Flooding

- Broadcast a message to all nodes in the network
- Inefficient

1. Send incoming message to all other neighbours
2. Remember the message so that you only send it once

- Inefficient, because one node may receive multiple copies of message
  - → To remember, use source-IP and sequence-number
    - → called duplicate suppression, done with Bloom filter

Link State Routing

- Each node forwards vector-table as in distance vector routing

1. Each node learns full topology
2. Each node runs shortest path locally

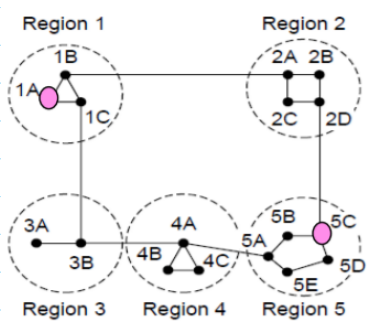
- Fast (flood and compute)
- Not as scalable as distance-vector-routing

Multipath routing

- Allow multiple routing path to destination
- Problem: More reordering because of different routes
- Solution: Use one route per TCP connection

Hierarchical routing

- Route first to the region, then to the IP prefix within the region
- Smaller vector-routing-tables
- But, longer paths
- Outside a region, nodes have one route to all hosts inside a region



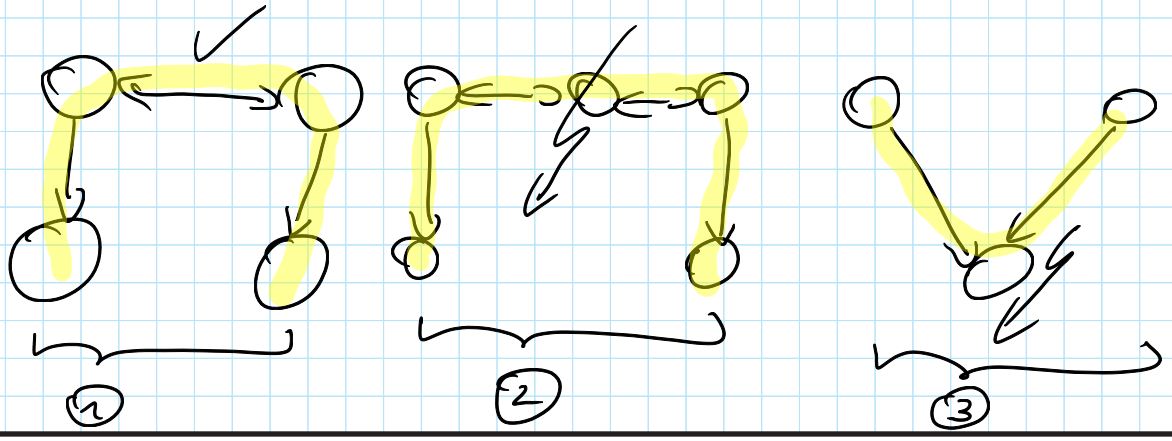
Subnets and Aggregation

- Subnet: 192.168.1.x and 192.168.2.x
- Aggregation: Join multiple prefixes into one

Inter-domain routing

- Network of Autonomous Systems (=AS)
- Between AS: BGP sessions
- BGP announcements carry complete AS-path instead of only the distance
- 2 ASes connect only, if the have a business relationship
  - Customer/provider: - Customer pay provider to get Internet
    - Amount paid is based on 95th percentile rule
      1. sorts all values
      2. removes top 5% values
      3. Bills highest value
  - Peer/Peer: - Don't pay each other

(1) Providers transit traffic for their costumers
(2) Peers do not transit traffic between each other
(3) Customer do not transit traffic between their provider



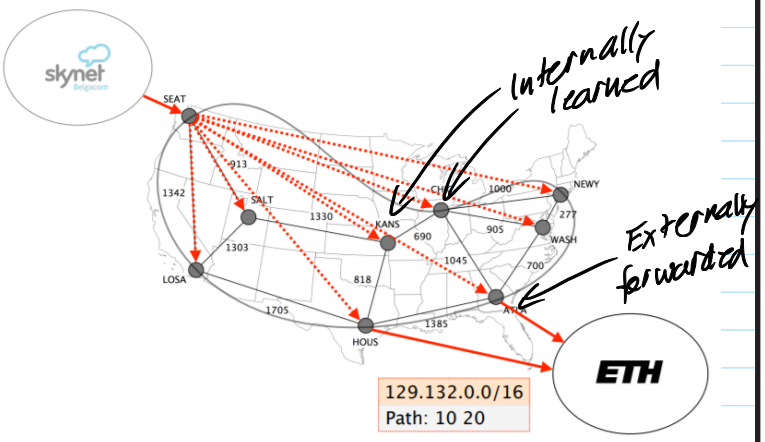
- For destination p, prefer routes coming from:
  1. customers over (have to pay you)
  2. peers over (you don't have to pay)
  3. providers (you have to pay)

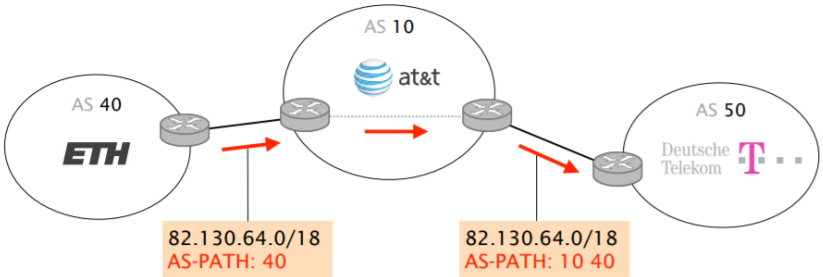
		send to: customer	peer	provider
from	customer	✓	✓	✓
	peer	✓	-	-
	provider	✓	-	-

Es kommt nicht drauf an, in welche Richtung Data fliesst, Customer zahlt in beide Richtungen

iBGP

- Used for internal network of provider
- iBGP sessions are used to disseminate externally-learned routes internally
- Routes disseminated internally are then announced externally again, using eBGP





- Local pref: · How preferred a route is
  · The higher the better
  · Strongest attribute where flow is routed
- MED: · Distance metric
  · How others should send traffic to you
- If two routes have same local-pref, you can decide with MED, on which route you want to receive data from that AS
- Prefer routes: 1. with higher local-pref
  2. shorter AS-path length
  3. lower MED
- Problems: · Because of policies, routing does not guarantee reachability, even if graph is connected

  · ASes can advertise any prefix
  · BGP may fail to converge

Link Layer

Framing methods

- How do we interpret the bitstream as a sequence of frames?
- Byte count: · Count each byte
  · Difficult after framing error
- Byte-stuffing: · Have a special flag byte value that means start/end of a frame
  · Flag inside content: Esc Flag
  · Esc inside content: Esc Esc
  ↳ insert escape code before
- Bit-stuffing: · Same as byte stuffing, but now with bits

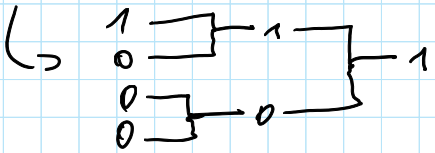
Error Handeling

- checksum: Detect corruption
- Correction Codes: Repair some of the corruption
- Hamming Distance: · Number of flipped bits needed to change $D+R_1$ to $D+R_2$
  · For code of Hamming distance $d+1$, d errors will be detected
  · For code of hamming distance $2d+1$, d errors can be corrected

- Hamming Distance 1: · No error detection
  · Send 1 ⇝ 0 received
- Hamming Distance 2: · Error detection possible
  · 11 ≙ 1 / 00 ≙ 0
  · Send 00 ⇝ 01 received
  ↳ We know, its an error
- Hamming Distance 3: · Error detection + repair possible
  · 000 ≙ 0 / 111 ≙ 1
  · send 000 ⇝ 011 received
  ↳ Repair to 111

Parity bit

- Take D data bits and suffix 1 check bit, that is the XOR of the D bits

↳ 

Internet Checksum

1 Hex ≙ 4 Bits · 4 = 16 Bits

Sending:

1. Arrange data in 16-bit words
2. Put zero in checksum position, add
3. Add any carryover back to get 16 bits
4. Negate (complement) to get sum

```
  0001
  f203
  f4f5
  f6f7
+(0000)
------
 2ddf0
   ↓
  ddf0
+    2
------
  ddf2
   ↓
  220d
```

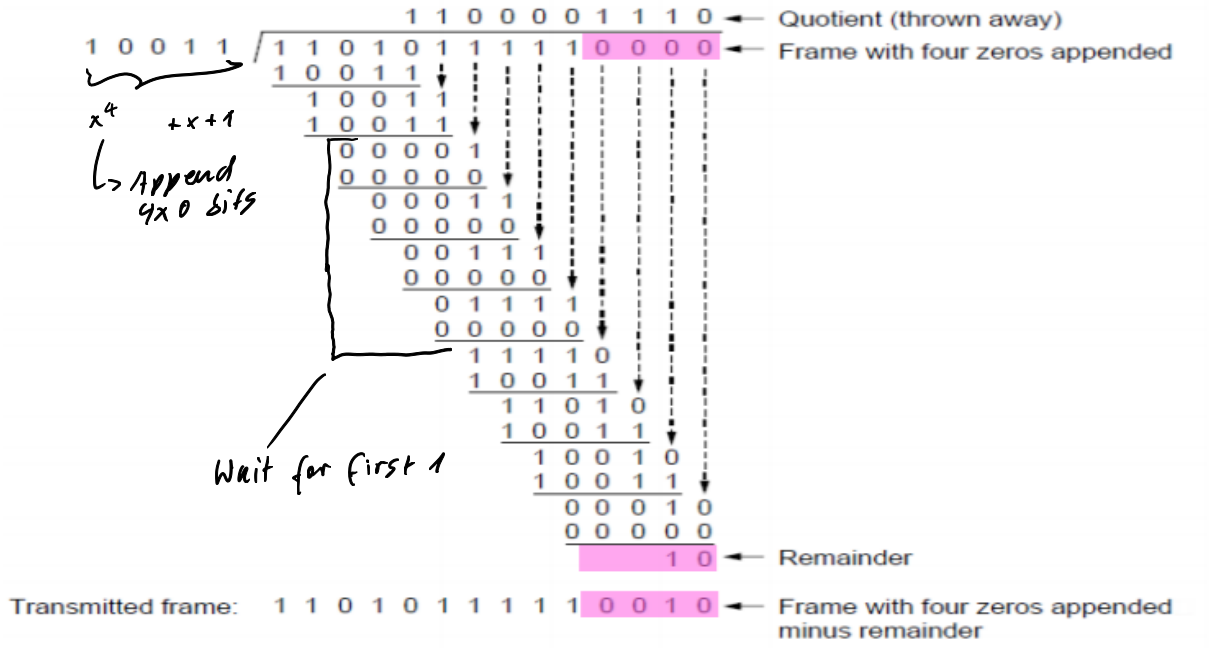
Receiving:

1. Arrange data in 16-bit words
2. Checksum will be non-zero, add
3. Add any carryover back to get 16 bits
4. Negate the result and check it is 0

```
  0001
  f203
  f4f5
  f6f7
+ 220d
------
 2fffd
   ↓
  fffd
+    2
------
  ffff
   ↓
  0000
```

CRC

- Works with polynomials



Error correction

Hamming code

data check message

- $n = 2^k - k - 1$ e.g. $n=4, k=3, N=7$
- Put check bits in positions p that are power of 2, starting at position one

7	6	5	4	3	2	1
1	0	1	p_3	0	p_2	p_1

- Example: data = 1010, 3 check bits =>
  - 7 bit code, check bits at position $2^0=1 / 2^1=2 / 2^2=4$
  - Check 1 covers bit positions xx1 ≙ 1, 3, 5, 7
  - Check 2 covers bit position x1x ≙ 2, 3, 6, 7
  - Check 3 covers bit positions 1xx ≙ 4, 5, 6, 7

- $p_1 = 0+1+1 = 0$
- $p_2 = 0+0+1 = 1$
- $p_3 = 1+0+1 = 0$

} 1010010

- Check: 1010010 : $p_1 = 0+0+1+1 = 0$
  $p_2 = 1+0+0+1 = 0$
  $p_3 = 0+1+0+1 = 0$

} 000 => no error

- Check: 1~~1~~10010 : $p_1 = 0+0+1+1 = 0$
  $p_2 = 1+0+1+1 = 1$
  $p_3 = 0+1+1+1 = 1$

} 110
↳ flip position 6

Parameterkombinationen bei Hamming-Codes		
n	***k***	***N* = n + k**
Datenbits (Datenwort)	**Paritätsbits (Kontrollstellen)**	**Nachrichtenbits (Gesamtlänge des Codewortes)**
1	2	3
4	3	7
11	4	15
26	5	31
57	6	63
120	7	127
247	8	255

} If code is too short add 0 on left side, where number-index increases

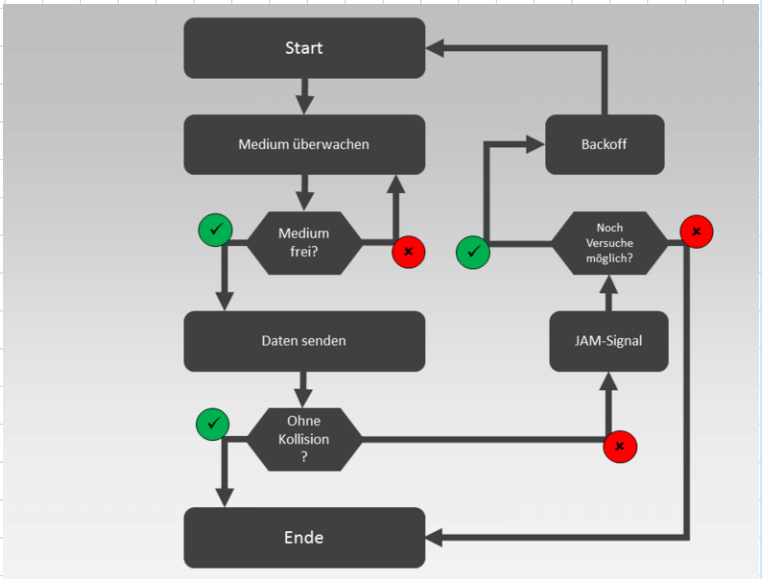
Multiple Access Protocol

- How users share a connection

Randomized Multiple Access

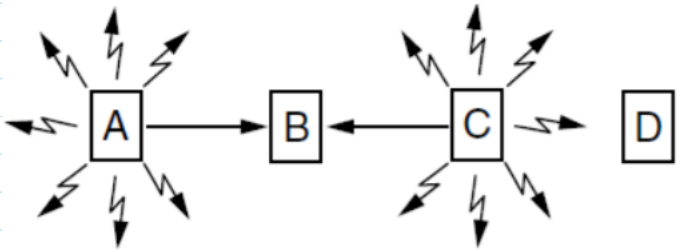
- ALOHA protocol: If there was a collision (no Ack received) wait a random time
- Carrier Sense Multiple Access (=CSMA)
  - ALOHA with listening before sending
  - Still possible to listen and hear nothing, when another node is sending because of delay
- Carrier Sense Multiple Access with collision detection
  - (=CSMA\CD)
  - When station is sending, it continously monitor transmission if it is successful or collision happened
  - If collision is detected station sends jamming signal to inform other station, that their sending is also corrupted

$2 \cdot t_{prop}$



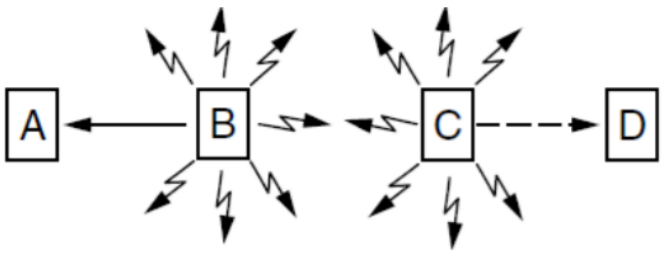
Wireless Multiple Access

- Nodes can't hear while sending
- Hidden terminal:



  - A and C can't hear each other while sending to B
  - Avoid that => collisions

- Exposed terminal:



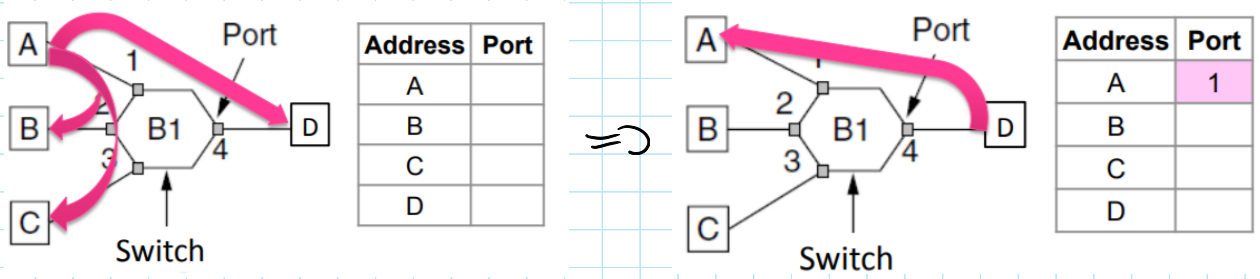
  - B and C can hear each other but don't collide at receiver A/D
  - We want that to send concurrently
  - MACA uses short handshake
    1. Sender sends ready to send
    2. Receiver replies clear to send
    3. Sender transmits to receiver while nodes hearing the CTS stay silent
    4. Senders can send

Switches

- Switch needs to find right output port for destination address

↳ **Backward Learning**

1. If switch doesn't know destination, it broadcasts packet and wait for response
2. Now it knows in the future, on what port destination is



- **Backward learning with multiple switches**
  - Just works with no loop
  - But if there is a loop

  ↳ **spanning tree solution**

  1. Elect a root node of tree
     ↳ switch with lowest address
  2. Grow tree as shortest distance from root (Same distance → lower address)
  3. Turn off port if they are not in spanning tree
     - All switches run same algorithm and initially think they are the root
     - Exchange messages to figure out the root

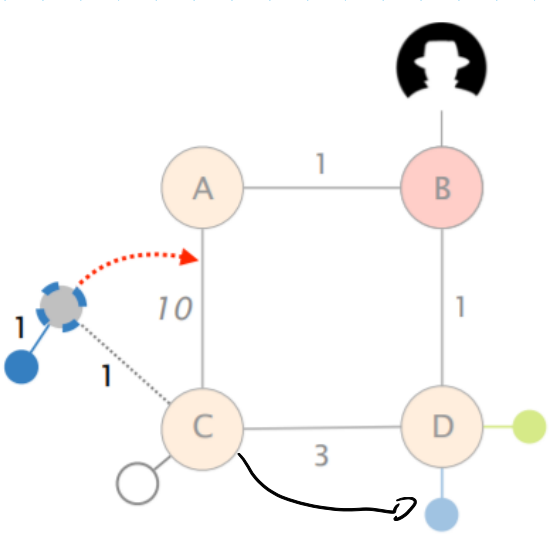
- If we flood the finite routing table, the switch is going to repeater mode

QUIC - protocol

- Encapsulated in UDP
- Security built in
- Makes crypto handshake more efficient
- Can send encrypted payload data before the handshake completes on resumption to previously connected server
- Pipelining allows multiple requests to be sent at once
- Has built in connection_id
- Encrypted headers

Security

- Asymmetric
  - Public-private key
  - Used to establish a symmetric connection
  - slow
- Eavesdrop (intradomain)



1. Attacker wants to eavesdrop blue connection
2. Insert fake node that says, that C is in reach through fake node with length 2 to blue node
3. But sender sends traffic to A through fake node

- BGP security (Intra)
  - BGP does not validate origin of advertisement
  - Prefix hijacking: say that you are connected to that prefix
  - Just announce more-specific prefix

→ Secure BGP

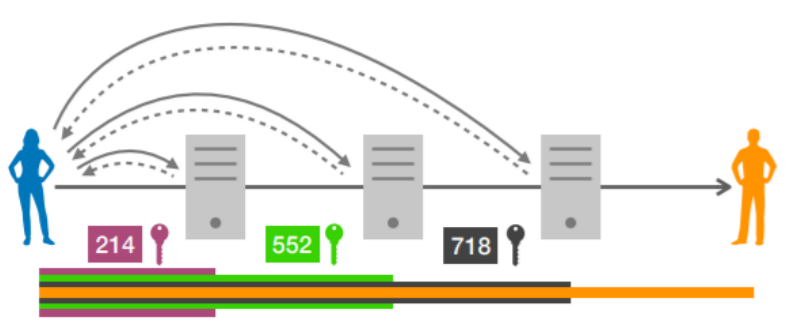
- Origin authentification + signatures

- Data plane attacks (=Data plane => how to route)
  - Buy compromised router online
  - Detect via longer Traceroute

Scion

Anonymous communication

- Certificate: · Trusted party can certify Bob's public key
- Tor: · Telescopic setup:

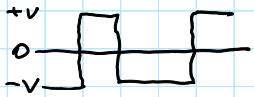


· Passive traffic analysis: observe the edges of the network

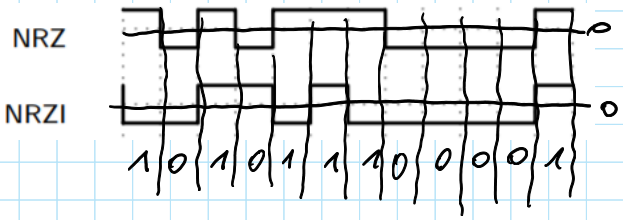
· Active traffic analysis: Inject marking-bit

Baseband-Modulation

- Non-return-to-zero: +V ≙ 1 / -V ≙ 0



- Non-return-to-zero-inverted: Invert signal-level on a 1 to break up long runs of 1s



- Clock recovery: -How can we know, how many 0s are contained in a 0-run
  - 4B/5B: - Map every 4 data-bits to 5 code bits without long runs of zeros
    · Has at most 3x0 in a row
    1. Decode data-bits -> code-bits
    2. Use NRZI to get it on the wire