|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **PHYSICAL LAYER(PL)**  Dämpfung(приглушение): je länge Leitung desto mehr Dämpfung  Verzerrung(искажение): Übertragungsmedien dämpfen Frequenzen verschieden stark. Meist nur Frequenzen bis zu einem max Wert(gut übertragbar. Verlust hoher Frequenzteile -> Signal nicht rekonstruierbar.  Bandbreite(B) – Größe des übertragbaren Frequenzbereichs  Datenrate: == Bitrate,  V-Anz Symbole pro Schritt  Baudrate - Schrittgeschwindigkeit // Länge von einem Symbol 2 Bit  Baudrate = D/SymbolLänge    Shannon:  S-Leistung des Nutzsignals, N-Leist.des Rauschens  **je größer -> besser**  Duplex  **Full-Duplex**: beide Übertragungsrichtungen gleichzeitig (Kabel)  **Half-Duplex**: beide Übertragungsrichtungen gleichzeitig, aber nicht gleichzeitig (WLAN)  **Simplex**: nur eine Richtung (unüblich)  Modulation  **Modulation**: Umwandlung Bitsequenz in übertragbares Signal.  **Demodulation**: Rückübersetzung beim Empfänger  **Baseband** (drahtgebundene Kom-tion):  Signal und wird direkt in diesem Frequenzbereich übertragen(Ethernet, USB-Schnittstelle).  **Passband** (drahtloser Kom-tion)[fmin,fmax]:  Nutzsignal wir in höheren Frequenzbereich verschoben  Nutzsignal verändert Trägersignal(WLAN, Blutooth)  Rückgewinnung am Empfänger durch Demodulation    Manchester will verhindern, dass das Signal zu lange auf 0 oder 1 bleibt -> So wird gleichzeitig Daten und Clock in einer Leitung übertragen (BitStream XOR Clock)  Bipolar encoding ist günstig, da wir Gleichstrom verwenden können  NRZI hat den Nachteil, dass bei langen 0-len oder 1-sen die Synchronisation außer Tritt kommen kann  NRZ(Non-Return-to-Zero) hat den Nachteil, dass der Empfänger bei der Übertragung einer längeren Folge gleicher Symbole unsicher wird über die Länfge der Folge  Taktrückgewinnung durch Leitungscodes  Häufige Symbolwechsel nötig, damit Empfänger die Symbole rückgewinnen kann. 1000000 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 (16 Blockcodes auf 32 Leitungscode) der Code ist nicht gleichstromfrei:    Passband Transmission  Verschiebung des Nutzdaten *in höheren Frequenz* ist notwendig, *da nur ein gleichzeitiges Signal möglich*, falls nur Bereich [0; fmax] zur Verfügung steht und Antene für tiefe Frequenz müssen riesig sein    **ACK und FSK werden häufig kombiniert** -> Ergibt mehr Symbole -> höhere Bitrate & Baudrate const    BPSK/QPSK ändert nur die Phase, QAM ändert Amplitude & Phase  Multiplexing (mehrere User 1 Übertragunsmedium)  **Frequency Division Multiplexing (FDM):**  - Jeder Benutzer hat eigenen Frequenzbereich  **Time Division Multiplexing (TDM):**  - Frequenzbereich wird über Zeit geteilt  - Round-robin  - Benutzer wechseln sich zeitlich ab  Auch Kombination aus beiden Möglich.  Slotted ALOHA  Alle Frames :gleiche Größe. Zeit in **gleich große Zeitslots** unterteilt. Intervall reicht zum Senden des kompletten Pakets. Hosts müssen bzgl Zeit synchronisiert sein.  **Kollision**: Falls >=2 Hosts in gleichem Slot -> Kollision  Sobald neuer Frame vorhanden. Versuche im nächsten Slot zu senden. Falls **keine Kollision**: fertig. Sonst **p-persistent ALOHA**: Sendeversuch beim nächsten Slot mit Wahrscheinlichkeit p erneut.    (+) nur 1 Host aktive: volle Ratedes Kanals,(+)dezentral,(+)einfach  (-) Kollisionen verschwenden Zeitslots. (-) Host könnte Kollision erkennen bevor Übertragung fertig und Senden abbrechen (-)Sync  der Uhren notwendig  MAC-Adressen (Adresse der Link Layer)  Nur lokal gültig. Identifiziert Nachbarn.  Jedes Interface eines Hosts hat eigene MAC-Adresse  Bei Ethernet & WLAN 48 Bit.  Broadcast-Adresse FF-FF-FF-FF-FF-FF  Jede NIC muss **eindeutige MAC-Adresse** haben **innerhalb eines lokalen Netzwerkes** | **LINKLAYER(LL)**  **Link** – verbindet benachbarten Nodes (WLAN, Ethernet, Mobilfunk)  **Frame(Gruppe von Bits)** – Nachricht auf Schicht2. „Briefumschlag“ für Schicht3 Packet  Dienste von LL:  - Übertragung von Frames zw benachbarten Nodes  - Rahmenbildung(Framing): Positionsrichtige Erkennung von Zeichen, Erkennung von Blockgrenzen. Frame = Header+Payload. Payload = IP Paket  - Vielfachzugriff: Wer darf Medium wann nutzen (nötig, falls Mehrpunktmedium: WLAN, Sattelitennetze)  - Fehlererkennung/-Korrektur: Umgang mit Bitfehlern auf PL. Hinzufügen von Redundanz, um Fehler zu erkennen  - Zuverlässige Datenübertragung: Korrektur von Paketverlusten, korrekte Reihenfolge, Vermeidung von Duplikaten. Bei WLAN teilweise, bei Ethernet gar nicht.  NIC  Man impl LL in allen Nodes. Impl der meisten Funktionalität in HW (Fehlererkennung, Rahmenbindung usw). Netzwerkkarte(NIC) impl-t große Teile der LL und PL. Die wird über Bus mit CPU verbunden. Jede NIC hat HW-Adresse . Anhand dieser Adr lässt sich eine Station auf der Bitübertragungsschicht adressieren  Rahmenbildung  Problem: Erkennung beim Empfang, wann Frame beginnt und endet.  **Byte Count**: jeder Frame beginnt mit dem Feld, das Anz enthaltener Bytes angibt (Anz. inkl diesem Feld)  (-) Nach Fehler erneute synchronisation schwer  **Byte Stuffing**: FLAG-Anfang und Ende. ESC.damit FLAG nicht in Nutzdaten vorkommt.  (+) einfache Sync-tion nach Fehler  (-) Platz Overhead    **Bit Stuffing**: Frame beginnt mit speziellen reserviertem Bitmuster(01111110).  Beim senden wird nach 5 1en immer ein 0 eingefügt und beim Empfang nach 5 1en immer ein 0 gelöscht.  (+) Framelänge muss kein Vielfaches von 8Bit sein.  Fehlererkennung und -korrektur  **Ursache**: Rauschen, Dämpfung, Verzerrung usw  **Fehlerkorrektur**: Keine F-korrektur (zu viel Redundanz)  Fehlererkennung: Fehler wird erkannt aber nicht behoben, Maßnahmen: Ethernet 802.3 keine Retransmission (nur durch TCP, falls Timeout eintritt), WLAN (durch LL(Active Repeat Request))  Checksumme    Cyclic Redundancy Check (CRC)  Generator G hat Länge r+1. Gesucht Rest der Längen r    Ethernet 802.3 Frames:  Das CRC Verfahren wird deshalb bei Ethernet eingesetzt, da es sich mittels Schieberegister sehr gut in Hardware implementieren lässt.    ***Präambel***:\*\* 7mal 10101010, dann 1mal 10101011  *\*\** Synchronisation Sender & Empfänger, Start desFram  ***Adressen***: je 6 Byte Sender & Empfänger MAC. Normalerweise, NIC leitet Frame nur an BS weiter, wenn destination Adr passt. Ausnahmen: Broadcast(dest.Adr FF:FF:FF:FF:FF:FF) oder Promiscuous Mode  ***Type***: 2Byte Art des Netzwerkprotokolls IPv4/IPv6 ***CRC***: 4Byte  **Eigenschaften**:  **Verbindungslos**: Kein Verb. Aufbau vor Datenaustausch  **keine zuverlässige Verbindung**: Frameverlust mögl. Absicherung durch höhere Schichten erfolgt  **Vielfachzugriff**: Nur bei Punkt-zu-Punk:Unsl CSMA/CD  Arten von Link:  \* Punkt-zu-Punkt(2kommunzier. Nodes) Ethernet LAN, das nur Switches verwendet und PPP für SONET und DSL  \* Mehrpunktverbindung(>2 kom Nodes): WLAN 802.11, Bluetooth 802.15, Klassisches Ethernet(Hub)  Vielfachzugriff  Geteilter Broadcastkanal. Interferenz == **Kollision** falls mehrere Stationen gleichzeitig senden.  **Multiple Access Control**: Algorithmus, der entscheidet, wann Host senden darf.Entscheidung muss **inband** getroffen sein (kein extra Kanal).  **Anforderungen**(Link hat Kapazität R):   * 1 Host sendet -> Host sendet mit Rate R * M Hosts -> jeder mit R/M(„Fairness“)   Klassifizierung von MAC:  **Multiplexverfahren**: Sender darf nur zur bestim t mit einer bestim Freq senden.  **Random Access Verfahren**: Kollisionen 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  Carrier Sense Multiple Access (CSMA)  Carrier Sensing==Mitlauschen auf dem Kanal.  *Kanal fre*i-> Beginne Übertragung. *Kanal beleg*t->Verschiebe Übertragung.  *Kanal belegt*:  1-persistent: Sende sobald Kanal wieder frei wird  p-persistent: Sende im nächsten Slot, mit Wahr-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.  Bei spät erkannter Kollision ist losgesendetes Paket wertlos.  dprop = t senden – t empfangen  CSMA/CD (Collision Detection)  **Kollision Erkennung**: Sender (NIC) hört während senden Medium weiter ab.  **Sofortiger** Abbruch der Übertragung & **Jam Signal** bei Kollision  Erneuter Sendeversuch nach **zufälliger** Wartezeit  **Binary Exponential Backoff**: Verdoppelung mittlerer Wartezeit nach jeder erneuten Kollision.  **Voraussetzung**: Sender fähig zu „listen while talk“.  Leicht im LAN: messen des empfamg Signals & vergleichen mit gesendetem -> Vollduplex  Schwer im WLAN: Empfangene Signale viel schwächer als gesendete -> Halbduplex  Sobald Sender Kollision erkennt: Frame wird nicht zu Ende übertragen -> Sender sendet Störsignal **JAM** umNachbarn ihre Sendertätigkeit anzustellen | Ethernet CSMA/CD: 1.NIC hört Kanal ab (frei->Frame wird gesendet; belegt-> NIC wartet bis Kanal frei ist) 2. NIC beobachtet ob Kollision auftritt(falls keine->erfolgreiche Übertragung) 3.Falls Kollision->Abbruch->JAMsenden. 4. Nach Abbruch:**Exponent Backoff**: mehr Kollisionen -> Backoff größer  Erneuter Sendeversuch nach **zufälliger** Wartezeit. • Binary Exponential Backoff: **Verdoppelung der mittleren Wartezeit** nach jeder Kollision  **Minimale Paketgröße** für verlässliche Kollisionserkennung:  Mindestlänge von Ethernet Frames  WC: 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: dtrans > 2dprop  Vielfachzugriff bei WLAN 802.11  CS: Wie Ethernet wird vor Senden Medium abgehört  ÐÐ°ÑÑÐ¸Ð½ÐºÐ¸ Ð¿Ð¾ Ð·Ð°Ð¿ÑÐ¾ÑÑ hidden collision problem**Kollision Erkennung nicht möglich**, weil: 1. WLAN ist **halbduplex**: empfang Signal sehr schwach 2. WLAN Stationen können sich nicht gegenseitig hören(**Hidden Station Problem:**  A hört bis zum B, C hört bis zum B  Beim B **(Empfänger) werden Kollisionen erkannt werden**  CSMA/CA (Collision Avoidance) bei WLAN  DIFS> SIFS Priorisierung von ACK, damit keine Verschachtelung  **Sender**: *1. FREI*: Kanal mind. für Zeitspanne DIFS frei -> sende kompletten Frame(ohne Carrier Sense) *2. BELEGT:* exponent Backoff **Unterschied zu CSMA/CD**  Höre Kanal ständig ab & **dekrementiere Timer nur** während Zeiten, wo Kanal frei ist. Erneute Übertragung, wenn Timer ausfällt  Falls **kein ACK** eintrifft -> Wieder in belegt Fall.->Ggfs backoff Intervall erhöhen.  **Empfänger**: bestätigt Datenempfang durch ACK nach Zeitspanne SIFS.    Switched Ethernet  **Hub**: Alle Leitungen quasi miteinander verbunden. Eine einzige Kollisionsdomäne(1Host gleichzeitig sprechen darf)CSMA/CD notwendig  **Switch**: Isoliert jeden Port in eigene Kollisionsdomäne ohne CSMA/CD  \*\*Jeder Host direkt mit Switch-Port verbunden.\*\*Keine Kollision möglich, falls Vollduplex.\*\*Kein CSMA/CD notwendig  Switches speichern Frames zwischen und leiten Frames weiter  Gleichzeitige Übertragung von A zu A’ und B zu B’ möglich  Switch: Forwarding  **Forwardingtabelle** enthält Info, an welchen Port ein Frame weitergeleitet werden muss:  Ziel MAC-Adr | Ziel Port | TTL  Selbstlernend: ankommender Frame: Eintragen von Port & MAC des Senders.  Nachschlagen, ob Eintrag für MAC Zieladresse 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.  **NETWORKLAYER**  Forwarding & Routing   * Ende-zu-Ende Beziehing * Sender verpackt ein Transport Layer Segment in Datagramm * Empfänger liefert das Datagramm an TransportLayer aus * Router interessiert sich nicht für Schicht 4/5 und kümmert sich nur um Weiterleitung zum Zielhost   Aufgaben von NL:   * Adressierung von IP * Forwarding * Zu welchem Ausganginterface muss ein Datagramm in Richtung Ziel * Bei Router oft in HW * Weiterleitung: Lok.Funktion jedes Routers (Navigation teilt Fahrer links oder rechts) * Routing (Wegewahl): * Berechnung Wege imn Netz * Eintragen der Ergebnisse in Weiterleitungtabelle * Routingprotokolle(konstruieren Routing Tabellen) * Mein in SW impl-t * Routingnachrichten zw Routern   **IP ist verbindungslos**  Zusammenarbeit von heterogenen Netzen  Auf einer oder mehreren Schichten Heterogenität aufweist  (+) IP ist Bindglied  (+) die LL kann unterschiedlich sein  Best efforts: keine Bevorzugung von Paketen + Jeder Router tut sein Bestes + keine Garantie bzgl Reihenfolge, Bandbreite  Architektur von Router   * Eingangsport: Pufferung ankommender IPPakete, terminiert LL * Fabrik: „Network innerhalb Router“ Weiterleitung zu passenden Ausgangsporten * Ausgangsports: Pufferung bis Übertragung möglich, LL/PL Funk-tät * Routingprozessor: Ausführung der Routingprotokolle     Queuing, falls Ankunftsrate schneller als Weiterleitung durch Fabrik.  **Bei IP wird Ausgangsport nur anhand der IP Zieladresse bestimmt!**  Forwarding-Table: **Problem**: viele Einträge, da IP 32 Bit => L**ösung**:  Aggregation / Hierarchie. In die Tabelle werden Adressbereiche eingetragen  Weiterleitung nach Zieladresse  Router leitet nach Bereichen weiter  Longest Prefix Matching(Präfix)   * Nachschlagen einer ZielIP (32Bit) in Forwarding-Tabelle des Routers * Suche längsten Adresspräfix, der mit der Zieladr übereinstimmt   Switching Fabrik: Aufgabe: Weiterleitung von Paketen vom Eingangs- zum Ausgangsport  **IDEAL**: Durchsatzrate == Ankunftsraten aller Eingangsports    Queueing am **Eingangsport Nötig falls Fabrik langsamer**  Problem vorderes Paket in Queue block-rt andere    Queuering am **Ausgang**: Nötig falls Ankunftsrate von SwitchFabrik > des Ausgangslinks  Ursache für Delay(Verzögerung) + Paketverluste  Paket hat keinen Platz mehr -> Paketverlust | Internet Protokoll IPv4  Adressing + PaketHandling convertions + Datagramm format  ÐÐ¾ÑÐ¾Ð¶ÐµÐµ Ð¸Ð·Ð¾Ð±ÑÐ°Ð¶ÐµÐ½Ð¸Ðµ  20 Bytes für IP-Header == **Overheads**  IP Fragmentierung  **Max Transfer Unit (MTU)** verschiedene LL Technologien erlauben verschiedene max Paketgrößen  **IP-Datagramm > MTU**:   * Fragment/Host zerlegt in kleinere Fragmente * Zusammenbau am Endhost * IP-Header Bits um Fragmente zu identifizieren und zusammenzufügen   Wiederzusammenfügung MTU = 1500 Datagramm = 4000  Header = 20  //Data  1500 -20=1480  1480%8=0 ->  1480/8=185  1480+20(Paket1)  1480+20(Paket2)  3980-1480\*2+20 (Paket3)  3980/1480(MTU-Header)  IP Adresse  32 Bit \*\* Identifiziert Host im Internet \*\* Gehört zu Interface.  Interface  \*Verbindung zw Host/Router und Link \*\* Router haben mehrere Interfaces \*\* Jedes Interface benötigt 1 IP Adresse  IP-Subnetz  Hosts teilen sich gleichen IP Adresspräfix \*\* Hosts können sich ohne Router erreichen(Bsp: Ethernet, WLAN,..)  Adresse eines IP-Subnetz  Subnetz ist über gemeinsamen Präfix adressierbar! \*\*Subnetzmaske: Länge des gemeinsamen Präfixes \*\*Hostanteil: Bits der IP Adresse, die sich für jeden Host unterscheiden.  **(+)man muss nur Subnetzadressen in den Rou-Tabellen halten**  BSP: 223.1.3.0/24 (die ersten 24 Bits sind für alle Hosts des Subnetzes gleich) NetzID 24Bit **Netzmaske** höheste 24Bits auf 1 setzen -> 255.255.255.0  Classfull 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)  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  BSP:172.16.37.186/26   |  |  | | --- | --- | | 1. Subnetzmaske | 255.255.255.192 | | 2. Subnetzadresse(Begin) | 172.16.37.128 | | 3.Boardcastadresse(Ende) | 172.16.37.191 | | 4. Number of Hosts |  | | AnfangsHostAdresse | 172.16.37.129 | | Letzte HostAdresse | 172.16.37.190 |   4 ohne 2 und 3  IPv4Unterteilung  /20->/23 23-20=3-> 2^3=8Organisationen  /23->/20->Umgekehrt 2^3 Subnetze  DHCP(Dynamic Host Configuration Protocol)  Manuell: ifconfig oder ip addr add • Persistent: /etc/network/interfaces  Server leitet automatisch IP aus Pool aus.  Host kann zugewiesene IP Adresse ggfs verlängern **SCHICHT 4**  **Funktionsweise**:   * Host sucht DHCP Server: **DHCP Discover** (optional)   Ziel: 255.255.255.255 (Broadcast)   * DHCP Server antwortet mit **DHCP Offer** (optional)   Ziel: 255.255.255.255 (Broadcast)   * Host forder explizit IP Adresse an: **DHCP Request** * DHCP Server weist Adresse zu: **DHCP ACK**   *Per DHCP lassen sich weitere Params konfig: DNS Server, Subnetz, IP des Gateway*  ICMP(Internet Control Message Protokoll)   * Error reporting * Router signalling   Austausch von Info zw Host und Routern:   * Bei Fehler sendet Router Fehlerbericht * Echo Request / Reply: Ping   ICMP wird als IP-Paket versendet + ICMP-Nachricht enthält TypeCode und die ersten 8Bits des IP-Pakets, das den Fehler verursacht  ARP(Address Resolution Protocol)  32-Bit IP: NL Adr für ein Interface \*\* Forwarding auf Schicht 3  48-Bit MAC: fest verbunden mit NCAdapter \*\* Verwendung: lokal, für Zuställung auf einem „Link“  **Aufgaben**:   * Aufm Weg zum Ziel muss IP über mehrere Links weitergeleitet werden * Jeder Router/Host schlägt Ausgangsport nach und leitet dann Paket weiter * Welche Ziel-MAC-Adr gehört zum Next-Hop? * Nötig: übersetzen von IP in MAC     **ARP Auflösung: IP zu MAC**  Jede IP-Knote verwaltet eine ARP-Tabelle (<IP>,<MAC>,<TTL>)  TTL(time zo live)- nachdem ungültig (oft nach 20Min)  ARP Tabelle:  IP | MAC  Routing Tabelle:  IP | Port  Switch forwarding Tabelle:  MAC | Port  **ARP direkt** auf Ethernet -> kein IP Header  Routing VS Forwarding  Routing: Berechnung der Routingtabelle für jeden Router \*\*Dazu tauschen Router untereinander Kontrollnachrichten aus (= Routingprotokolle)  Forwarding: Weiterleitung von Paketen zur Zieladresse mit **Longest Prefix Matching** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Routing  **Link State (zentral**): Jeder Router kennt komplette Topologie. Z.B. OSPF(OpenShirtestPathFirst) (über Dijkstra)  **Distance Vector (dezentral**): Jeder Router kennt nur direkten Nachbarn & nur ihre Kosten. Nachbarn teilen per Routingnachrichten mit welche Knoten sie mit welchen Gesamtkosten erreichen können. ASYNC z.B. Routing Information Protocol (RIP)).  **Statisch**: Manuelle Konfiguration von Forwarding Tabelle  **Dynamisch**: Periodischer Austausch von Routingformationen, Änderungen automatisch erkannt.   |  |  | | --- | --- | | Linkstate(LS) | Distanz Vector(DV) | | Routingsnachrichten | | | Jeder Router findet Infos über seine Links im ganzen Netz | Jeder Router informiert seine Nachbarn welche Ziele (mit Kosten) er erreicht | | Konvergenz | | | Berechnung auf jedem Router O(|E|\*|V|) falls binäre Heaps verwendet | Konvergenzgeschwindigkeit abhängig von der Reihenfolge des Nachrichtenaustautsches (Count-to-Infinity Problem) | | Robustheit (ein Router bösartig ist) | | | • Router kündigt falsche Linkkosten an.  • Fehler begrenzt, da jeder Router seine eigene Tabelle berechnet. | • Router kann falsche Pfadkosten ankündigen.  • Fehler pflanzen sich fort, da Tabelle eines Routers Einfluss auf andere Router hat. |   Hierarchisches Routing  gruppiere Router in „Autonome Systeme“ (AS) (aka "domains") Bsp: Deutsche Telekom, Deutsches Forschungsnetz  **Intradomain Routing**: Routing für Ziele im gleichen AS // Bestimmt welche externen Ziele über welches „Transfer-AS“ / Gateway erreichbar sind RIP, OSPF, IGRP  **Interdomain Routing**: Routing für Ziele in anderen Asen // Bestimmt, wie die Gateways zu den Nachbarnetzen aus dem lokalen Netz erreichbar sind BGP    Open Shortest Path First (OSPF)  Router fluten Link State Advertisement Nachrichten (enthalten Infos über alle Nachbarrouter) an alle anderen Router im gesamten AS (direkt über IP(nicht TCP oder UDP))  **Weitere Merkmale** \*\*Authentifizierung der OSPF Nachbarn. \*\*Linkgewichte sind konfigurierbar (nicht zwingend 1!) \*\*Lastverteilung möglich falls mehrere Pfade mit gleichen Kosten \*\*Hierarchisches OSPF für große Netze  Router lernt über welches Interface kommt er aus seinem AS raus zum Ziel. Das steht in seiner Routing Tabelle  BGP: Border Gateway Protocol  BGP Session: TCP zw 2 BGP-Router + jeder R teilt mit, welche Ziele (IP-Präfix) er kennt  iBGP(zw R-ten im gleichen AS)  eBGP(zw R-ten in benachbarten Asen)  **BGP Attribute**  \*\***AS-PATH**: Liste von ASen, durch die das Prefix Advertisement gelaufen ist. \*\***NEXT-HOP**: IP Adresse des Gateway Routers  Policy-based Routing  \*\*BGP Router verwendet Import-Policies um einen Pfad zu akzeptieren oder abzulehnen\*\* Bsp1: Ignoriere Pfade durch AS Y Bsp2: Gib Routinginfo nicht an Nachbarn AS X weiter    Ein Router kann mehrere alternative Routen für einen Ziel-Präfix lernen  Aufgrund der konign Policy ("wähle immer kürzeren ASPfad") entscheidet sich R 1c für Pfad AS3, X und kündigt nur diesen Pfad über iBGP intern im AS an  Die beste Route wird nach folgenden Kriterien gewählt:   * Local Preference(Zuweisung von Prio beim Import) * Kürzer AS-Pfad (Route, die der man am wenigsten Asen durchqueren muss * Route mit dem am schlechtesten erreichbaren Next-Hop (=Gateway) Hot-Potato   Hot Potato Routing   * Jedes Netz möchte Pakete so schnell wie möglich aus eigenem AS/Netz loswerden. * Wähle lokales Gateway mit den geringsten Intradomain-Kosten   IPv6  **HEADER 40 Byte**  **Notation**: 128 Bit in 8 Blöcke, je 16 Bit (4Hex Zahlen), mit **:** getren  - Führende Nullen darf man weglassen  - Nur einmal dürfen ein oder mehr aufeinanderfolgende Blöcke mit 0000 ausgelassen werden und durch :: ersetzt werden.  Bsp: 2001:0db8:0:0:0:0:1428:57ab 2001:db8::1428:57ab  IPv4: ::192.31.20.46  Es gibt keine /80 Subnetze, wel Host-ID immer genau 64 Bit.  **Unterschiede zu IPv4**: \*\***keine Fragmentierung**(R informiert Sender per ICMPv6, dass Nachricht zu groß) \*\*Jeder IPv6Host verfügt automatisch über **Link local IPv6 Adr** (abgeleitet von MAC und nur im lokalen LAN gültig) \*\***Kein ARP**  **Tunneling**: Tunnel (IPv6 Datagramm wird in den Nutzdaten eines IPv4 Pakets transpot-t ( mit IPv4-Lagency-Leitung) +++ Dual-Homed: die Geräte müssen IPv4 & IPv6 sprechen  **TRANSPORT LAYER** ist Teil des BSs  Kommunikation zw Prozessen auf Sender- Empfängerseite  **Betrifft Hosts, nicht Router**! **Sender**: Unterteilt Nachricht in Segment, Weitergabe zur Network Layer **Empfänger**: Baut aus Segmenten Nachrichten zusammen, Weitergabe zur Anwendung  UDP Ein minimales NWProtokoll. ermöglicht Anwendungen den Versand von Datagrammen in IP-basierten Rechnernetzen. **Transport-Layer Multiplexing**: Ordnet IP Pakete Prozessen des  BSs zu. **Fehlererkennung** – Checksumme. Kaum Mehrwert zu IP  + Paketverluste möglich + Keine Einhaltung der Reihenfolge  TCP UDP +   * **Verbindungsorientiert**(Verbind-aufbau vorher) * **Zuverlässig Bestätigt einzelne Bytes, keine Pakete!** (Feedback bekommen: keine Übertragungsfehler, Paketverluste, Veränderung der Reihenfolge. Bestätigt einzelne Bytes, keine Pakete! Problem: unzuverlässigen Kanal) * **Flow Control** (Vermeide Überlastung des Empfängers) * **Congestion Control**(des NWs) **UDP und TCP: Keine Garantie bzgl. Delay und Bandbreite**! * **Vollduplex**(**MSS** (=Maximum Segment Size) richtet sich nach **MTU** (= Maximum Transfer Unit) der Link Layer) * **Pipelining**(Mischung aus Go-Back-N und Selective Repeat)   Transport-Layer Multiplexing(UDP)  **Socketadresse** == Portnummer, **DestPort** muss **bekannt** sein, **SourcePort** beliebig | |  |  | | --- | --- | | UDP | TCP | | verbindungslos (nur Dst IP/Port, Src IP/Port spielen keine Rolle) | verbindungsorientiert (beide Dst und Src IP/Port notwendig) | | Segmente mit gleichem Dst an gleiches Socket | Segmente mit gleichem Dst und Src (alle4) an gleiches Socket |   **Sequenznummern (Seq) zählen Bytes und nicht Pakete!**  **Acknowledgment (ACK) - Seq des next erwarteten Bytes von der Gegenseite (entspricht dem ältesten, noch nicht empfangenen Byte**  **BidirektKom-tion Seq in einer Richtung sind ACK- in der anderen**  P: Einzelne Bits in den Paketen können während Übertragung umkippen  L: Checksum, CRC, Parität + ACK NACK (Retransmission RTS) Stop-and-Wait  P: ACK/NACK Paket corrupt  L: Checksumme auch für ACK/NACK Pakete. Nur ACK. 2 gleiche Seq hintereinander – Duplikat -> NACK braucht man nicht mehr  P: Braucht man NACKs?  L: Seq beim ACK + Seq muss gleich dem PaketSeq sein 2gleiche ACK - RTS  P: Daten und/oder ACK Paket geht komplett VERLOREN  L: Timeout: Falls Sender kein ACK innerhalb Timeout – RTS. Beim Empfänger werden Duplikate erkannt und verworfen. Beim Sender nach dubliertem ACK passiert nix – Sender tut gar nix  P: **Stop-and-Wait**: Sender muss mindestens die **Round Trip Time (RTT -** von Quelle zum Ziel und zurück**)** abwarten, bevor er das next Paket sendet  L: Pipelining: mehrere Pakete gleichzeitig senden: dazu max Anz den Paketen, die gleichzeitig los (*Empfänger NICHT überlasten*) +  **Seq** notwendig!!!  Pipelined Datenübertragung   |  |  | | --- | --- | | Go-Back-N | Selective Repeat | | Sender kann bis zu N unbestätigte Pakete in der Pipeline (Sendefenster) haben | | | Empfä-r send kumulative ACK  Pakete mit Seq 𝑠 ≤ S wurden schon empf-en Sender weißt welches Seq Empf braucht | Empfä-er send individuelle  ACKs für jedes Paket | | Timer für das älteste unbestätig Paket: RTS aller aktuell unb Pak-te | Timer für jedes unbestätig Paket  RTS nur des betreffenden unb Pak |   TCP: Hybrid von Go-Back-N und Selective Repeat  **Go-Back-N**   * Kumulative ACKs: Es wird **nicht jedes** Segment einzeln bestätigt. * **Nur 1 RTS-Timer**, der sich auf ältestes noch unbestätig Segmen bezieht   **Selective-Repeat**   * Alle TCP Empf-er haben einen Empfangspuffer und merken sich Pakete,selbst wenn ältere Pakete noch ausstehen (на очереди) * Bei Timeout wird nur das verlorengegangene Paket erneut gesendet   TCP Fast Retransmit   * Timeout ist oft relative lang * 3 Duplicate ACK == Paketverlust * Nach 3m DuplACK RTS * RTS des ältesten, unbestätigten Segments   TCP: 3-way Handshake TCP Flags: SYN, ACK  TCP Verbindungsabbau Bei Empfang eines FIN-Pakets: Bestätige Empfang mit ACK + Schicke ebenfalls eine FIN Nachricht    Anpassung der TCP Senderate  Der Sender muss seine Geschwindigkeit verringern falls:   * Empfänger nicht schnell genug ist → Flow Control * Netzwerk nicht schnell genug ist → Congestion Control   Flow Control  P: Anwendungsprozess im Empf-er holt Daten zu langsam bzw. zu spät aus TCP Empfangspuffer ab  L: Sender-Reaktion - Angleichen der Senderate an Verarbeitungsrate des Empfängers + Empf-er sagt Sender wie viel Bytes er schicken soll bevor Puffer überläuft:   * Empfänger teilt **rwnd** dem Sender mit * **rwnd** ist oberste Schranke für Sender | Congestion Control (dt. Staukontrolle)  P: 1. Großes Sendefenster → mögliche Überlastung des Netzwerks!  2. Kleines Sendefenster → geringe Datenrate  **Anpassung der Senderate**:  *Netzwerk-unterstützt*  • Router geben Feedback an Hosts bei Überlastung  • Neuer seltener Ansatz: Explicit Congestion Notification (ECN)  *Ende-zu-Ende*  • Auslastung des NW wird durch Beobachten der Verzögerung und Auftreten von Paketverlusten abgeschätzt.  • **TCP wählt diesen Ansatz**!*Additive Increase*: Vergrößere nach jeder RTT Congestion Window (cwnd) um 1 MSS(tcplinkunt) bis Paketverlust erkannt wird  *Multiplicative Decrease*: Halbiere cwnd nach erkanntem Paketverlust  **Senderate begrenzen**: Sendefenster **≤ min{cwnd, rwnd}**  TCP Reno  TCP Sender erkennt „Congestion“: Timeout + 3 Dupl ACKs  **Sender-Verhalten**:  **Slow Start / exponentiell**: Verdoppele cwnd nach jeder Round Trip Time  bis zu einem gewissen Schwellwert (**ssthresh -** Grenze zw Slow Start und Congestion Avoidance)  **Congestion Avoidance / linear**: Dann vergrößere cwnd um 1 nach jeder  Round Trip Time  **Bei Paketverlust / Problem**: Halbiere cwnd  **Nach Timeout**  - TCP Slow Start  - ssthres=cwnd/2  - cwnd = 1  **Nach 3 Dupl ACKs**  - Congestion Avoidance  - Hinweis, dass Netz zumindest  nicht vollständig überlastet ist.  - ssthres = cwnd/2  - cwnd = cwnd/2 + 3NAT: Network Address Translation  NAT Table LAN(Source IP, Port #) zu WAN(NAT IP, neue Port #)  Lokales Netzwerk benutzt nur 1 IP: (+)Einsparen von IP (+)IP im lokalen NW ändern ohne anderen zu informieren (+)ISP(ServiceProv) wechseln ohne IP zu ändern (+)Geräte in einem lok NW sind direkt adressierbar  **Implementierung eines NAT Routers**  ***Senden ins Int:*** Ersetze (Source IP, Port #) durch (NAT IP, neue Port #). Ziel von entfenten: (NAT IP, neue Port #).  ***Empfangen aus Int:*** Schlage in NAT Translation Table nach ++ Ersetze (NAT IP Adresse, neue Port #) jedes ankom-den Datagramms mit (Source IP, Port #)  16-Bit für Ports (> 60000) gleichzeitige Verbindungen mit 1 öffentlichen IP  Routern sollten die Schicht 4 (Portnummer) nicht berücksichtigen!  Relaying: Skype beide Cleients verbinden sich mit Relay  **DNS (Domain Name System) APPLICATION LAYER(AL)**  Aufgaben   * **Übersetzung Hostname** → IP Adresse * **Host Aliasing**: Host kann mehrere Namen haben, Übersetzung(Canonical Name: relay1.west-coast.enterprise.com -> Alias Name: [www.enterprise.com](http://www.enterprise.com)) * **Mail Server Aliasing**: MX Record: Speichert Canonical Name des Mail Server für eine Domain * **Load Balancing**: • Replizierte WebServer: Viele IP haben den gleichen Namen • Antwort des DNS Servers bestimmt, welcher physikalische Server / IP verwendet wird.   Komponenten   * Verteiltes Verzeichnis: Hierarchie von Name Servern * Protokoll der AL: Hosts und DNS Server kommunizieren miteinander + Wichtige Internetfunktion wurde in der AL impl-t + Prinzip: „Komplexität am Rande des Internets“   **kein zentralisiertes DNS**, da zu hohes Verkehrsvolumen + Server eventuell sehr weit von anfragendem Host entfernt → hohe RTT für DNS Anfragen  ÐÐ°ÑÑÐ¸Ð½ÐºÐ¸ Ð¿Ð¾ Ð·Ð°Ð¿ÑÐ¾ÑÑ DNS: Hierarchischer Namensraum  **Root** kennt alle, **Top-Level** kennt alle mit .com, z.B, Second-Level – **Authoritative**(DNS Server einer Domain. Kennt IP von Hosts, seine Auskunft ist verbindlich), **Resolver**(Stellt DNS Anfragen an Root/TLD/Authoritative. Stellt Anfragen im Namen von Hosts („Proxy“) Speichert selber keine verbindliche Info, nur Caching! Häufig vom ISP bereitgestellt)  Einträge == Resource Records (RR)  RR Format: (name, value, type, ttl)  **Type**: A-IPv4, AAAA-IPv6, CNAME-Alias(Canonical/DomainName) kein Typ mehr mit diesem Namen, MX-MailServer, NS-NamevomServer für dieses Domain  DNS-Anfrage   * **autoritativ** (der Server holt die Daten aus einer lokalen Zonendatei) * nicht-autoritativ * **rekursiv** (der Server holt die Daten von einem anderen Nameserver) * Jeder Server holt rekursiv die Daten vom nächsten Name Server * Hohe Last auf den höheren Hierarchieebenen! * **iterativ** (der Server antwortet mit einem oder mehreren Verweisen oder einem Resource Record auf andere Nameserver) * Resolver hat Antwort nicht im Cache * Kontaktierter DNS Server kann Namen nicht auflösen * Er informiert aber über den „nächsten“ Name Server, der kontaktiert werden muss     DNS Caching  TTL bleibt ein Eintrag im Cach danach muss geprüft werden. IP der TLD Server sind immer im Cache des Resolvers. **Änderung von DNS Einträgen** (DNS Update). Einträge können sich **veralten**  DNS Protokoll: Aufbau der Nachrichten   * *Query* und *Reply* haben das gleiche Format * Identification:16 Bit Zahl: Identisch für *Query* und zugehörige *Reply* * Flags (Query oder Reply? ++ Rekursive Query erwünscht ++ Rekursion verfügbar ++ Antwort ist „authoritative“) * Query: Name und Typ * Antworten auf eine Query (Ggfs. mehrere IP für einen Namen)   Registriere networktec.de:  (networktec.de, dns1.networktec.de, NS)  (dns1.networktec.de, 212.212.212.1, A)  **WEB-REQUEST**   1. **Verbinden mit dem Internet**  * Laptop erhält per DHCP IP für sich selbst + IP des Gateways + IP des DNS Servers * Aufbau des DHCP Pakets mit UDP + IP + 802.3 Ethernet * DHCP arbeitet mit Ethernet Broadcasts * DHCP Server sendet DHCP ACK mit IP für Client + IP des Gateways + IP des DNS Servers | 1. **DNS**   **2.1 ARP**  Vor dem Senden des HTTP Requests wird  die IP Adresse von [www.google.com](http://www.google.com)  benötigt: DNS   * DNS Paket: UDP + IP + Ethernet * Um Paket zum Router/Gateway zu senden wird   MAC des Router Interfaces benötigt: **ARP**  (ARP Query Broadcast wird von Laptop gesendet >>  Router/Gateway antwortet mit ARP Reply)  **2.2 Verbindung mit DNS Server**   * IP Datagramm mit DNS Query wird über Switch zum Gateway weitergeleitet * IP Datagramm wird vom Heimnetz in das Netz des ISPs und zum DNS Server weitergeleitet(Route über BGP, RIP, etc.) * DNS Server antwortet mit IP Adresse von [www.google.com](http://www.google.com)      1. **TCP Verbindungsaufbau**   Zuerst ein TCP Socket zum Web Server erzeugt werden (TCP SYN Segment wird zum Web Server geleitet >> Web Server antwortet mit TCP SYNACK)   1. **HTTP Request / Reply**  * HTTP Request wird in TCP Socket gesendet * IP Datagramm wird zum Server geleitet * Web Server antwortet mit HTTP Reply * IP Datagramm wird zurück zum Client geleitet   **EINFÜHRUNG**  Messen von Internet-Routen mit Traceroute  Weg vom Starthost zum Zielhost messen  Für alle i:   * Sende Pakete die nur i laufen können richtung Ziel * Der i-te Router sendet Pakete zum Sender zurück * Sender lernt alle Router kennen – vgl. Breitensuche   Leitungsvermittlung (Circuit Switching)  Benötigte **Ressourcen** müssen vorab **reserviert** werden  Verbindung wird zugelassen, g.d.w *ausreichend Netzkapazität*  Dann Senden eines kontinuierlichen Datenstroms.  **Übertragungsrate ist garantiert**. **Ressourcen werden verschwendet**  Paketvermittlung (Packet Switching)  Host teilt Nachricht in kleine **Pakete der Länge L** auf und schickt sie unabhängig voneinander los.  Pakete reisen **unabhängig** über Links & Router in Richtung Ziel  **Jeder Link** mit Max **Übertragungsrate(R**): Gleichzeitige Pakete müssen sich einen Link teilen und zeitlich hintereinander gesendet werden.  **Jeder Router**: **Store-and-Forward**. Jeder Router muss gesamtes Paket empfangen bevor er das Paket auf den ausgehenden Link weiterleitet  Paketverlust und Paketverzögerung   * Puffering verursacht Delay(**Verzögerung**) * Puffer: ankom-de Pakete werden verworfen falls Puffer nicht frei Packet Loss (**Paketverlust**)   TCP/IP Schichtenmodell   1. **Application Layer → Message**(FTP, SMTP, HTTP, DNS)   4. **Transport Layer → Segment**(TCP, UDP)  3. **Network Layer → Datagramm**(IPv4, IPv6, X.25)  2. **Data Link Layer → Frame**(Linktechniken: Ethernet, WLAN, PPP)  1. **Physical Layer → Bits**(Frequenzen, Modulation, Spannungsverläufe + Übertragungsmedien (Kupfer, Glasfaser, Funk)  Das ISO/OSI Schichtenmodell   1. Presentation Layer: Semantik der übertragenen Kommunikation (Bsp: Kompression, Verschlüsselung, Big Endian vs, Little Endian) 2. Session Layer: Sitzungsaufbau- und Sitzungsabbau + Synchronisierung zwischen beteiligten Prozessen   Vorteile (+) Herunterbrechen der Komplexität. (+) Jede Schicht hat klar definierte Aufgaben, Datenstrukturen und Steuerungsmechanismen (+) Transparenz: Schicht muss nicht wissen was in höherer oder tieferer Schicht genau passiert. (+) Eine Schicht kann geändert werden, ohne Einfluss auf andere Schichten  Nachteile: (--)Leichter Overhead (--)Gleiche Aufgabe in mehreren Schichten implementiert (Fehlerkorrektur auf Schicht 2 und Schicht 4) (--)Höhere Schicht benötigt Information aus niedrigerer Schicht  **Routing Tabelle**: Linux: route Windows: route print  **MAC-Adresse**:   * Windows: ipconfig /all -> physische Adresse * Linux: ifconfig -> ether   **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  **Traceroute**: Windows: tracert Linux: traceroute  **MAC ändern**: Windows: ipconfig/release -> ipconfig/renew NEW\_MAC  Linux: ifconfig <eth0> down -> ifconfig<eth0> hw ether NEW\_MAC -> ifconfig <eth0> up  **Resoursen verfügbarkeit:** Windows: ping IP-Adr oder ping DOMAIN  Wireshark: frame.nummer<90 |