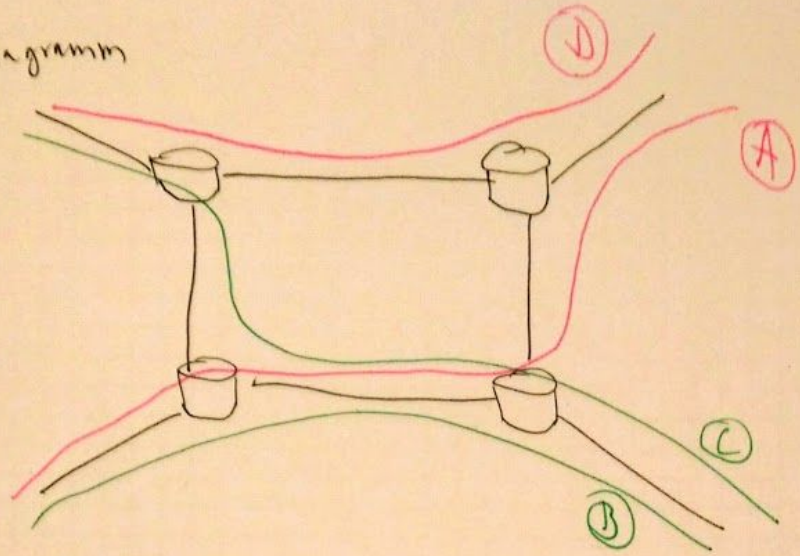
Network Protocols and Architectures

Prüfungsinhalte Klausur am 24.02.17

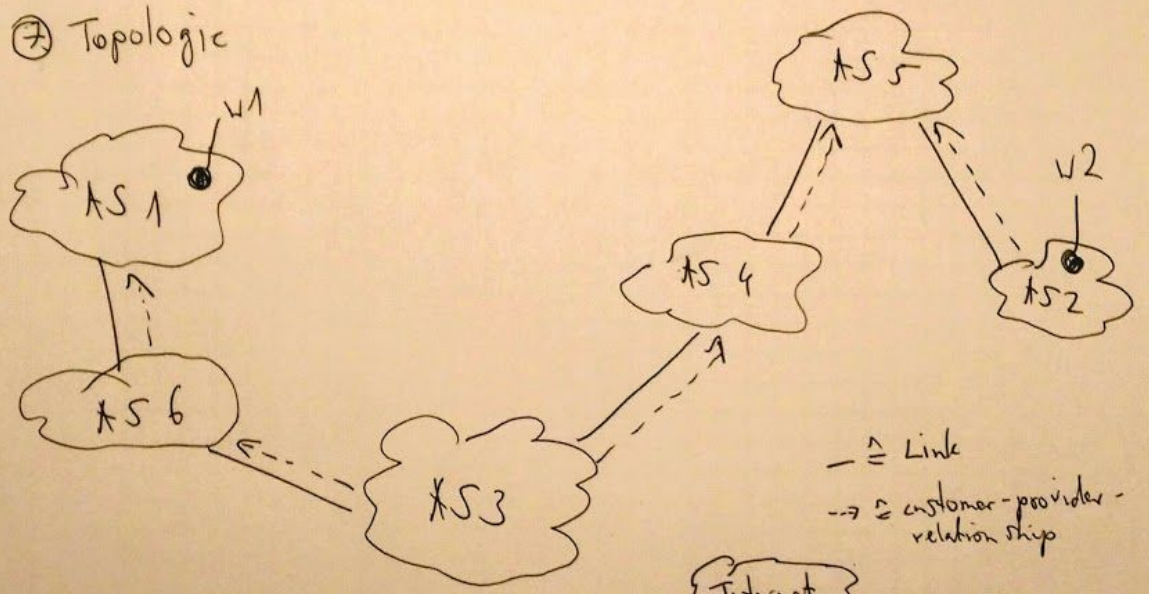
insgesamt 100 Punkte, 50 reichen für 4,0 und 90 reichen für 1,0

1. Aufgabe Multiple Choice 20 Punkte
   war eigentlich kein Problem, wenn man alle Folien mal durchgelesen hat
2. Aufgabe DNS
   a) eine DNS-Anfrage für www.tu-berlin.de in eine Tabelle eintragen mit src-IP, dst-IP und Anfrage (es war eine Tabelle mit Domains und IP-Adressen gegeben)
   b) DNS-Anfrage für mail-box.tu-berlin.de auch in so eine Tabelle eintragen; wichtig ist hier, dass frühere Anfragen schon gecacht sind
   c) erklären, was Authoritative Entry und Additional Entry ist (keine Ahnung xD)
   d) erklären, warum man manchmal unterschiedliche DNS Antworten bekommt (wegen load-balancing)
3. Aufgabe TCP
   Sequenznummern und ACKs in Diagramm eintragen und dazu Fragen beantworten wie: an welcher Stelle wurden 3 duplicate ACKs empfangen? Warum wird das Fenster halbiert? usw. (genau wie im Tutorium TCP-Reno)
4. Aufgabe TCP Fairness
   a) Was passiert wenn mehrere TCP-Verbindungen A,B,C,D über eine Leitung gehen? Diagramm dazu war gegeben. (siehe Bild)
   b) Was passiert, wenn die TCP-Verbindung D zu UDP wird?
5. Aufgabe TCP cwnd-Zeit-Diagramm vervollständigen für TCP Reno 7 Punkte
6. Aufgabe: Netzwerktopologie gegeben (siehe Bild) je 4 Punkte, also 20 insgesamt
   MAC-Adressen eintragen, IP-Adressen eintragen
   Subnetze angeben
   Forwarding-Tabelle für 2 Router aufstellen
   Tabelle mit Übertragungen aufstellen für eine SSH-Verbindung, also zwischen allen Subnetzen ARP-Nachrichten und dann TCP-Austausch
7. Aufgabe BGP (ich hatte kein Plan, was da hin soll ^^) 15 Punkte
   Topologie war gegeben (siehe Bild)
   es solle von AS3 zu dem Webserver w1 oder w2 eine Verbindung aufgebaut werden
   a) AS Pfade aufstellen, Würde ein Client c aus AS3 einen Webserver erreichen?
   b) 2 der ASs (AS3 und AS6) bilden Peer-to-Peer-Verbindungen, was würde sich ändern? Würde ein Client c aus AS3 einen Webserver erreichen?
   c) ein Link (AS3 und AS4) wird unterbrochen, AS Pfade aufstellen und Würde ein Client c aus AS3 einen Webserver erreichen?

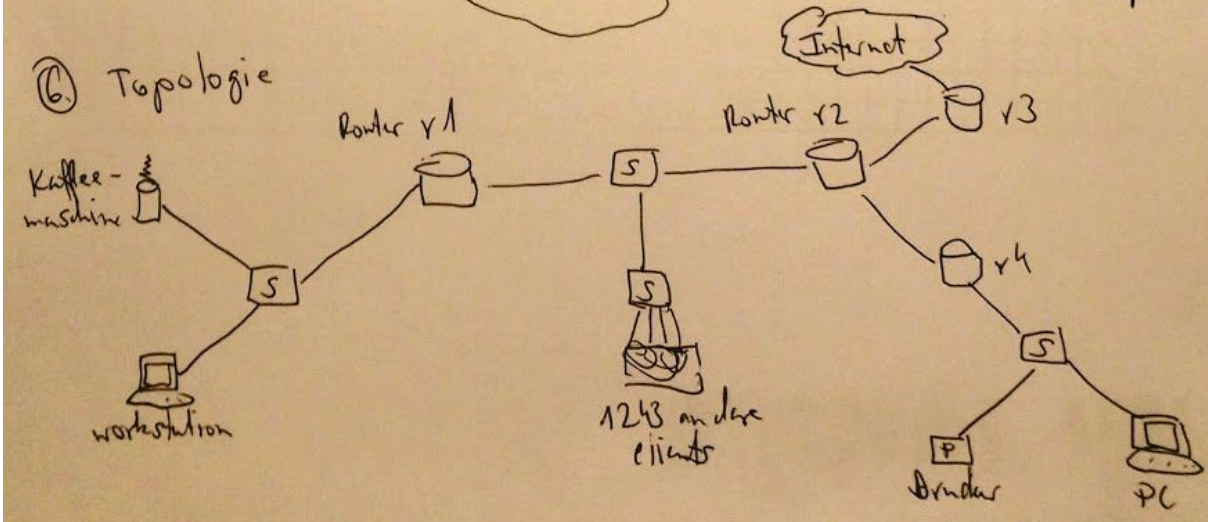
⑤ Diagramm



⑦ Topologie



⑥ Topologie



wichtig für die Prüfung

- HTTP
- DNS
- TCP/UDP
- IP
- MAC-Protokolle (ARP,...)
- Routing-Protokolle: BGP
- andere Themen nur passiv relevant für multiple choice

- 10 Multiple-Choice Fragen mit je 2 Punkten
- praktische Anwendungen zum Netzwerk
- immer öffentliche statt private IP-Netze verwenden (keine 192.168.x.x, stattdessen 200.x.x.x)
- für Host-Router Verbindungen immer /30 Netzwerkmaske verwenden
  für Transit-Verbindungen (Router-Router) immer /31
- TCP-Verbindungen im Detail kennen, UDP nicht so wichtig
- Forwarding Table Header besteht aus Prefix, next hop, interface
- Unterschied zwischen TCP Reno und Tahoe kennen
- Unterschied zwischen Hard- & Softstate ist wichtig
  damit ist immer die Transaktion gemeint, nicht der Sate selbst
- Wenn alle Netzwerk-caches leer sind, zuerst ARP, dann DNS, IP, TCP/UDP
- Was ist anycast? - Anfrage an mehrere, Antwort von Host mit kürzester Route
- DNS-Anfragen an lokalen Router immer rekursiv, Typ A
  DNS-Anfragen an root- und TLD-Server immer iterativ, Typ NS
  DNS-Anfragen an Domain-Server sind iterativ, Typ, A

Hypertext Transfer Protocol (HTTP)

- Anwendungschicht-Protokoll
- Client-Server, asynchron, über TCP, Port 80, statuslos, Request/Response
- Request-Methoden: GET, HEAD, POST, PUT, Delete, Trace, Connect
- non-persistente Verbindung: ein Objekt pro TCP-Verbindung
- persistente Verbindung: mehrere Objekte in einer TCP-Verbindung (HTTP 1.0)
- Pipelined persistente Verbindung: mehrere Anfragen ohne auf Antwort zu warten, Reihenfolge der Antworten wie Anfragen (HTTP 1.1)
- Persistente Verbindung mit out-of-order Lieferung: Mehrere Anfragen, Server kann in abweichender Reihenfolge antworten und/oder zusätzliche Objekte schicken
- Cookies: Autorisierung, Warenkorb, Empfehlungen, Session State, Tracking

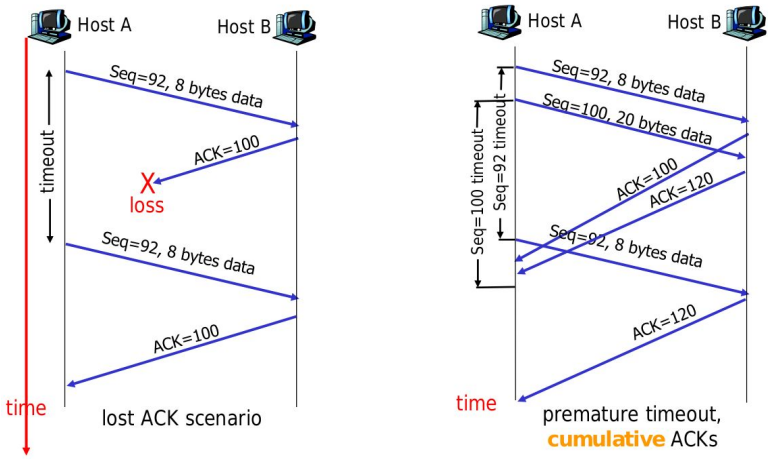
DNS

- Applikationsschicht-Protokoll
- verteilt, hierarchisch, UDP, Port 53
- Lokaler Name-Server: jeder ISP, Firma hat einen, wie ein Proxy
- Authoritative Name-Server: für einen best. Hostname, z.B. eine Firmendomain
- Root Name-Server: 13 weltweit, per anycast
- Typ A für Hostname und IP-Adresse
- Typ NS für Autoritative Name Server
- Typ CNAME für Aliase (CDNs)
- Typ MX für Mail
- Typ AAAA für IPv6

Transmission Control Protocol (TCP)

- verbindungsorientiert: Handshakes vor Datenaustausch
- Point-to-Point: ein Sender, ein Empfänger
- zuverlässiger, in richtiger Reihenfolge Bytestream
- Pipelined: TCP Fluss- und Staukontrolle bestimmen die Fenstergröße
- Volle Duplex Daten: bidirektionaler Datenfluss in 1 Verbindung
- Flusskontrolle: Sender wird den Empfänger nicht überfluten
- Staukontrolle: Sender wird das Netzwerk nicht überfluten
- Sequenznummer: Byte Stream Nummer des 1. Bytes der Segmentdaten
- ACKs: Seq.nr. des nächsten zu erwartenden Bytes von der anderen Seite
- Paketverlusterkennung beim Sender: mit Retransmission Timeout
  fast retransmit: 3 doppelte ACKs
- Retransmission Timeout: je 1 Timer für 1 Paket
  wird dynamisch berechnet auf Grundlage der Round Trip Time
  Neue RRT = alpha (alte RTT) + (1-alpha) alpha meistens = 0,875

TCP: Retransmission Scenarios



https://hpbn.co/building-blocks-of-tcp/

TCP Flusskontrolle: Sliding Window Protocol

- Flusskontrolle: Sender soll Buffer des Empfängers nicht durch zu viel und zu schnelles Übertragen überschreiten
- Empfänger informiert Sender über freien Buffer-Speicher
- sliding window: bei einer Fenstergröße n können n bytes gesendet werden ohne eine Bestätigung zu erhalten; wenn Daten bestätigt wurden, wird das Fenster verschoben
- ideale Fenstergröße = delay * bandwidth = RTT * bottleneck bitrate

3-Wege-Handshake

1. SYN von Client zum Server
   - bestimmt initiale Sequenznummer und Fenstergröße
2. Antwort mit SYNACK
   - bestätigt SYN mit ACK, teilt Buffer zu
   - bestimmt ebenfalls Sequenznummer und Fenstergröße für den Rückweg
3. Client emfängt SYNACK und sendet Daten mit ACK

Verbindungsende

1. Client sendet TCP FIN control segment an den Server
2. Server antwortet mit ACK, sendet auch FIN zurück
3. Client antwortet mit ACK
   - timed wait: Timer bei der der Empfänger mit ACKs auf FINs antwortet
4. Server empfängt ACK → Verbindung geschlossen

TCP Staukontrolle

- vermeiden von verlorenen Paketen (Buffer overflow in Routern)
- vermeiden von langen delays (Queueing in Router Buffern)
- Variablen:
  - cwnd: Anzahl bestätigter Pakete
  - threshold: Grenze zwischen 2 slow start Phasen, Congestion Control Phase
- Slow Start
  - exponentielle Steigerung nach jeder RTT
  - bei Timeout (Tahoe TCP) wird threshold auf cwnd/2 gesetzt
  - oder 3 ACK-Duplikate (Reno TCP) wird threshold auch auf cwnd/2 gesetzt
    fast retransmit: senden des fehlenden Segments
    fast recovery: benutzen von congestion avoidance
- Congestion avoidance
  - nach Slow Start und erstem Paketverlust linearer Anstieg
  - nach 2. Paketverlust threshold=cwnd/2 und cwnd=1

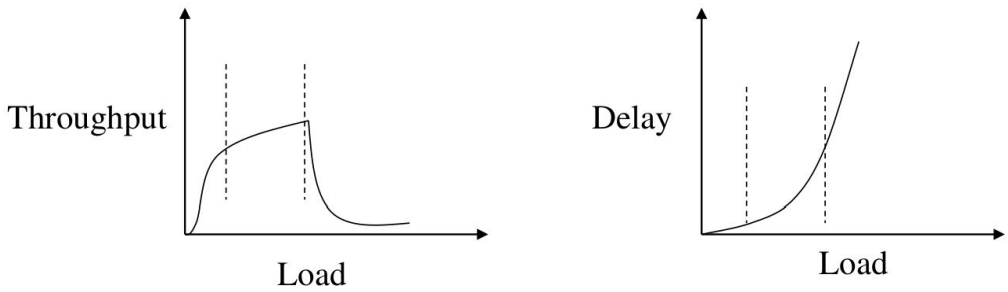
TCP Varianten

- TCP Tahoe
  - slow start
  - congestion avoidance
  - bei Timeout, 3 doppelte ACKS → cwnd = 1 und slow start
- TCP Reno

- slow start
- congestion avoidance
- fast retransmit, fast recovery
- bei Timeout → cwnd = 1 und slow start
- 3 doppelte ACKS → fast recovery und congestion avoidance

Congestion collapse

- Def: Steigerung der Netzwerklast hat zur Folge, dass nützliche Arbeit abnimmt



End-end congestion control: kein explizites Feedback des Netzwerks, wie bei TCP
Network-assisted congestion control: Router geben den Endsystemen Feedback
Additive increase, multiple decrease (AIMD)

- Ansatz: Steigerung der Übertragungsrate, testen der Bandbreite bis ein Verlust auftritt
- Additive increase: steigern von cwnd um 1 jede RTT, bis Verlust entdeckt wird
- Multiplicative decrease: halbieren von cwnd nach Verlust

TCP Fairness

- Fairness-Ziel: wenn n TCP-Sitzungen die gleiche Engpass-Verbindung benutzen, sollten sie 1/n der Verbindungskapazität bekommen
- im Idealfall ist es fair, da “additive increase” sich um 1 steigert, wenn sich auch der Durchsatz steigert und “multiplicative decrease” nimmt mit dem Durchsatz proportional ab

User Datagram Protocol (UDP)

- Best effort Service
- verbindungslos: kein Handshake, jedes Segment wird unabhängig gehandhabt
- UDP-Segmente können verloren gehen, in anderer Reihenfolge ankommen
- zuverlässiger Transfer über UPD muss auf Applikationsschicht hinzugefügt werden
- Prüfsumme: Einer-Komplement in 16 bit Wörtern, Daten + 12 byte header

Routing

- 3 wichtige Funktionen der Netzwerkschicht:
  - Adressierung
  - Pfadbestimmung mit Routing-Algorithmen
  - Switching/forwarding

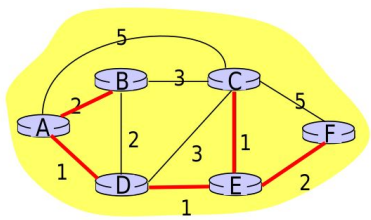
- Routing-Algorithmen
  - global: alle Router haben die komplette Topologie und Verbindungskosten
    → Link state Algorithmen
  - dezentral: Router kennt physikalisch verbundene Nachbarn und deren Verbindungskosten
    iterativer Berechnungsprozess, Austausch von Infos mit Nachbarknoten
    → Distance vector Algorithmen
  - statisch: Routen ändern sich langsam mit der Zeit
  - dynamisch: Routen ändern sich schneller - durch periodische Updates oder zusammen mit Änderungen der Verbindungskosten

Dijkstra's Algorithmus

- ist ein link state Routingalgorithmus
- Netzwerktopologie und Verbindungskosten sind allen Knoten bekannt
  über link state broadcast verbreitet, alle Knoten haben die selben Infos
- berechnet die kleinsten Kostenpfade von einem Knoten zu allen anderen
  es entsteht eine Routingtabelle für den Knoten
- iterativ: nach k Iterationen sind kleinste Kostenpfade zu k Zielen bekannt

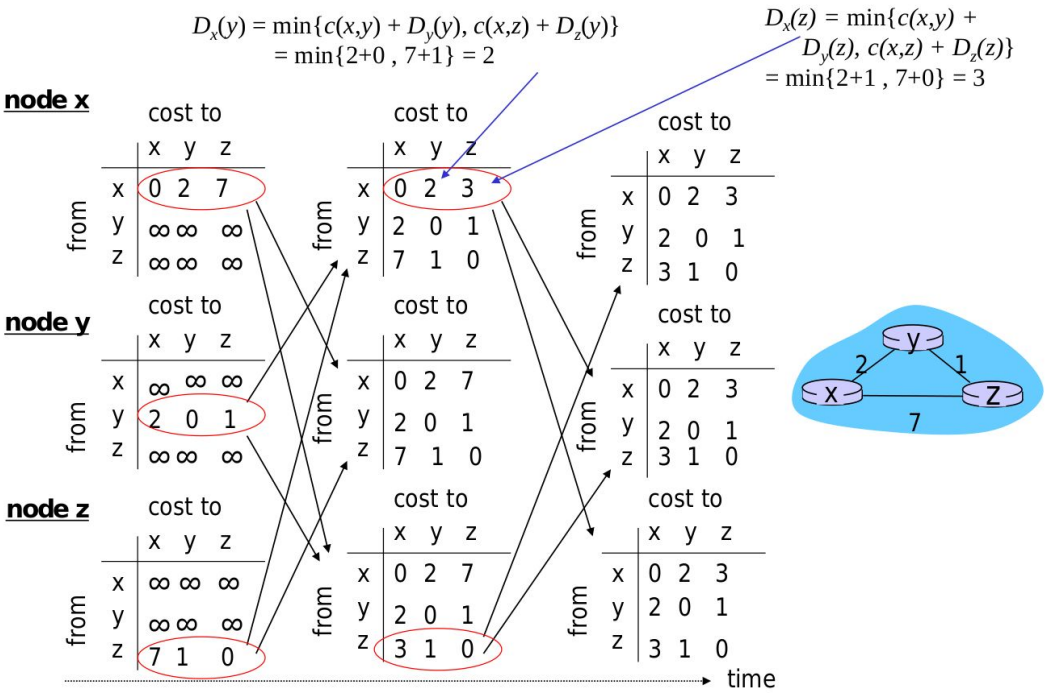
Dijkstra's algorithm: Example

Step	start N'	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					



Bellman-Ford Algorithmus

- ist ein Distance vector Algorithmus
- B-F Gleichung für dynamisches Programmieren: d(y) = min{c(x,v) + d(y)}
- jeder Knoten sendet periodisch seinen Distanzvektor an seine Nachbarn
- wenn ein Knoten einen neuen Distanzvektor von seinen Nachbarn bekommt, updatet er seinen eigenen Distanzvektor mit der B-F Gleichung
- unter natürlichen Bedingungen konvergiert dieser Vektor zu den aktuellen kleinsten Kosten
- jeder Knoten bemerkt seine Nachbarn nur, wenn sich sein Distanzvektor ändert



Internet Routing

- Internet = Netz der Netze
- jeder Admin will das Routing in seinem eigenen Netzwerk steuern
- Router werden in Regionen eingeteilt → “Autonome Systeme” (AS)
- so gibt es unterschiedliche intra-AS Routingprotokolle

Intra-AS Routing

- Interior Gateway Protocols (IGP)
- Routing Information Protocol (RIP)
- OSPF: Open Shortest Path First
    - Link State Algorithmus
    - state: jeder Router hat Infos über seine erreichbaren Netzwerke
    - OSPF advertisements: state verbreiten, Datenbank-Synchronisierung
    - link state Datenbank: states von alles Routern
    - Hello-Protokoll: Nachbarn finden
    - hierarchisches OSPF in großen Umgebungen
      1 boundary router < n backbone routers < m area border routers < internal routers

Inter-AS Routing

- de facto Standard: Border Gateway Protocol (BGP)
- Routerpaare (BGP peers) tauschen Routinginformationen über semi-permanente TCP-Verbindungen aus: BGP sessions
- BGP ist ein Pfad-Vektor-Protokoll
    - Attribut AS-PATH - enthält alle ASs für eine Route

    - Attribut NEXT-HOP - interner AS Router zu next-hop AS
    - Prefix + Attribute = Route
    - wenn Gateway-Router eine Route-Vorschlag bekommt, benutzt er Ingress-filter zum Bestätigen oder Ablehnen (z.b. um Schleifen zu verhindern)
- Routing policy entspricht den Zielen des Netzwerk-Anbieters (kosteneffizient)
- lokales Preferenz-Attribut: Anbieter bevorzugen Routen mit höchster Präferenz
- Auswahl der Route:
    - lokales Preferenz-Attribut
    - kürzester AS-PATH
    - bester MED (multi-exit-disciminator) (???)
    - nahester NEXT-HOP Router - hot potato routing
    - IP-Adresse des Peers

Internet Protocols IPv4/IPv6

- IP Adresse: Identifier für Host oder Router Interface
    IPv4: 32 bit dezimal IPv6: 128 bit hexadezimal
- Interface: verbindet einen Host oder Router to einen physikalischer Link
- Netzwerk: physikalischer Erreichbarkeit ohne dazwischenliegenden Router
- Unterschied in IPv6: meistens mehr als 1 Adresse pro Host, spez. link-local Netzw.
- Classless InterDomain Routing (CIDR)
    - bitweise aufteilung von Netz/Subnetz-Teil und Host-Teil
    - Adressenformat a.b.c.d/x
- spezielle Adressen für IPv4
    - Loopback 127.0.0.0/8
    - Multicast 224.0.0.0/4 (224.0.0.0 bis 239.255.255.255)
    - Private Netze 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
    - Link-Local 169.254.0.1/16
- spezielle Adressen für IPv6
    - Loopback ::1/128
    - Global Unicast 2000::/3
    - Unique Local FC00::/7 (private Adressen)
    - Multicast FE00::/8
    - Link-local Unicast FE80::/10
    - Adressen für die Nutzung im Internet sind global Unicast und Teile von Multicast
- Multicast - adressieren einer Gruppe von Hosts mit einer Nachricht
- link-local Adressen: nicht-routbare Adressen
- DHCP: Dynamic Host Configuration Protocol - Adresse von einem Server anfragen
- IPv6 SLAAC: Stateless Address Auto-Configuration - Router geben IPv6-Prefix bekannt und Hosts fügen Interface-ID als Hostteil hinzu

Forwarding:

- Prozess, bei dem Pakete mithilfe einer Forwarding-Tabelle von Input zum Output weitergeleitet werden
- die Information ist im Paket
- Forwarding Entscheidung: longest prefix matching

Routing:
- Prozess, bei dem eine Forwarding-Tabelle aufgebaut und gewartet wird
- 1-n Routing-Protokolle

NAT - Network address translation
- eine IP-Adresse für alle Geräte eines lokalen Netzwerks
- lokale Netzwerkgeräte nicht direkt von außen adressierbar
- in NAT translation table wird die Zuordnung von WAN- und LAN-Adressen und Ports notiert

MAC-Protokolle

Data link layer beinhaltet
- Fehlererkennung, -korrektur
- broadcast channel: multiple Zugriffe
- link layer Adressierung
- Half-duplex und full-duplex

MAC-Protokolle
- Channel Partitionierung z.B. Zeitfenster, Frequenz
- Zufälliger Zugriff: Kollisionen erlauben und wiederherstellen
- Taking Turns: gemeinsamen Zugriff koordinieren um Kollisionen zu vermeiden
- → Ziel: effizient, fair, einfach, dezentralisiert

Adressen
- IP-Adressen
    - Netzwerkschicht-Adressen
    - genutzt, um Daten zu dem Zielnetzwerk zu bringen
- MAC/LAN/physikalisch/Ethernet - Adressen
    - Data link layer Adressen
    - genutzt, um Daten von einem physikalisch-verbundenen Interface zu einem anderen zu übermitteln (gleiches Netzwerk)
    - 48 bit MAC Adresse: eingebrannt in NIC ROM

Ethernet
- dominante LAN Technologie
- benutzt CSMA/CD (Carrier Sense Multiple Access / Collision Detection)
- bei Kollision: Jam Signal, 48 bit, somit wissen alle Sender Bescheid
- exponential backoff: Zeit zu Warten nach einer Kollision
    - 1. Kollision: wähle k zufällig aus {0,1}, Delay ist k*512 bit
    - 2.-9.Kollision: wähle k zufällig aus {0,1,2,3}
    - ab 10. Kollision: wähle k zufällig aus {0,1,2,3,4,...,1023}

Address Resolution Protocol (ARP)

- jeder IPv4-Knoten im LAM hat eine ARP-Tabelle, die die Mappings von IP zu MAC-Adresse für einige LAN-Knoten enthält
- ARP Tabelle: IP-Adresse, MAC-Adresse, TTL

Indirection

Def: anstatt eine Entity direkt zuzuweisen, wird sie indirekt über eine andere Entity, die auf die originale Entity zugreifen kann, zugewiesen
“Every problem in CS can be solved by adding another level of indirection.”

Multicast

- Senden von Daten zu mehreren Empfängern mit einer Operation
- Multicast Gruppe: D-Klasse-IP Adresse reserviert für Multicast
- Internet Group Management Protocol (IGMP): lokale Multicast-Gruppen
- auch wide area multicasting mit DVMRP, MOSPF, PIM,...
- zuverlässiger Multicast mit ACK-Tabelle beim Sender
  Paket wird erst aus dem Buffer entfernt, wenn alle ACK in der Tabelle sind
  selective repeat bei Verlust
  Mit randomization wird verhindert, dass der Sender überlastet wird

Mobility

- indirektes Routing: Kommunikation vom Sender zu mobilen Nutzer geht über den home agent und wird weitergeleitet zu remote
  - permanente Adresse: durch den Korrespondent benutzt
  - Care-of-address: benutzt home agent um Daten weiterzuleiten
  - triangle routing: Korrespondent-Heimnetzwerk-mobiles Netzwerk
- direktes Routing: Sender bekommt die Fremdadresse des mobilen Nutzers und kommuniziert direkt dahin
  - transparenter für den Korrespondenten
- Registration: Foreign agent registriert mobilen Nutzer bei Home agent

Content Delivery Networks

- Idee: replizierte den Inhalt und stelle ihn lokal bereit
- Content-Server sollten so nah wie möglich bei den Endbenutzern stehen
- im Gegensatz dazu: P2P-Netzwerke - in der Theorie unendliche Skalierbarkeit
- möglich z.B. mit DNS Weiterleitung → Load balancing
- bester Server nach Belastung, Fehleranfälligkeit, Netzwerkauslastung ausgewählt

Ressourcenzuteilung

Multiplexing: Teilen der Ressource unter allen angeschlossenen Nutzern
QoS (Quality of Service): das Netzwerk bietet Applikationen mit Leistungsniveaus Garantien, die sie für ihrer Funktionsfähigkeit brauchen
Serviceklassen sind Best-effort, hard real-time und soft real-time.
5 Prinzipien für QoS:

1. Traffic Spezifikation
   Applikationen sollen spezifizieren, wie viel Bandbreite sie brauchen und welche Serviceklasse sie möchten
2. Traffic Klassifikation
   Paketmarkierungen sind nötig, um die verschiedenen Klassen durch den Router zu erkennen und die Pakete nach den Policys richtig zu behandeln
3. Traffic Isolation
   Schutz/Isolation für eine Klasse von anderen anbieten
4. Anruf zulassen
   Netzwerk muss einen Anruf blockieren (z.B. durch Besetzt) wenn die Bedingungen nicht erfüllt werden können
5. Ressourcenteilung
   Ressourcen so effizient wie möglich benutzen

QoS im Internet
IETF Integrated Services **(IntServ)**

- reserviert Ressourcen bei jedem Router zwischen beiden Endpunkten für jede Verbindung
- Call Admission
  - R-Spec: definiert die QoS-Notwendigkeit
  - T-Spec: definiert Traffic-Charakteristik
  - RSVP: Signaling Protokoll zum Übertragen von R-Spec und T-Spec entkoppelt Routing von Reservierung

IETF Differentiated Services **(DiffServ)**

- priorisiert die Paket-Flüsse
- schafft qualitative Serviceklassen, “Verhalten wie ein Kabel”
- markiert Paket in den Edge-Routern, priorisiertes Scheduling in Core Routern

Scheduling

Def: wähle das nächste Paket aus, welches gesendet werden soll
FIFO (first in first out)

- senden in der Reihenfolge, wie die Pakete ankommen

Strict Priority Scheduling

- Übertragung nach Höhe der Priorität, nach Markierung oder anderen Header-Infos

Round Robin

- mehrere Klassen, zyklische je einen aus jeder Klasse senden

Weighted Fair Queuing

- wie Round Robin, nur mit gewichteter Servicezuteilung

Signaling

- **Def**: Austausch von Nachrichten zwischen Netzwerkentitäten um Verbindungen/Anrufe zu ermöglichen
- beinhaltet den Verbindungsauf-, abbau und -haltung, Messung, Rechnung
- zwischen Endnutzer - Netzwerk, End. - End. oder Netz-elem - Netz-elem.
- Voraussetzung: Transportprotokoll braucht State- und Variableninitialisierung
- SIP: Applikationsschichtprokotoll, das Benutzern ermöglicht erreichbar zu sein unabhängig von seinem Gerät oder Ort
- RSVP: Ressourcen reservieren entlang des End-to-End-Pfades für QoS

State

Designprinzipien:

1. Trennung von Daten und Kontrolle
   - RSVP: Signalisierung separiert vom Routing, Forwarding
   - HTTP: In-Band Signalisierung (Request-Nachricht ist Signalisierung, keine Daten, Antwort mixt Signalisierung mit Daten)
   - FTP: Out-of-Band Signalisierung (separate Kontrollverbindung Port 21, Datenverbindung Port 20)
2. Hard state vs. soft state
   State: gespeicherte Informationen von den Protokollen in den Netzwerkknoten
3. Randomization
4. Indirektion
5. Netzwerk-Virtualisierung/-Überlagerungen
6. Ressourcenteilung
7. Design zur Skalierung

Hard State vs. Soft State

Hard State

- Status, der bei Verbindungsaufbau installiert wird und bei Verbindungsabbruch gelöscht z.B. TCP, SS7
- Standard-Annahme: Status ist solange gültig, bis es anders definiert wird
- zuverlässiges Signaling, explizite State-Löschung
- Inkonsistenz, Overhead nehmen mit zunehmender State-Lebenszeit ab
- explizite Löschung verbessert die Konsistenz mit kleinem Overhead

Soft State

- Status wird bei Verbindungsaufbau installiert, wird aufrechterhalten durch refresh-Nachrichten und gelöscht durch einen Timeout oder keine refreshs z.B. RSVP, IGMPv1
- Standard-Annahme: Zustand wird ungültig, wenn er nicht aktualisiert wird
- mit Refresh-Timer beim Sender und State time-out Timer beim Empfänger

- einfachere Fehlerfindung, nur implizite Zuverlässigkeit

Virtualisierung

- Virtualisierung der Ressourcen: kraftvolle Abstraktion in System-Engineering
- Overlay Network über dem Routing Network für Applikationen
- RON: Resilient Overlay Networks
    - Steigerung von Performance, Zuverlässigkeit und Routing
- VPN: Virtual Private Networks - Netzwerk über gemeinsame Infrastruktur aufbauen

The Big Picture

- wie kann man Funktionalitäten in Protokollschichten aufteilen
    - jeden Schritt zuverlässig machen und alle konkatenieren
    - jeden Schritt unzuverlässig lassen und End-to-End checken, notfalls wiederholen
    - → Mix aus beiden ist gut

Internet End-to-End Argument

- Netzwerkschicht: best effort Daten/Paket-Lieferung
- Transportschicht: End-to-End Fehlerkontrolle durch TCP am Netzwerk edge
- alle anderen Funktionen finden auf Applikationsebene statt

Internet Design Philosophie

1. Survivability/Überlebensfähigkeit
   weiter operieren, auch wenn Netzwerkfehler auftreten
   Entscheidung: E2E-Transport-State wird nur auf den Endpunkten gespeichert
   Internet ist statuslose Netzwerkarchitektur
2. Servicetypen
   UDP und TCP für verschiedene Anwendungsfälle
3. Vielfalt an Netzwerken
   IP over everything (the glue of the internet)

- erlaubt verteiltes Management
- ist möglichst kosteneffektiv