|  |  |  |  |
| --- | --- | --- | --- |
| Traceroute Misst Weg vom Starthost zum Zielhost  Für alle i:  - Sende Pakete die nur i laufen können richtung Ziel (TTL Feld)  - i.ter Router sendet Pakete zurück -> Sender lernt alle Router kennen – vgl. Breitensuche  Basiert auf ICMP   Home Router Aufgaben DHCP Server, WLAN Access Point, IP Router, DSL Modem / Kabelmodem, DNS Resolver | Leitungsvermittlung (Circuit Switching) Benötigte Ressourcen müssen vorab reserviert wrdn  Verbindung wird nur zugelassen, falls ausreichend Netzkapazität vorhanden. Sonst abgelehnt.  Dann Senden eines kontinuierlichen Datenstroms.  Übertragungsrate garantiert. Paketvermittlung (Packet Switching) Host teilt Nachricht in kleine Pakete auf und schickt sie unabhängig voneinander los.  Gleichzeitige Pakete müssen sich einen Link teilen und zeitlich hintereinander gesendet werden.  Router: **Store-and-Forward**.  Jeder Router muss gesamtes Paket empfangen, bevor er es auf den ausgehenden Link weiterleitet | Paketverzögerungen/-Verlust Verlust: Pakete verworfen, wenn Puffer nicht frei  Verzögerung: durch Pufferung  **dnodal** = dproc + dqueue + dtrans + dprop  **dtrans** = Paketlänge(Bits) **L** / Bandbreite d. Links(bps) **R**  **dprop** = Länge d. Links **s** / Ausbreitungsgeschwindigkeit **v** (~2\*108 m/s)  Datenraten:  10er Potenz  Speicher: 2er  B: Bytes  b: bits  **Bottleneck** bestimmt den Throughput: | Schichtenmodell (TCP/IP - Internet) **Vorteile**: Verringert Gesamtkomplexität. Einfacheres Hinzufügen neuer Features. modulares Entwickeln von für die Datenkommunikation benötigter Hardware, Treiber und Anwendungen. Schichten können geändert werden, ohne Einfluss auf andere Schichten.  Jede Schicht fügt an die Nachricht ihren eigenen Header hinzu  **5 Application** (HTTP, SMTP, RTP, DNS)  **4 Transport** (TCP, UDP): Kommunikation zw. Prozessen auf Sender u. Empfängerseite  **3 Network** (IP, ICMP): Adressierung, Forwarding, Routing  **2 Link** (DSL, SONET, 802.11, Ethernet): Rahmenbildung, Fehlererkennung/Korrektur, Vielfachzugriff, zuverl. DatÜbertr  **1 Physical**: Signalübertragung, Modulation, Übertragungsmedien |
| Schichtenmodell (ISO/OSI)  |  | | --- | | 7 application | | 6 presentation | | 5 session | | 4 transport | | 3 network | | 2 link | | 1 physical |   Zusätzlich:  **Presentation**: Semantik der übertragenen Kommunikation  (Kompression, Verschlüsselung, BE, LE)  **Session**: Sitzungsauf- und abbau  Synchronisierung zwischen beteiligten Prozessen  **Nachteile**: Leichter Overhead, teilweise gleiche Aufgaben in mehreren Schichten implementiert (historische Gründe), Höhere Schicht benötigt Infos aus niedrigerer Schicht (z.B. WLAN Routing Protokolle (Schicht3) benötigen Infos aus Schicht2)  Änderungen an Service API Schicht k hat Auswirkungen auf k+1 | Signalübertragung **Dämpfung**: Längere Leitung -> mehr Dämpfung  Leistung/Amplituden verringert  **Verzerrung**: Frequenzen werden von Übertragungsmedien verschieden stark gedämpft.  Meist nur Frequenzen bis zu einem max Wert gut übertragbar  **Bandbreite**: E-Technik: Frequenzbereich der gut übertragen werden kann Duplex vs Simplex **Vollduplex**: Beide Richtungen gleichzeitig möglich  z.B. meist bei Kabelübertragung  **Halbduplex**:Beide Richtungen, aber nicht gleichzeitig  z.B. WLAN  **Simplex**: Nur eine Richtung möglich | Nyquist (Datenrate D bei unverrauschtem Kanal) Bandbreite **B**; Anz. verw. Signalstufen **V**  **D = 2 \* B \* ld( V )** [bit/s] Shannon (Datenrate D bei verrauschten Kanal) Gilt zusätzlich zu Nyquist!  Nutzsignalleistung S; Rauschleistung N  **D = B \* ld( 1 + S/N )** [bit/s]; S/N in dB: 10 \* log10(S/N)  Wenn in dB geg., dann erst umrech Bit vs Baud Bitrate:  20 bit/s  Baudrate(Anz.  Signalschritte  pro Sekunde):  10 Baud | Digitale Modulation **Modulation**: Umwandlung Bitsequenz in übertragbares Signal.  **Demodulation**: Rückübersetzung beim Empfänger.  **Baseband** (bei drahtgebundener Übertragung):  Signal beinhaltet Frequenzen 0 bis fmax und wird direkt in diesem Frequenzbereich übertragen.  Bei Ethernet Kabel z.B. 1000BaseT bedeutet:  1Gps, Baseband, Twisted Pair Kupferleitung  **Passband** (bei drahtloser Übertragung):  Nutzsignal in höheren Frequenzbereich verschieben  Nutzsignal verändert Trägersignal  Rückgewinnung am Empfänger durch **Demodulation** |
| Baseband Transmission (Leitungscodes) Manchester: XOR Clock mit Bits  Bipolar: abwechselnd + u. – für **jede** 1 | Taktrückgewinnung durch Leitungscodes Häufige Symbolwechsel nötig, damit Empfänger den **Takt rückgewinnen** kann. 100000 schwierig wie viele 0en.  **Lösungen**:   * Synchrone Uhren * Manchester Code (Takfreq = 2 \* Bitfreq) * Coderung: z.B. **4B/5B** bildet 4 Bits auf 5 Bits ab mit vielen Wechseln (so bestehen Codewörter niemals nur aus 1en oder 0en):     Weitere Vorteile: Effizientes Ausnutzen der Bandbreite.  Gleichspannung unterdrücken(AMI) | Passband Transmission Nutzsignal ändert Trägersignal  Bei Frequency Vereinbarung welche Freq 0 und 1  Grund für Passband: **Multiplexing** u. Antennengröße | Passband: Kombination von Modulationsarten ASK und PSK oft kombiniert -> ergibt mehr Symbole, daher höhere Bitrate bei gleicher Baudrate    GrayCode als Bitcodes. -> Kleine Fehler bei der Symbolerkennung verursachen nur wenige Bitfehler. |
| Multiplexing (mehrere User 1 Übertragunsmedium) **Frequency Division Multiple Access (FDMA):**   * Jeder Benutzer hat eigenen Frequenzbereich * Bei **Vollduplex** zwingend notwendig!   **Time Division Multiple Access (TDMA):**   * Frequenzbereich wird über Zeit geteilt * Round-robin * Benutzer wechseln sich zeitlich ab   Auch **Kombination** aus beiden Möglich. | Link Layer **Host**: Endpunkt eines Ende-zu-Ende Pfades  **Node**: Jedes Gerät, das am Netzwerk teilnimmt (Host, Router, Switch, Access Point,...)  **Frame**: Nachricht auf Schicht 2.  Wird in allen Nodes implementiert (**Netzwerkkarte/HW**).  Nicht in Hubs!  **Übertragung von Frames zwischen benachbarten Nodes**  **Rahmenbildung**: Positionsrichtige Erkennung von Zeichen, Erkennung von Blockgrenzen. Frame = Header+Payload. Payload = IP Paket  **Vielfachzugriff**: Wer darf Medium wann nutzen?  **Fehlererkennung/-Korrektur:** Umgang mit Bitfehlern auf phyical layer. Hinzufügen von Redundanz  **Zuverlässige Datenübertragung**: Korrektur von Paketverlusten, korrekte Reihenfolge, Vermeidung von Duplikaten. Bei WLan teilweise, bei Ethernet gar nicht. | Rahmenbildung (Zerlegung des Bitstroms in endliche Seq) Erkennung, wann Frame beginnt und endet.  **Byte Count**: Zu Beginn jedes Frames Feld, das Anz enthaltener Bytes angibt (Anz. inkl diesem Feld)  Nachteil: Nach Fehler erneute synchronisation schwer  **Byte Stuffing**: FLAG markiert anfang und Ende. Falls FLAG in Nutzdaten, ESC. ESC aber auch escapen. Overhead, aber einfaches Synchronisieren nach Fehler.    **Bit Stuffing**: Frame beginnt mit speziellen reserviertem Bitmuster. Beim senden wird nach 5 zusammenhängenden 1-er Bits immer ein 0 Bit eingefügt und beim Empfang nach 5 zusammenhäng. 1er Bits immer ein 0 Bit gelöscht.  Vorteil: Framelänge muss kein vielf. von 8Bit sein. | Fehlererkennung und -korrektur **Keine Fehlerkorrektur** (zu viel Redundanz)  Bei Fehlererkennung:  Ethernet: keine Retransmission. Nur durch TCP, falls Timeout eintritt.  WLAN: Aktive Wiederanforderung des fehlerhaften Blocks durch Link Layer  **Checksumme** (IP und UDP/TCP Header):   * Bits in Gruppen von 16 Bit Wörtern * Summiere alle 16 Bit Wörter unter Berücksichtigung des Übertrags (Übertrag addieren) * 1er Komplement des Ergebnisses ist Checksum   Empfänger:   * Addiere übertragene Wörter UND Checksum. Übertrag auch wieder addieren! * Wenn Ergebnis nur 1er: Kein Fehler |
| Cyclic Redundancy Check (CRC): Ethernet und WLAN Zu übertragende **Datenbits D**  **Generator G**, den Sender u. Empfänger kennen (r+1Bits)  Sender bestimmt **r zusätzliche Bits** und hängt diese an d an.  d+r wird dann übertragen. **d+r muss durch G teilbar** sein.  **Sender**: R berechnen:  R = Rest( (D\*2r)/ G )  R ggfs mit führenden Nullen zu  r Stellen ergänzen.  **Empfänger** prüft, ob durch G teilbar  D \* 2r XOR R = n\*G?  Falls rest übrig bleibt -> Fehler  Funktioniert schlecht, wenn Bitfehler  direkt hintereinander | MAC-Adressen (Adresse der Link Layer) Nur lokal gültig. Zur Identifikation von Nachbarn.  Jedes Interface eines **Hosts** hat eigene MAC-Adresse.  Switch hat keine MAC Adresse, weil kein Host!  Bei Ethernet u. WLan 48 Bit.  **Broadcast-Adresse** FF-FF-FF-FF-FF-FF  Jede Netzwerkkarte muss eindeutige MAC-Adresse haben innerhalb eines lokalen Netzwerkes.  Teilweise MAC Adresse fest mit Netzwerkkarte verknüpft. | Ethernet 802.3 Frames:   **Präambel**: 7mal 10101010, dann 1mal 10101011   * Synchronisation Sender u. Empfänger   **Adressen**: je 6 Byte Sender u Empfänger MAC. Normalerweise, NW Karte leitet Frame nur an BS weiter, wenn des adresse passt. Ausnahmen: Broadcast oder Promiscuous Mode  **Type**: 2Byte Art des Netzwerkprotokolls IPv4/IPv6...  **CRC**: 4Byte  **Eigenschaften**:  **Verbindungslos**: Kein Verb. Aufbau vor Datenaustaus  **keine zuverlässige Verbindung**: Frameverlust mögl.  **Vielfachzugriff**: Nur bei Punkt-zu-Punk:Unsl CSMA/CD | Vielfachzugriff Geteilter Broadcastkanal – **Interferenz** == Kollision falls mehrere Stationen gleichzeitig senden.  **Multiple Access Control**: Verteilter Algorithmus, der entscheidet, wann Host senden darf.Entscheidung muss inband sein (kein extra Kanal).  Link hat Kapazität R. Wenn einer, dann Rate R, wenn mehrere Senden Rate R/M  **Arten von Multiple Access Control:**  **Multiplexverfahren**: siehe physical Layer.  **Random Access Verfahren**: Kollisionen (>2 Stationen senden) werden zugelassen. Mechanismen um sich von Kollision zu erholen. z.B. Un-/Slotted ALOHA, CSMA /CD /CA  **Token-Verfahren**: Kollisionen werden verhindert. Nur wer Token hat darf auf Kanal zugreifen |
| Slotted ALOHA   **Annahmen**: Alle Frames gleiche Größe. Zeit in gleich große Zeitslots unterteilt. Intervall reicht zum Senden des kompletten Pakets. Hosts müssen bzgl Zeit synchronisiert sein.  Sobald neuer Frame vorhanden. Versuche im nächsten Slot zu senden.   * Keine Kollision: fertig. * Kollision: Versuche beim nächsten Slot mit Wahrscheinlichkeit p erneut. (p-persistent)   **Vorteile**: nur 1 Host: volle Rate, Dezentral, Einfach  **Nachteile**: Kollisionen verschwenden Zeitslots. Sync  der Uhren notwendig. Hosts erkennen evtl. Kollision bevor Übertragung beendet ist und brechen das Senden ab. | Carrier Sense Multiple Access (CSMA) **Carrier Sensing**: Mitlauschen am Kanal.  Kanal **frei**: Beginne Übertragung.  Kanal **belegt**: Verschiebe Übertragung.   * 1-persistent: Sende sobald Kanal wieder frei * p-persistent: Sende im nächsten Slot, mit W’keit p falls Kanal frei ist. * Non-persistent: Warte Zufällige Zeit und prüfe erneut, ob Kanal frei. => **Ethernet**   Wegen dprop erkennen Sender erst verspätet, ob es zu Kollisionen kommt.  dprop hat Einfluss auf Kollisionswahrscheinlichkeit.  Problem: Bei spät erkannter Kollision ist losgesendetes Paket wertlos. | CSMA     CSMA/CD | CSMA/CD (Carrier Sensing + Collision Detection) **CS**: Nur senden falls Medium gerade frei ist.  **CD**: Sender (**Netzwerkkarte**) hört während senden Medium weiter ab.   * Sofortiger Abbruch + Jam Signal bei Kollision. Jam Signal zum Sicherstellen, dass andere Nodes Sendetätigkeit einstellen. * Erneuter Sendeversuch nach zufälliger Wartezeit * Binary exponential Backoff: mittlere Wartezeit nach jeder erneuten Kollision verdoppelt.   **Voraussetzung**:  Sender muss zu **listen while talk** fähig sein.   * LAN: Leicht möglich -> *Vollduplex* * WLAN: Schwierig. Empfangene Signale viel schwächer als gesendete -> *Halbduplex* |
| Mindestlänge von Ethernet Frames Worst-Case: Kollision wird erst nach 2\*dprop erkannt.  Ziel ist es, dass Sender **Kollision noch erkennt, bevor er alle Bits seines Frames losgesendet hat**. (Transmission delay)   * Minimale Paketgröße nötig   dtrans > 2dprop => L/R > 2 \* s/v => **L > 2 \* R \* (s/v)** | Vielfachzugriff bei WLAN 802.11 **CS**: Wie Ethernet wird vor Senden Medium abgehört  Collision Detection nicht möglich, weil:   * WLAN ist **halbduplex**: empf. Signal sehr schwach * WLAN Stationen können sich nicht gegenseitig hören * **Hidden Station Problem**      * **Kollisionen müssen beim Empfänger erkannt werden!** | CSMA/CA (Collision Avoidance) bei WLAN ***Sender***:  Kanal min. **für DIFS frei** -> sende kompletten Frame ohne CS  Kanal **belegt**:   * hier schon exponential Backoff * Unterschied zu CSMA/CD * Höre Kanal ständig ab und dekrementiere Timer nur während Zeiten, wo Kanal frei ist. * Erneute Übertragung, wenn Timer ausläuft   Falls kein ACK eintrifft -> Wieder zu belegt Fall.  Ggfs backoff Intervall erhöhen.  ***Empfänger***: bestätigt Datenempfang durch ACK nach Zeitspanne SIFS.  SIFS < DIFS Priorisierung von ACK | Switched Ethernet **Hub**: Alle Leitungen quasi miteinander verbunden. Eine einzige Kollisionsdomäne. CSMA/CD notwendig  **Switch**: Isoliert jeden Port in eigene Kollisionsdomäne  Bei Vollduplex-Kabeln kein CSMA/CD nötig   * Jeder Host direkt mit Switch-Port verbunden. * Keine Kollision möglich, falls voll-duplex. * Kein CSMA/CD notwendig. * Switches speichern Frames zwischen und leiten Frames weiter * Gleichzeitige Übertragung von A zu A’ und B zu B’ möglich. |
| Ethernet Switch Arbeitet auf **Link Layer**.   * Empfang, Zwischenspeicherung und Weiterleitung von Ethernet Frames * Untersucht MAC Adresse der ankommenden Frames und leitet Frame selektiv nur an richtigen Port weiter.   Klassischer Switch hat **keine IP-Adresse**.  **Transparenz**:   * Ethernet Hosts merken nichts von der Anwesenheit eines Switches   **Plug-and-Play:**   * Selbstlernend * Muss nicht konfiguriert werden. | Switch Forwarding **Forwardingtabelle** enthält Info, an welchen Port ein Frame weitergeleitet werden muss:   * Ziel **MAC**-Adr | Ziel **Ausgangsport** | **TTL**   **Ablauf:**   * Selbstlernend: Bei **ankommendem** Frame werden Infos des Absenders gespeichert. * **Nachschlagen**, ob Eintrag mit Ziel MAC schon in Tabelle. * **Vorhanden**: Weiterleiten an Zielport. Falls Zielport == Quellport. Frame verwerfen. * **Sonst**: Fluten. Weiterleiten an alle Hosts mit Ausnahme des Senders. Auch die, die er schon weiß, was dranhängt. | CSMA/CA Ablauf: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Network Layer Ende zu Ende Verb. Zw. Sender u. Empfänger  Router interessieren sich nicht für Schicht 4 / 5  **Adressierung**: IP Adressen  **Forwarding**: Welches Ausgangsinterface des Routers. Bei Router oft in HW implementiert  **Routing**: Berechnung der Wege im Netz. Eintragen d. Ergebnisse in ***Weiterleitungstabellen*** (ip Adr. Reichweiten jeweils). Routingprotokolle (konstruieren Routing Tabellen). Meist in SW implementiert  IP ist **verbindungslos**.  Link Layer kann unterschiedlich sein (WLAN, Ether..)  **Best Effort**: Jeder Router tut sein Bestes, aber keine Garantie bzgl Reihenfolge, Bandbreite… | Router Architektur Queuing, falls Ankunftsrate schneller als Weiterleitung durch Fabric.  Bei IP wird Ausgangsport nur anhand der IP Zieladresse bestimmt! | Switching Fabric 3 Typen: Memory, Bus, Crossbar Queuing an Eingangsports Nötig, falls Fabric langsamer als Ankunftsrate.  Head-ofthe-Line Blocking: Vorderstes Paket blockiert andere Pakete, obwohl andere Pakete zu einem Ausgangsinterface müssen, das frei ist. Queuing an Ausgangsports Nötig, falls Ankunftsrate von Fabric die Übertragungsrate des Ausgangslinks übersteigt.  Wenn Queue voll -> **Paketverluste**! | IP (20 Bytes Overhead für IP Header)  * Adressierungskonventionen TTL: Anz Hops * Datagram Format verbleibend * Packet handling conventions  IP Fragmentierung MTU: Verschiedene Link Layer Technologien haben versch. max. Paketgrößen (Ethernet 1500 Byte)  Router/Host zerlegt in kleinere Pakete.  Zusammenbau am End-Host!  **16-Bit Identifier**: Identisch für alle Pakete eines Frames  **Fragmentation Flag**: 1: Da kommt noch was, 0: letztes Fragment eines Pakets  **Offset**: Byteposition innerhalb des Gesamtpakets, an die das Fragment gehört. (Offset 185: 185 \* 8 = 1480, also nach Byte 1480 kommt dieses Fragment) |
| IP Adresse 32 Bit. Identifiziert Host im Internet, gehört aber logisch eigentlich zum Interface des Hosts. Jedes Interface (am Router mehrere) eigene IP-Adresse. Subnetze Mehrere Hosts teilen sich gleichen IP Adresspräfix.  Innerhalb von Subnetz können sich Hosts ohne Router erreichen.  Vorteil: In Routingtabellen müssen  nur Subnetzadressen stehen  **Beispiel**: 223.1.3.0/24  (die ersten 24 Bits sind für  alle Hosts des Subnetzes gleich) | Classful Addressing Früher: **Feste länge** für Subnetzpräfixe (/8, /16, /24)  /24 Netz kann 2^(32-24) = 2^8 Hosts haben Classless Addressing **Beliebige Länge** für Subnetzpräfixe (CIDR)  Präfixnotation: 200.23.16.0/24  **Netzmaske**: 255.255.255.0  zeigt, welche Bits zum Subnetz gehören Spezielle Ipv4 Adressen 127.0.0.1: Localhost, eigener PC. Netmask 255.0.0.0  Private Ipv4 Adressen: 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, 169.254.0.0/16  Beispiel:192.168.0.0/16 Netzmaske: 255.255.0.0 Broadcst:192.168.255.255 Netzadresse: 192.168.0.0 | DHCP (Dynamic Host Configuration Protocol) Server weist automatisch IP Adressen zu  Eigentlich Schicht 4!  Host kann zugewiesene IP Adresse ggfs verlängern.  **Ablauf**:   * Host sucht DHCP Server: DHCP Discover (optional)   Ziel: 255.255.255.255 (Ethernet Broadcast)   * DHCP Server antwortet mit DHCP Offer (optional)   Ziel: 255.255.255.255 (Ethernet Broadcast)   * Host fordert explizit IP Adresse an: DHCP Request   DHCP Server weist Adresse zu: DHCP ACK | ICMP (Internet Control Message Protocol) Error reporting und Router signaling  Austausch von Infos zw. Host und Routern  ICMP Information wird als  IP Paket versendet  **Inhalt ICMP Nachricht**:  Type, Code  Erste 8 Bytes des IP Pakets,  das Fehler verursacht |
| ARP (Address Resolution Protocol) Herausfinden, welche Ziel-MAC zum Next-Hop Router/Host gehört. ARP Paket wird in Ethernet Frame verpackt   * Übersetzen von IP in MAC Adressen   Arp Tabelle:  IP | MAC  Routing Tabelle:  IP | Port  Switch forwarding Tabelle: MAC | Port   * ARP direkt auf Ethernet => kein IP Header | ARP Ablauf (Sender und Empfänger in einem Netz) A möchte Datagram zu B senden  B’s MAC nicht in A’s ARP Tabelle  A schickt Broadcast ARP Query Paket das B’s IP Adresse enthält (Ziel MAC: FF-FF-FF-FF-FF-FF)  Alle Hosts im LAN empfangen diese ARP Query  B empfängt ARP Query und informiert A in Antwort über B’s MAC Adresse -> Unicast Frame zu MAC A  A speichert IP/MAC paar in ARP Tabelle bis veraltet  ARP bedarf keiner Konfiguration (plug and play) | ARP Ablauf (Sender u. Empfänger in versch. LANs) | Routing **Link State (zentral)**: Jeder Router kennt komplette Topologie. Z.B. OSPF  **Distance Vector (dezentral)**: Jeder Router kennt nur direkten Nachbarn u. Kosten zu diesem. Nachbarn teilen per Routingnachrichten mit welchen Knoten sie mit welchem Gesamtkosten erreichen können. Z.B. Routing Information Protocol (RIP)).  **Statisch**: Manuelle Konfiguration von Forwardingtable  **Dynamisch**: Periodischer Austausch von Routinfoormationen. Änderungen autom. erkannt. |
| Links State vs Distance Vector **Routingnachrichten**:  LS: Jeder Router flutet Infos über seine Links im ganzen Netz  DV: Jeder Router informiert seine Nachbarn welche ziele er zu welchen Kosten erreichen kann  **Robustheit**: Was wenn ein Router bösartig ist?  LS: Router kann falsche Linkkosten ankündigen.Fehler begrenzt, da jeder Router seine eigene Tabelle berechnet. DV: Router kann flasche Pfadkosten ankündigen. Fehler pflanzen sich fort, da Tabelle eines Routers Einfluss auf andere Router hat. | Hierarschiches Routing Router werden In autonome Systeme (AS) guppiert. Intradomain Routing (für Ziele in anderen Asen) Wie sind Gateways aus Nachbarnetzen aus lokalem Netz erreichbar?  Protokolle: **RIP, OSPF, IGRP** Interdomain Routing (für Ziele im gleichen AS) Welche externen Ziele sind über welches Transfer-AS / Gateway erreichbar  Protokolle: **BGP**  Zusätzliche Routing Policies notwendig:  Jeder Router kann bestimmen, welche Nachrichten er bevorzugt.  Auch wirtschaftliche Aspekte spielen eine Rolle. | OSPF (Open Shortest Path First) Router fluten **Link State** Advertisement Nachrichten (enthalten Infos über alle Nachbarrouter) an alle anderen Router im gesamten AS. Dann Routenberechnung über Dijkstra.  OSPF Advertisements werden direkt über IP gesendet (kein TCP oder UDP).  Router muss lernen „über welches Interface komme ich aus meinem AS raus zum Ziel, welches sich evtl. in einem ganz anderen AS befindet“. Das steht dann in der **Routing Tabelle** der Router | BGP (Border Gateway Protocol) Teile Dem Rest der Welt die Existenz eines IP Präfix mit.  **eBGP**: Verbindungen zwischen ASen. Teilt anderen ASen Mit, dass man über diesen Router zu einem bestimmten IP Subnetz kommt.  **iBGP**: Verbindungen innerhalb eines AS. Teilt anderen Routern innerhalb eines AS mit, dass man über diesen Router zu einem bestimmten Ziel kommt.  **BGP Attribute**:   * AS-PATH: Liste von ASen, durch die Prefix Advertisement gelaufen ist.   NEXT-HOP: Router im eigenen AS, der das AS mit dem nächsten AS auf dem AS Pfad verbindet. |
| BGP **Policy-based Routing**: Beispiele:   * Ignoriere Pfade durch AS Y * Gib Routinginfo nicht an Nachbarn AS X weiter   **BGP, OSPF**: Welches Ausgangsinterface muss Router verwenden, um zum Gateway Router zu kommen, der über iBGP mitgeteilt wurde.  **Beste Route nach Kriterien**:  1: Local Pref, 2: Kürzester AS Pfad, 3: Route mit am schnellsten erreichbarem Next-Hop (Hot Potato).. Hot Potato Routing AS will, dass Pakete so schnell wie möglich das eigene AS verlassen, also lokales Gateway mit **geringsten Intradomain Kosten** wählen. | IPv6 40 Byte Header Kein ARP  Keine Fragmentierung und Keine Checksumme  **Notation**: 128 Bit in 8 Blöcke, je 16 Bit (4Hex Zahlen), mit „:“ getrennt.   * Führende Nullen darf man weglassen * EINMAL dürfen ein oder mehr aufeinanderfolgende Blöcke mit 0000 ausgelassen werden und durch „::“ ersetzt werden. * Beispiel: 2001:0db8:0:0:0:0:1428:57ab wird zu: 2001:db8::1428:57ab   Es gibt keine /80 Subnetze, wel Host-ID immer genau 64 Bit.  IPv4 Notation: ::192.31.20.46   * **Tunneling**:IPv6 Paket in IPv4 Pak(bei legacy Leitung | Transport Layer Kommunikation zwischen **Prozessen** auf Sender u. Empfänger Seite. (Betrifft Hosts, nicht Router!)  Teil des **Betriebssystems** Transport Layer Multiplexing IP Pakete werden Prozessen des BS zugeordnet.  Port: 16 Bit.   UDP Verbindungslos: Socket definiert durch: **Dst IP/Port**  UDP und TCP, garantieren nicht Delay/Bandbreite! | TCP (alles in SW implementiert) Verbindungsorientiert: Socket definiert durch:  **Src IP/Port, Dst IP/PORT**  Zuverlässig: Keine Bitfehler, kein Datenverlust, korrekte Reihenfolge  **Problem**: Zuverlässige Übertragung über unzuverlässigen Kanal (IP)  **ACK**: Um zu sagen, ob die Daten angekommen sind, und OK (Checksum) sind.  **Sequenznummern**: Um zu sehen, ob ACK Corrupt ist und um zu sehen, ob das letzte Paket richtig angekommen ist (Ersatz für NACK).  **Timeout**: Schützt vor Daten/ACKs die auf dem Weg verloren gehen.  **Pipelining**: Für verbesserte Performance. |
| Go-Back-N Empfänger bestätigt immer nur mit Sequenznummer für die gilt, dass alle kleineren bereits empfangen.  Bei Timeout werden **alle** Pakete nochmal geschickt, die noch nicht bestätigt wurden. Selective Repeat Retransmission nur für verlorengegangene Pakete.  Empfänger schickt ACKs für jedes einzelne Paket individuell | Go-Back-N | Selective Repeat | TCP **Zuverlässig**: Bestätigt einzelne Bytes, nicht Pakete!  **Pipelining**: Misch. aus Go-Back-N u. Selective Rep.  **Vollduplex**: Max Segment Size (**MSS**) richtet sich nach MTU der Link layer  **Verbindungsorientert**: Verbindungsaufbau vorher. Sender u. Empfänger initialisieren State Machine  **Flow Control** u. **Congestion Control**  **Sequenznummer:** Nummer des BYTES, NICHT PAKETNUMMER!  **Acknoledgement:** Sequenznummer des nächsten erwartetet Bytes == ältestes noch nicht empf Byte.  **Bidirektional**: Sequenznummer in eine Richtung sind ACK-Nummer in die andere Richtung |
| Hybrid von Go-Back-N und Selective Repeat **Von Go-Back-N:**   * Kumulative ACKs * Nur 1 Retransmission Timer. ältest unbestät Segm   **Von Selective-Repeat:**   * Empfangspuffer * Bei Timeout wird nur das verlorengegangene Paket erneut gesendet   **Auslösen von Retransmissions durch:**   * Timeouts * Duplikat ACKs (3) => **Fast Retransmit**:   3x Duplicate ACK: Indiz für Paketverlust  Sender erhält mehr als 3x gleiches ACK => Retransmission des ältesten unbestätigten Segment | TCP TODO: BILD | TCP Verbindungsaufbau (3-Way-Handshake) Verbindungsaufbau == Overhead | TCP Verbindungsabbau |
| Flow Control (Empfänger zu langsam) Freier Puffer **rwnd** =  RcvBuffer – (LastByteRcvd – LastByteRead)  Empfäner teilt Sender rwnd mit. Congestion Control (Netzwerk zu langsam) 2 Ansätze:  **Netzwerk-unterstützt**: Router geben Rückmeldung an Hosts bei Überlastung.  **Ende-zu-Ende**: Auslastung des Netzwerks wird durch Beobachten der Verzögerungen und Auftreten von Paketverlusten abgeschätzt. (TCP!!)  **Additive Increase**: Nach jeder RTT wird cwnd um 1 MSS erhöht bis Paketverlust erkannt  **Multiplicative Decrease**: Halbiere **cwnd** nach erkanntem Paketverlust  **Window Size <= min{cwnd,rwnd}** | TCP Reno **Slow Start**: Verdopple *cwnd* nach jeder RTT bis zu ssthresh  **Congestion Avoidance**: Dann – vergrößere *cwnd* um 1 nach jeder RTT  **Nach Timeout**: TCP Slow Start | ssthres = cwnd/2 | cwnd = 1  **Nach 3 Duplicate ACKs:** Congestion Avoidance,  ssthres = cwnd/2, cwnd = cwnd/2 + 3  https://raw.githubusercontent.com/SebSDev/semester_4_uebungen/master/Anastasia/Rechnernetze/automatTCPReno.png | NAT (Network Address Translation) Lokales Netzwerk benutzt nur 1 IP Adr. um mit Rest des Internets zu kommunizieren.  16 Bit Portnummer-Feld => >60000 gleichzeitige Verbindungen mit 1 öffentlichen IP.  Umstritten, da Schicht 4(Portnummer) nicht vom Router berücksichtigt werden sollte.  **Port forwarding:**  Verbindungsanfragen an bestimmten Port werden fest zu bestimmten Server weitergeleitet.  **Beispiel**: (123.76.29.7, Port 2500) wird immer  zu (10.0.0.1, Port 25000) weitergeleitet.  Manuelle Konfiguration notwendig! **NAT Tabelle:** | Alle Datagramme, die das lokal Netz verlassen, haben gleiche Src IP, aber verschi. Src Ports   |  |  |  |  | | --- | --- | --- | --- | | WAN Seite/Internet | | LAN Seite/Heim-Netz | | | IP Adr | Port | IP Adr | Port | | 239.120.38.24 | 5100 | 192.168.0.4 | 9929 | | 239.120.38.24 | 5101 | 192.168.0.5 | 9929 | | 239.120.38.24 | 5102 | 192.168.0.6 | 9929 | | 239.120.38.24 | 5103 | 192.168.0.4 | 9930 | | ... | ... | ... | ... | |
| TODO: letztes Thema |  |  |  |

Abkürzungen

## Einführung

DSL: Digital Subscriber Line

ISP: Internet Service Provider

TCP: Transmission Control Protocol (Netzwerkprotokoll, das definiert auf welche Art und Weise Daten zwischen Netzwerkkomponenten ausgetauscht werden sollen)

UDP:

IP: Internet Protocol (Protokoll das die Grundlage des Internets darstellt)

HTTP: Hypertext Transfer Protocol (Protokoll zur Übertragung von Daten in der Anwendungsschicht)

RFC: Request for Comments (legt Internet Standards fest)

VoIP: Voice over IP

DSLAM: Digital Subscriber Line Access Multiplexer (übersetzt hochfrequente Töne in digitale Signale, bevor Daten zum Modem im Heimnetz kommen)

CMTS: Cable Modem Termination System (Wie DSLAM aber für Kabelmodem)

DHCP Server: Dynamic Host Configuration Protocol Server (Verteilt automatisch Adressen an Hosts in einem Netzwerk)

DNS Server: Domain Name System Server (Weist im Internet einer URL die richtigen IP-Adresse zu)

SAP: Service Access Point (Im Schichtenmodell stellt jede niedrigere Schicht der jeweils höheren Schicht einen SAP zur Verfügung. Somit kann die Höhere Schicht die Services der niedrigeren benutzen)

ISO: International Organization for Standardization

OSI: Open Systems Interconnection

## Network Layer

CIDR: Classless Interdomain Routing (Subnetzteil einer Adresse kann beliebige Länge haben)

Windows/Linux Befehle

Messen der Round Trip Time:

* Windows: ping

Wege eines Pakets durch das Internet verfolgen:

* Windows: tracert
* Linux: traceroute

Routing Tabelle anzeigen:

* Linux: route
* Windows: route print

MAC-Adresse rausfinden:

* Windows: ipconfig /all -> physische Adresse
* Linux: ifconfig -> ether, oder ip addr

MAC-Adresse ändern:

* Windows: Systemsteuerung -> Gerätemanager

Adresszuweisung

* Linux: ifconfig eth0 200.23.16.4 netmask 255.255.255.0 oder ip addr add 200.23.16.4/24 dev eth0 oder persistent: /etc/network/interfaces
* Windows: Systemsteuerung -> Netzwerk- und Freigabecenter -> Adaptereinstellungen

IP Adresse von DHCP Server anfordern

* Linux: dhclient
* Windows (ipconfig /release)

ARP-Tabellen anzeigen

* Linux: arp
* Windows: arp -a

nmap – Portscanner, scannt auf offene Ports im Netzwerk

TCP Verbindungen anzeigen:

* Windows: netstat -p tcp

TODO

Schichtenmodell Aufgaben

Winwos/Linux Befehle