Grundbegriffe

ISO/OSI – Modell: International Standardisation Organisation / Open System Interconnection
Merksatz für engl. Bezeichnung: "All People Seem To Need Data Processing"

- → 7 Application --> Aufsetzen von Protkollen für eMail, www, Multimedia, Verschlüsselung,..
  - Telnet, FTP, SMTP, DNS
- → 6 Presentation --> Syntax & Semantik: Codes, Zahlformate, Datumsformate, Datenstrukturen virtuelles Terminal
- → 5 Session --> Fixpunkte der Übertraung, Token Management (MutEx), Sitzungsaufbau
- → 4 Transport --> Auf/Abbau von Verbindungen, Flow Control, QoS
  - TCP, UDP
- → 3 Network (Vermittlung) --> Routen, Protokollanpassung, Adressierungsfragen, Tunneling, Überlastungsüberwachung
  - Router, IP, ICMP
- → 2 Data Link (Sicherung) -> Data Frames, ACK, garantiert Fehlerfreiheit, puffert, synchronisiert
  - Bridges, ARP (Address Resolution P), ARQ (Automatic Repeat Request)
- → 1 Physical
  - Verkabelung, Repeater, Async, Sync

TCP/IP Referenzmodell:
- 7 Application
- 4 Transport
- 3 Internet
- 1+2 Host-to-Network

Tanenbaum'sches hybrides Modell fasst nur die Ebenen 5,6,7 auf eine Application-Ebene zusammen, die unteren Ebenen 1-4 bleiben separat, da physische Details wie Signalschwankungen etc nicht in Fehlerkorrektur einfließen sollten.

Hub / Repeater
- Layer 1
- nicht mehr als 4 Repeater zwischen 2 Stationen
- keine Verknüpfung unterschiedlicher Ethernetstandards
- trennt Netzwerk **nicht** in Collisondomains auf

Bridge / Switch
- Layer 2
- trennt Netzwerk in Collisiondomains
- leitet Broad- und Multicasts weiter (außer bei VLANs, dann nur ins richtige VLAN-Segment)
- leitet Frames nur weiter, wenn es nötig ist
- um Zyklen durch Redundanzen zu verhindern --> Spanning Tree Protocol
- Die Forwardingtabelle speichert MAC-Adresse, Ausgangsport und Zeipunkt, kommt ein Frame dessen Ziel bekannt ist wird er an den Port weitergeleitet, wenn nicht an alle Ports außer den Eingangsport verschickt.
- Aging: nach einer gewissen Zeit vergisst die Bridge Einträge wieder.

Router
- Layer 3
- trennt Netzwerk in Broadcastdomains
- Routingtabellen für die Wegwahl
- kann zwischen verschiedenen Netzwerken routen (versch. Protokolle, Netzwektypen)

Topologie beschreibt die Stuktur eines Netzes (Stern, Token-Ring, Bus)

Verzögerungen in paketvermittelten Netzen:
- Verarbeitungsverzögerung d_proc
- Warteschlangenverzögerung d_queue
- Übertragungsverzögerung d_trans
- Ausbreitungsverzögerung d_prop

neue 80-20 Regel besagt: früher 80% lokal, 20% dezentral ausgeführt, heute ist es umgekehrt.

Die Sicherungsschicht (Data Link):

Auf Wunsch kann auch Fehlererkennung implementiert sein, bei Wireless eher wichtig als bei sicheren Fiber-kabeln.
Manchester-Kodierung zur Erhöhung der Zuverlässigkeit: 1 = LO_HI, 0 = HI_LO. Verbraucht allerdings doppelte Bandbreite.

wichtigste Aufgaben:
- Unwandlung von reiner Übertragungsleitung in garantiert fehlerfreie Leitung aus Sicht der oberen Schichten
- Aufteilung des Bitstroms in Frames
- Bestätigungsframes erzeugen (ACK)
- Fluss-Steuerung: Pufferung, Synchronisation

character stuffing: tritt bei Aufteilung in Frames auf: kommen Rahmenmarkierungen o.ä. im Text vor werden sie verdoppelt (DLE STX / ETX)
bit stuffing: jeder Frame endet / beginnt z.B. mit 01111110, das heißt in der Nachricht muss immer nach 5 Einsen eine 0 gestopft werden.

Fehlererkennung / Fehlerbehebung:
Hamming-Distanz zweier Wörter ist die Anzahl der Bits an denen sie sich unterscheiden.
Auffinden von d Einzelfehlern: Hammingdistanz von d+1
Korrigieren von d Einzelfehlern: Hammingdistanz von 2d+1

Gegeben: Nachricht mit m Nachrichtenbits.
Gesucht: Code mit r Prüfbits, sodass alle Einzelfehler korrigiert werden können.
$r > ld(m+1+r)$ .. Ungleichung muss erfüllt werden.

CRC – Cyclic Redundancy Code – Polynom Code
Der zu sendende Rahmen wird als Polynom gedacht und durch r Bits ergänzt, sodass das Ergebnispolynom durch das Generatorpolynom G(x) teilbar ist. Der Empfänger muss nur die empfangene Nachricht durch G(x) dividieren, wenn 0 Rest bleibt war die Übertragung erfolgreich.
die r Kontrollbits bekommt man indem man die Nachricht so lange durch G(x) dividiert bis ein Rest übrigbleibt, dieser Rest = r

Protokolle der Sicherungsschicht:

HDLC – High Level Data Link Control
- rein bitorientiert, Felder: Address, Control, Data (beliebig lang), Checksum

SLIP – Serial Line Protocol
- rohe IP-Pakete mit Flag 0xC0 zu Schluss werden versendet
- IP muss im vorhinein bekannt sein
- keine Authentifizierung

PPP – Point to Point Protocol
- → verbindet genau 2 Partner
- → verschickt werden Datagramme der Ebene darüber (L3)
- → 3 Merkmale:
  - Framing: Rahmenerstellung, Ende/Anfang gekennzeichnet, Fehlererkennung
  - LPC – Link Control Protocol: für Auf/Abbau von Verbindungen, Verhandeln von Options
  - NCPs – Network Control Protocols: Protokollreihe die je ein höherschichtiges Protokoll enthält

LCP – Link Control Protocol
- wird nach Verbindungsaufbau aufgerufen, verhandelt Optionen und führt zum Authentifizieren
- z.B. Größe des Rahmens, Protkolltyp, Header-Kompressionsverfahren,.

NCP – Network Control Protocol
- nach erfolgter Authentifizierung wird das darüberliegende Protokoll der Vermittlungsschicht konfiguriert

L2TP – Tunnelprotokoll
- ermöglicht sichere Verbindung eines mobilen Clients über eine Wählleitung zu einem ISP
- Client wählt sich zum Ortstarif beim ISP ein, von dort getunnelt zur Ziel-IP

(R)STP – (Rapid) Spanning Tree Protocol
- Redundanzen von Bridges/Switches (L2) schaffen Probleme, z.B.: Broadcaststorm
- --> Redundanzen müssen weg (Spanning Tree), aber bei Ausfall dynamisch hinzugefügt werden
- aus den gegebenen Knoten ist ein Gerüst (spannender Baum) zu erstellen
- mit Kantenbewertung kann ein minimales Gerüst erstellt werden.
  - nimm die kürzeste Kante, dann füge die kürzeste anschließende Kante hinzu so dass sich kein Kreis bildet bis alle Punkte erreicht sind.
- Aboreszenz (Wurzelbaum)
  - ein Graph ist eine Aboreszenz wenn er ein Baum ist, eine Wurzel besitzt und jeder Knoten T kreisfrei erreichbar ist.
  - Minimalität: 1) Summer der Kantenbewertungen minimal oder 2) Summe der Weglängen von jedem Knoten x zum Blatt minimal (wird bei Dijkstra-Algo und STP verwendet)

- Dijkstra-Algorithmus / STP-Funktionsweise
  1. Auswahl des Wurzelknotens (wichtig: gute Wahl durch richtige Prioritäten des Admin)
  2. jede Bridge vergleicht auf ihren Ports den Weg zur Wurzel, der kürzere gewinnt.
  3. Voraussetzung: Weg ist die Wurzel und eigener Abstand zu ihr muss durch das Netz propagiert werden.
  - es findet **kein** Lastenausgleich statt

MAC- Teilschicht (Media Access Protocol)

es geht um: Broadcast-Netze und Mehrfachzugriffsprotokolle

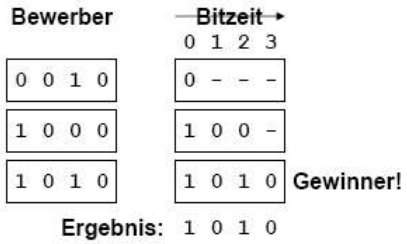
Dynamische Kanalzuordnung:

1. Stationenmodell (jede der N Stationen ist so lange blockiert bis ein einmal erzeugter Frame korrekt übertragen wurde)
2. Einzelkanalannahme (Alle Stationen teilen sich einen Kanal --> Multiple Access)
3. Kollisionsannahme (Wenn mehrere Frames gleichzeitig übertragen werden --> Kollision)
4. Zeitunterteilung (Continuous -> Übertragung jederzeit / Slotted -> nur zu best. Zeitpunkten)
5. Träger (Trägererkennung: Stationen können erkennen ob eine Übertragung im Gange ist, falls ja warten sie mit dem Senden)

- Kriterien für dyn. Kanalzuordnung
  - Effizienz, gerechte Aufteilung, Verfügbarkeit/Fehleranfälligkeit, Determinismus, Rahmengrößen (fix/flexibel),..

Kollisionsfreie Protkolle:

- Bitmusterprotkoll: Anmeldung von Konkurrenzschlitzen aller N Stationen hintereinander, dann dürfen die Angemeldeten je einen Frame übertragen.
- Binärer Countdown: Konkurrenzschlitze werden parallelisiert, jede Station sendet ihre ID und horch mittels OR mit. Wenn auf der Leitung eine höhere ID anliegt als die eigene gibt man auf.

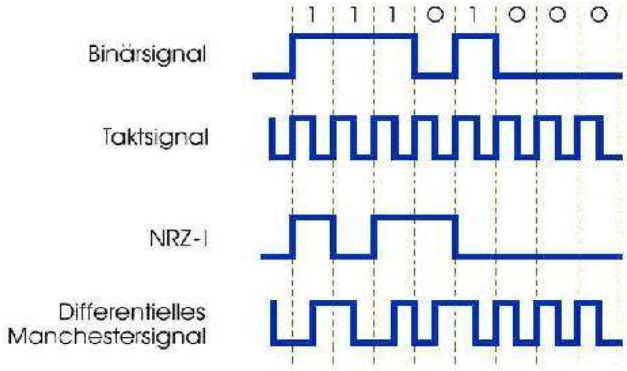


Mehrfachzugriffsprotokolle mit Kollisionen:

- reines/slotted ALOHA, CSMA, Ethernet (IEEE 802.3)

Token Ring:

- Ein Token kreist durch das Netz, nur die Station, die das Token besitzt darf auch sprechen.
- Verhalten des Systems bei Tokenverlust etc sollte gut durchdacht sein
- verwendet Differentielle Manchester-Kodierung
  - sie ergibt sich aus der XOR Verknüpfung des Taktes und einem Signal, das bei jeder zu übertragenden 1 invertiert wird (NRZ-I – Non Return Zero-Inverted).
    Das heißt bei jeder 1 die in der Nachricht vorkommt wird das in der Manchester-Kodierung verwendete Schema "1 = LO_HI, 0 = HI_LO umgedreht!



Ethernet:

Ethernet betrifft OSI Layer 1 und 2, von Layer 1 Media Specifications und Physical Signaling Sublayer, von Layer 2 jedoch nur die MAC-Sublayer. Logical Link Control gehört nicht mehr dazu.

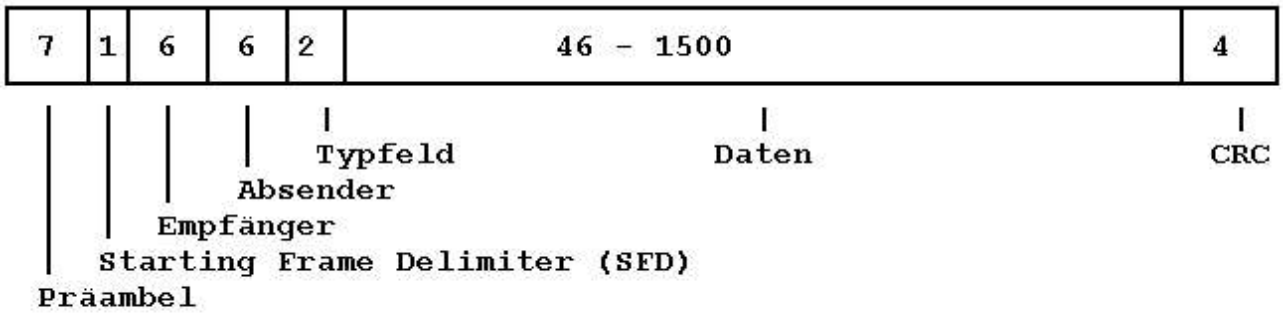
IEEE Bezeichner:

- 10BASE5 (thick wire ethernet) 10MBit, Basisband-Übertraung, max 500m Kabellänge
- 10BASE2 (thin wire ethernet) 10MBit, Basisband-Übertragung, max 200m Kabellänge
- 100BASETX 100MBit, Basisband-Übertragung, Twisted Pair-Kabel (CAT-5)

verschiedene Topologien:

- Bus-System: alle Stationen sind an einem Bus angeschlossen
- Multisegment-Bus-System: ein Repeater o.ä. verbindet mehrere Busse (ausfallssicherer!)
- logischer Bus, physikalischer Stern: Bus wird repräsentiert durch Hub
- Stern: ein Switch verbindet alle Clients
- Baum: wie Stern, nur noch mit einem zusätzlichen Uplink --> Baumstruktur

Aufbau eines Ethernet-Frames (Angaben in Oktets):



Präambel diente bei 10Mbit der Synchronisation zwischen Sender und Receiver, bei 100Mbit noch aus Kompatibilitätsgründen vorhanden. (wegen constant signalling braucht man bei 100Mbit keine Synchronisation mehr)

Empfänger:
- 1. Bit: 0 --> Individual 1 --> Group Adress
- 2. Bit: 0 --> Globally Administered 1 --> Locally Administered

Bei MAC-Adressen sind die ersten 24 Bit herstellerspezifisch, das heißt jeder Hersteller bekommt einen oder mehrere Adressräume (OUI) und bestimmt die letzten 24 Bit selbst. MAC-Adressen sind weltweit eindeutig.

CRC: Frame Checksum mittels CRC32-Polynom, wenn falsch wird der Frame verworfen.

Nach dem CRC werden noch maximal 7 Dribblebits angefügt, da das Interface bei 10Mbit die Daten bei einem vielfachen von 8 abschneidet, kommt _meiner Meinung_ nach also vor, wenn die Datenmenge kein Vielfaches von 8 ist.

Es gibt noch eine spezielle Form der Ethernetframes die zwischen Absender und Typfeld 4 Oktets VLAN-Informationen einfügen. Dabei sind die ersten 2 Oktete der TPID (Tag Protocol Identifier), die letzten 2 TCI (Tag Control Information). Diese Zusatzfelder werden genutzt für Angabe von Paketprioritäten und die Erstellung von virtuellen LANs in einem logischen LAN. Dies muss natürlich von allen Komponenten im Netzwerk unterstützt werden.

Filterung der Pakete durch NIC (Network Interface Card):
1. Prüfung der Checksum, wenn falsch sofort verwerfen
2. Check der Destination Address
3. Überprüfung des Datenteils auf 'magic packets' z.B.: für Wake On LAN

Vorgänger von Ethernet:
ALOHA: 2 Versionen, reines und slotted. Beim reinen ist es jederzeit möglich Frames zu versenden, bei der slotted Version nur in Zeitschlitzen, was die Anzahl der Kollisionen deutlich verringert

CSMA/CD:
CS = Carrier Sense: Sprechen ist erlaubt wenn zumindest kurz Ruhe war.
MA = Multiple Access: Wenn diese Ruhezeit vorbei ist haben alle die selbe Chance zu reden.
CD = Collision Detection: Wenn mehrere Clients gleichzeitig sprechen wird das erkannt und aufgehört zu reden.

Bei Ethernet wird wenn eine Collision entdeckt wird ein JAM-Signal gesendet (außer bei Late Collisions) und der Frame verworfen.

Effizienz:
siehe K04b_Ethernet.pdf Seite 43 – 52.. ich baue auf die 'Menschlichkeit des FIMs', dass das _nicht_ kommt.

maximale Größe eines Kollisionsbereichs:
Die maximale Ausdehnung eines Kollisionbereichs wird durch die Ethernet-Roundtrip-Time bestimmt. Normalerweise erkennt eine Station die ein Kollision auslöst und erkennt ein JAM-Signal, der Sender der ersten Nachricht bekommt dieses JAM-Signal und schickt den Frame nochmals. Wenn jedoch das JAM-Signal aufgrund einer zu großen Ausdehnung das JAM-Signal zu spät bekommt denkt er sich, es betrifft nicht ihn, da für ihn der Sendevorgang schon abgeschlossen ist.

Berechnung der maximalen Kabellänge:

$$\frac{(\ddot{U}bertragungsgeschwindigkeit(bits/sec))}{(Kabelgeschw(m/sec))} * 2(roundtrip) = x(bits/m)$$

$$\frac{(Paketgröße(bits) - Zeit\ für\ Elektronik(bits))}{(x(bits))} = max.\ Kabellänge$$

Beispiel: (Kabelgeschwindigkeit = Lichtgeschwindigkeit * Anteil im Kabel)

$$\frac{100.000.000}{299.792.458 * 0{,}6} * 2 = 1{,}112\ bits/m$$

$$\frac{512 - 294}{1{,}112} = 205m$$

Simplex: Kann Informatoinen nur in eine Richtung übertragen
Half-Duplex: Informationsübertragung in beide Richtungen, allerdings nicht gleichzeitig
Full-Duplux: Informationsübertragung in beide Richtungen, auch gleichzeitig

Voraussetzungen für Full Duplex:
- eigene Empfangs/Sendepfade
- Punkt-zu-Punkt-Verbindung .. 1:1
- **keine** Repeater / Hubs sondern **nur** Switches / Bridges

Konsequenzen:
- Multiple Access nicht mehr gegeben --> CSMA/CD ausgeschaltet
- dadurch längere Leitungen möglich

Auto-Negotiation:
Aufgrund der verschiedenen Ethernet Geschwindigkeiten von 10 MBit bis 1 GBit ist es zweckmäßig wenn z.B. ein Computer mit 10 MBit Netzwerkkarte auch in einem LAN mit 100 MBit "mitreden" darf. Für das wird Auto Negotiation benötigt, es ist nämlich zuständig dafür, dass zwischen Sender und Empfänger der richtige Übertragungsmodus ausgewählt wird. Vereinbart werden dabei Geschwindigkeit (10 / 100 / 1000 MBit) und ob Full/Half-Duplex.
Der Vorteil von AutoNegotiation ist, dass die Verbindung automatisch so konfiguriert wird, dass man die maximale Performance erhält, außerdem erhält man wie schon erwähnt die Kompatibilität zwischen den verschieden schnellen Ethernet Varianten. Das zwischen den Stationen liegende Kabel wird dabei nicht überprüft! Bei Gigabit ist AutoNegotiation im Standard enthalten, bei 10/100 MBit nur optional.
Für Systeme ohne Auto-Negotiation gibt es Parallel Detection, überträgt automatisch Half-Duplex und kann selbst die Übertragungsgeschwindigkeit feststellen.

Encoding bei 100BASETX**:**

1. 4B/5B encoding
2. TP-PDM Stream Cipher
3. MLT-3 (Multi Level Transition), da mit 3 Spannungslevels gearbeitet wird: 1 / 0 / -1

verschiedene Repeatertypen

1. Class 1: Verbindet verschiedene Ethernettypen --> höhere Verzögerungszeit --> max 1 Repeater zwischen zwei DTEs (=Data Terminal Equipment)
2. Class 2: Verbindet gleiche Ethernettypen --> max 2 Repeater

5-4-3-2-1 – Regel für 10Mbit Ethernet

- max. Länge zwischen 2 DTEs
- 5 Segmente und damit
- 4 Repeater
- 3 Segmente können Knoten beinhalten, die anderen
- 2 Segmente müssen reine Link-Segmente sein (full duplex). Daraus entsteht
- 1 Collisiondomain mit max 1024 Knoten.

Bridge/Switch Latency:

- Store and Forward: ganzer Frame wird gelesen, dann erst weitergeleitet
- No Collision: Frame wird erst weitergesandt wenn sicher keine Collision aufgetreten ist (außer Late Collision). Dies ist nach >= 8 + 64 Oktets der Fall, da dann die Größe eines minimalen Ethernet-Frames erreicht ist. (8 = Präambel, 64 = 6 Byte Dest, 6 Byte Src, 2 Byte len/type, 4 Byte CRC und min. 46 Byte Daten)
- Cut Through: Frame wird sofort nachdem die Zieladresse empfangen wurde weitergeleitet.
- Adaptive: Die Switch beobachet die Anzahl der CRC-Errors und der Runts (=durch Kollisionen verstümmelte Pakete) und wechselt je nach Situation zwischen den oberen 3 Methoden. z.B. Wenn Cut Through & zu viele Runts --> No Collision. wenn dann noch zu viele CRC-Errors --> Store and Forward.

Zusammenfassung von Bridges / Switches:

- höhere Performance durch Traffic shaping und Full Duplex
- physisch größere Netze durch Trennung in Collisiondomains
- Verbinden von unterschiedlichen Netzwerktypen
- Einrichtung von VLANs
- nicht mehr als 7 Bridges pro Netz verwenden wegen der Verzögerungen

Distributed Backbone = von einem Punkt gehen für jedes Netz eine eigene Leitung weg
Collapsed Backbone = von einem Punkt geht eine Leitung weg, zum ersten Netz, weiter zum zweiten,..

LLC (Logical Link Control) und SNAP (Sub Network Access Protocol)

bietet 3 Dienstoptionen:

- Typ 1: unzuverlässiger Datagramm-Service (keine Fluss/Fehlerkontrolle)
- Typ 2: bestätigter verbindungsorientierter Service (mit Fluss/Fehlerkontrolle)
- Typ 3: bestägigter verbindungsloser Service (Bestätigung mittels ACK)

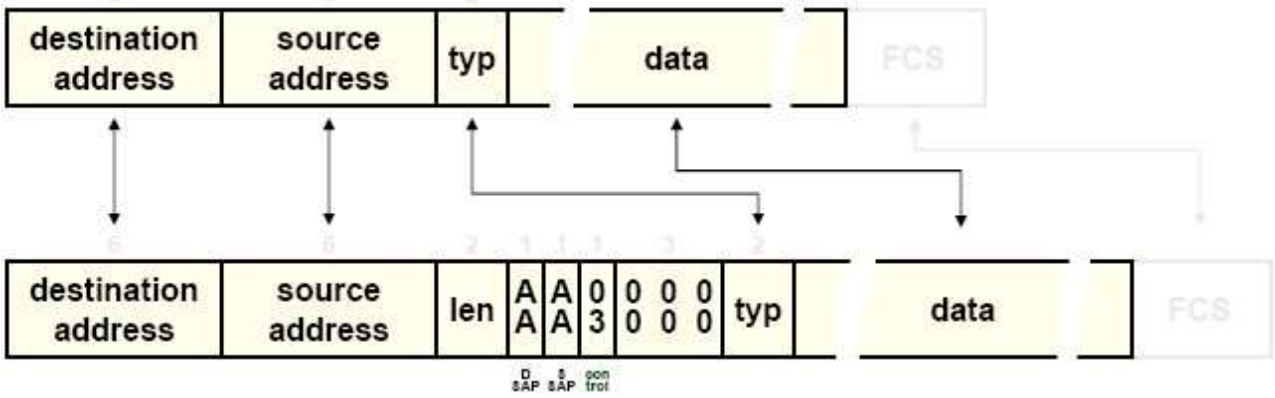
Wenn LLC und SNAP verwendet wird verändert sich auch das Aussehen des Ethernet-Frames:

Typfeld: Ist der Frame <= 1518 Bytes steht hier die Länge und der Typ in LLC und SNAP (siehe weiter unten), bei Frames größer als 1536 steht hier der Typ

Nach dem Typfeld wird ein 3-4 Bytes großes LLC-Feld eingefügt welches 1Byte Destination Service Access Point, 1 Byte Source SAP und 1-2 Bytes Control enthält. Bei der Destination ist das 1. Bit eine Flag für den Empfängertyp: individual/group. Das 1. Bit der Source enthält die Information ob die Nachricht ein Command oder eine Response ist. Bei Beiden steht im 2.Bit ein Userdefined Address.

Gleich danach werden noch 5 Bytes für SNAP eingefügt, davon die ersten 3 für organisational ID/code und die anderen 2 für type (z.B.: NetWare, IP DoD,..)

Beispiel:



Die Vermittlungsschicht (Network):

IP ist so konzipiert, dass es selbst verbindungslos ist aber darüberliegende Protokolle wie TCP einen verbindungsorientierten Dienst anbieten können.
TCP (Transport Control Protocol) und UDP (User Datagram Protocol) liegen genau über IP während ICMP (Internet Control Message Protocol) zwar als Teil von IP angesehen wird, andererseits aber auch darauf aufbaut.

Routing

Die Vermittlungsschicht ist für den Weg den ein Paket im Netzwerk nimmt zuständig, genauer ein Routing Algorithmus der im Routing Protokoll definiert wird. Beim Routen von Datagrammen wird für jedes Datagramm eine neue Route bestimmt, bei virtuellen Verbindungen wird am Anfang eine fixe Route festgelegt.

- statisches Routen: non-adaptive
- dynamisches Routen: adaptive
  - → gobales Routing: Berechnung des optimalsten Weges aufgrund Kenntnis des globalen Netzes
  - → lokales (verteiltes) Routing: jeder Knoten hat nur Infos über seine Ausgangsleitungen
    - Link State Verfahren
    - Distance Vector Verfahren

Router-Tabelle: enthält zu Beginn nur Informationen über die direkten Nachbarn, durch Austausch der Tabellen mit Nachbarroutern wächst die Kenntnis über das Netzwerk jedoch schnell an. Die automatische Anpassung von Routen hervorgerufen durch Änderungen in der Netztopologie oder Traffic wird allerdings nur beim dynamischen Routing gemacht! statisch --> admin ändert per Hand.

Beim statischen Routing betrachtet man das Netz als Graphen, hat also Kenntnis über die gesamte Topologie und kann somit Methoden der Graphentheorie anwenden. Ebenfalls möglich ist Kantenbewertung um kürzeste Wege festzustellen.

- kürzeste Wege kann man mittels Kantenbewertung (Entfernung, Durchschnittsverkehr, Bandbreite,..) und dem Dijkstra-Algorithmus ermitteln.
- beim flussbasierten Optimieren kann entweder der maximale Fluss oder die mittlere Gesamtverzögerung im Netz das Kriterium sein. Gegeben sind die Kapazitäten der einzelnen Leitung, für die m.Gesverz. braucht man auch noch den Datenverkehr pro Leitung. Mit diesen Daten kann man die mittlere Verzögerung einer Leitung berechnen und durch Ausgleichsalgorihmen die Gesamtverzögerung im Netz minimal halten.
  Der maximale Fluss in einem Netzwerk wird festgelegt durch den Schnitt durch das Netzwerk der die minimale Kapazität hat, also der Flaschenhals des Netzes.

- Flooding: jedes Paket wird an alle Leitungen geschickt außer die von der es gekommen ist. Damit keine Endlosschleifen entstehen enthalten die Pakete einen Zähler für die maximale Distanz im Netz.
- Hot Potato: Ein ankommendes Paket wird so schnell wie möglich verschickt, sobald eine Ausgangsleitung frei ist wird geschickt, auch wenn es die denkbar ungünstigste ist.

Link-State Routing:
Bei diesem Routingalgorithmus verbreitet jeder Router mittels Flooding regelmäßig Informationen bezüglich des Aufwands zum Erreichen aller seiner Nachbarn per Broad oder Multicast an alle Knoten im Netzwerk. Ein Problem dabei ist, dass es einige Zeit dauert bis jeder Router im Netzwerk vom Ausfall eines beliebigen Routers / Leitung etc erfährt.

Wenn ein Router frisch ins Netzwerk eingebunden wird muss er folgende Aufgaben erfüllen: seine Nachbarn entdecken und ihre Netzadressen feststellen (mittels Hello und Echo-Paketen), Leitungskosten messen, alles was er gelernt hat in ein Paket zusammenfassen und an alle anderen Router verschicken, den kürzesten Weg zu allen anderen Routern berechnen.
In einer Link-State-Liste steht der Name des Routers, die Folgenummer (höchste Nummer ist immer die Aktuellste), das Alter und die Namen der Nachbarsrouter inkl. der Entfernung (Übertragungsdauer / Bandbreite).
Das Alter wird bei den Routern die das Paket verwenden dekrementiert und bei 0 verworfen damit ein Paket mit durch Übertragungsfehler zu großer ID nicht ewig verwendet wird.

OSPF = Open Shortest Path First (Variante des Link-State, unterstützt Multi Access Networks)

Broadcast Routing (ein Sender soll Informationen an alle anderen Hosts versenden)

- statische Methode
  - Spanning Tree berechnen
  - Flooding
- dynamische Methode
  - Reverse Path Forwarding
    Bei diesem Verfahren wird in jedem Knoten i (wobei Knoten Router sind) geprüft, ob ein ankommendes Broadcastpaket der Quelle q auf der Leitung empfangen wurde, auf der auch der Knoten i seine Pakete an q übermittelt. Falls ja ist es offensichtlich auf dem kürzesten Weg gekommen und das Paket wird an alle anderen Leitungen weitergebroadcastet. Wenn nicht wird es verworfen, da der bisherige Weg nicht der Kürzeste war.

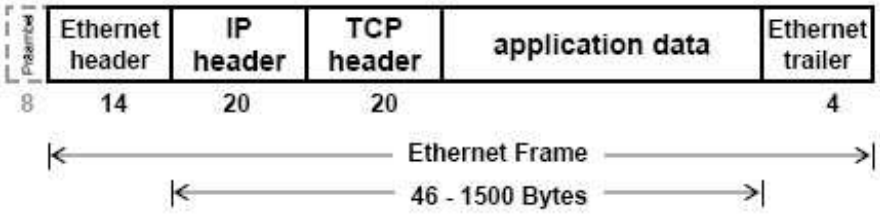
Tunneling = Kapseln von Paketen z.B. in andere Protokolle (wird z.B. bei VPN verwendet)

TCP/IP v4:

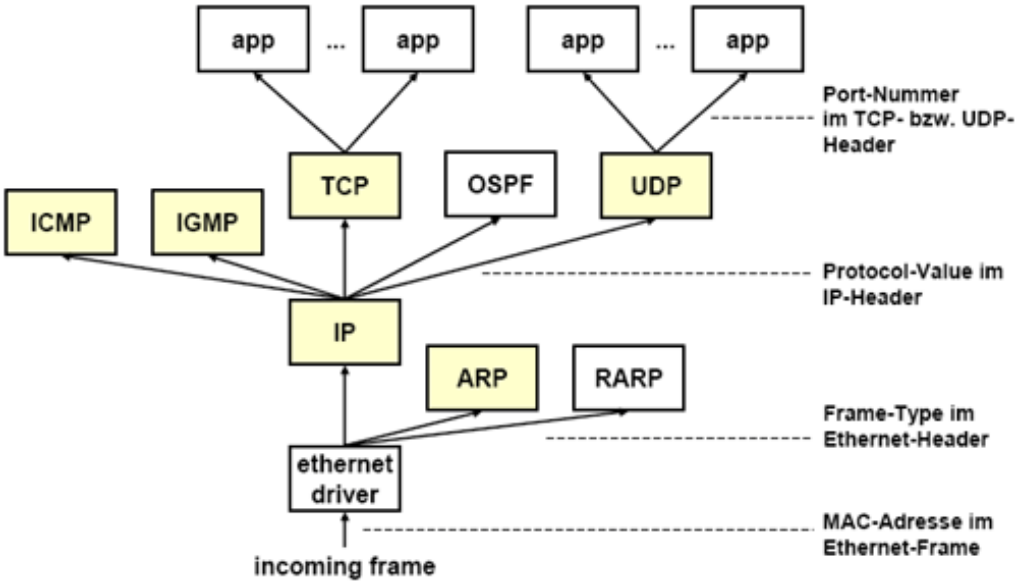
Überblick über die TCP/IP v4 Protokollfamilie

Application:	FTP, SMTP
Management:	SNMP
Routing:	RIP
Transport:	TCP, UDP
Network:	ARP, IP, ICMP, IGMP
Subnetwork:	X.25, Ethernet, Token Ring

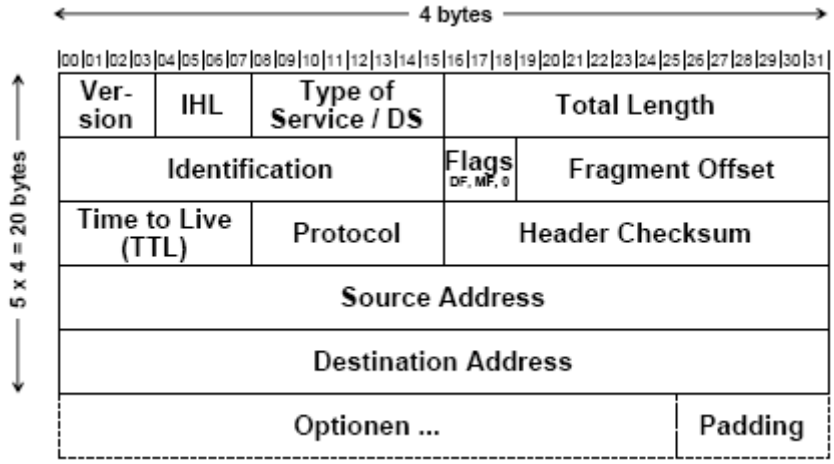
Einkapselung von Daten:



Demultiplexing von Incoming Frames:



Aufbau eines IPv4 Headers:



IHL = Internet Header Length
Type of Service ermöglicht die Vergabe von Prioritäten
Total Length ist maximal 65536Byte ~ 64KB
Identification: haben Datagrammfragmente die gleiche ID gehören sie zum selben Paket
Flags: 1.Bit: Don't Fragment, 2.Bit: More Following, 3.Bit: nicht verwendet

Fragmentierung bei IPv4 kann durch die Source und jeden Router unterwegs erfolgen, bei IPv6 fragmentiert nur die Source. Optionen haben eine 8-Bit-Kennung die bestimmt ob sie in jedes Fragment übernommen werden oder nicht.

MTU (Maximum Transfer Unit)
Sie ist die maximale Größe eines Paketes das z.B. ein Router erlaubt, wenn die MTU kleiner als das ankommende Paket und DF nicht gesetzt ist wird das Paket (eigentlich Datagramm) fragmentiert.

ARP (Address Resolution Protocol)

ist ein Netzwerkprotokoll, das die Zuordnung von Hardwareadressen zu Internetadressen ermöglicht. Obwohl es nicht auf Ethernet- und IP-Protokolle beschränkt ist, wird es fast ausschließlich im Zusammenhang mit IP-Adressierung auf Ethernet-Netzwerken verwendet. ARP gehört zur Netzwerkschicht der TCP/IP-Protokollfamilie.

Funktionsweise:
Rechner A will mit B kommunizieren, er schickt mittels Broadcast eine ARP-Anfrage für die IP von B und schick auch gleichzeitig die eigene MAC-Adresse zurück. B antwortet dann direkt an A mittels ARP-Reply und schickt natürlich auch seine MAC-Adresse mit, jetzt wissen beide bescheid.

ARP-Einträge werden in einem Cache gespeichert, nach 20 Minuten werden Einträge gelöscht, auch wenn sie benutzt werden.

Proxy-ARP: man kann z.B. manche Router so konfigurieren, dass sie auf ARP-Requests antworten, wenn der gewünschte Rechner außerhalb des eigenen physikalischen Netzes liegt.

ICMP (Internet Control Message Protocol)

ICMP ist nötig für die Erstellung einer sicheren und verbindungsorientierten Verbindung auf basis von IP, dabei arbeitet ICMP selbst auf IP.
ICMP-Typen sind unter anderem Destination Unreachable, Time Exceeded (Traceroute), Echo Request/Reply (Ping), Redirect , Router Solicitation/Advertisement,..

ICMP-Redirect kann Wege im Netzwerk automatisch verbessern:
Zum Beispiel verschickt ein Host eine Message an Gateway 1, dieses verschickt die Nachricht an Gateway 2. Wenn Gateway 2 im selben Netzwerk wie Gateway 1 liegt erkennt es das und schickt an den Host eine ICMP-Redirect Nachricht, da es eindeutig kürzer ist direkt an Gateway 2 zu schicken.

IGMP (Internet Group Message Protocol)

Das Internet Group Management Protocol basiert auf dem Internet Protocol (IP), das IP-Multicasting (Gruppenkommunikation) im Internet möglich macht. IP-Multicasting ist die Verteilung von IP-Paketen unter einer IP-Adresse an mehrere Stationen gleichzeitig. Um die Datenmenge auf das notwendigste zu reduzieren, bietet IGMP die Möglichkeit dynamisch Gruppen zu verwalten. Die Verwaltung findet nicht in der Sende-Station statt, sondern in den Routern auf dem Weg zum Empfänger. Dazu merkt sich der Router, an welcher ausgehenden Schnittstelle sich eine Station befindet, die bestimmte Multicast-IP-Pakete erhalten wollen. IGMP bietet Funktionen, mit denen sich Router untereinander verständigen und über die eine Station einem Router mitteilt, dass sie Multicast-IP-Pakete empfangen will. Der Sender von Multicast-IP-Paketen weiß dabei nicht, welche und wieviele Stationen seine Pakete empfangen. Denn er verschickt nur ein einziges Datenpaket an seinen übergeordneten Router. Der dupliziert das IP-Paket bei Bedarf, wenn er mehrere ausgehende Schnittstellen mit Empfängern hat. Damit Multicast-IP funktioniert müssen auf dem Weg zwischen Sender und Empfänger alle Netzknoten IGMP unterstützen.

Unicast: 1:1
Broadcast: 1:all
Multicast: 1:n (wird nur an eine bestimmte Zielmenge verschickt)
Anycast: nur bei IPv6, Zustellung erfolgt an eine (meistens die näheste) Schnittstelle

Multicast != Proxy, auch wenn das Ziel der Effizienzsteigerung gleich ist.
- verschiedene Ebenen des OSI Modells
- mit / ohne Datenzwischenspeicherung
- asynchron / synchron
- reliable / non reliable

UDP (User Datagram Protocol)

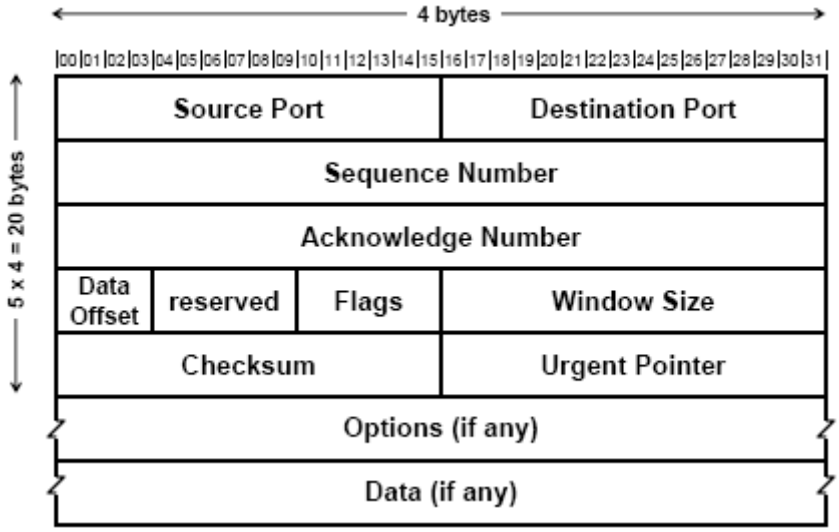
funktioniert datagrammorientiert und bietet keinerlei Verlässlichkeit. Der Header besteht nur aus Source Port Nr, Destination Port Nr, UDP Length, UDP Checksum und den Daten. Davor werden natürlich von IP noch Absender/Zieladresse, Protokoll (also UDP),.. angegeben.

TCP (Transport Control Protocol)

Überlegungen zur Verlässlichkeit:
Verlässlichkeit wird erreicht durch Bestätigung des Erhalts von Paketen mittels ACKs und dem Einsatz von CRCs. Die Pakete selbst haben Sequence-Nummern um ein fehlerfreies zusammensetzen zu garantieren, bei den ACKs wird immer die selbe Nummer verwendet. Um die Performance zu verbessern kann man auch mehrere Pakete verschicken ohne auf jedes einzelne ACK zu warten (Pipelining). Dabei kann allerdings ein Überlauf beim Empfänger auftreten. Nicht bestätigte Pakete werden nach einem Timeout einfach nochmals geschickt.
Die ACKs können auch gleichzeitig die freie Puffergröße mitschicken oder kumulative ACKs verwendet werden (ACK x bestätigt, dass alle Nachrichten von 1 bis x angekommen sind). Die kumulative Variante kann erweitert werden zu Delayed ACK, dabei wird mit dem Versenden des ACKs etwas gewartet und gleich mehr auf einmal zu erledigen.

Funktionalität von TCP:
- Verbindungsorientiert
- Zuverlässigkeit (siehe Überlegungen)
- Byte-Strom (aus Sicht der Anwendung überträgt TCP einen Bytestrom)
- Flusskontrolle (Empfänger teilt Sender seine Puffergröße mit)
- Full Duplex (eine TCP-Verbindung kann gleichzeitig empfangen und versenden)

Aufbau eines TCP Headers:



Sequence Number = Nummer des ersten Datenoktets (Bytes) im Paket
Acknowledge Number = Nummer des nächsten erwarteten Bytes (von Empfänger an Sender)
Flags:
- URG: Urgent Pointer field significant
- ACK: Acknowledgment field significant
- PSH: Push Function
- RST: Reset the connection
- SYN: Synchronize sequence numbers
- FIN: No more data from sender

Window Size: Dies ist die Größe des Clientpuffers, mehr Pakete dürfen nicht auf einmal verschickt werden, da sonst Pakete durch Pufferüberlauf verloren gehen.
Urgent Pointer: zeigt auf das erste Datenoktet nach dem Urgent-Bereich

TCP – Verbindungsaufbau:

```
     TCP A                                                TCP B

  1. CLOSED                                               LISTEN
  2. SYN-SENT    --> <SEQ=100><CTL=SYN>               --> SYN-RECEIVED
  3. ESTABLISHED <-- <SEQ=300><ACK=101><CTL=SYN,ACK>  <-- SYN-RECEIVED
  4. ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK>       --> ESTABLISHED

          Basic 3-Way Handshake for Connection Synchronization
```

TCP – Verbindungsabbau:

```
     TCP A                                                TCP B

  1. ESTABLISHED                                          ESTABLISHED
  2. (Close)
     FIN-WAIT-1  --> <SEQ=100><ACK=300><CTL=FIN,ACK>  --> CLOSE-WAIT
  3. FIN-WAIT-2  <-- <SEQ=300><ACK=101><CTL=ACK>      <-- CLOSE-WAIT
  4.                                                      (Close)
     TIME-WAIT   <-- <SEQ=300><ACK=101><CTL=FIN,ACK>  <-- LAST-ACK
  5. TIME-WAIT   --> <SEQ=101><ACK=301><CTL=ACK>      --> CLOSED
  6. (2 MSL)
     CLOSED

                          Normal Close Sequence
```

TCP-Übertragung mit Sliding-Window:
Die Window Size bestimmt wie groß das Sliding-Window ist. Beim Sender beginnt das Fenster beim 1. nicht bestätigten Paket. Er darf maximal <Fenstergröße> Pakete ausschicken bis wieder eine Bestätigung kommt und das Fenster nach rechts verschiebt.
Das Empfangsfenster dazu beginnt beim ersten Paket das von der Applikation noch nicht angenommen wurde, für dieses Paket kann bereits ein ACK verschickt worden sein. Wenn die Applikation wieder Pakete annimmt verschiebt sich das Fenster, der Client teilt dies dem sendenden Rechner mit.

Nagle-Algorithmus:
Dieser Algorithmus versucht Tinygrams (uneffiziente Minipakete) möglichst zu vermeiden indem er wenn noch Bestätigungen für bereits gesendete Pakete ausständig sind Segmente zusammenwarten lässt um sie dann auf einmal zu verschicken. Bei Echtzeitanwendungen oder z.B. einer TelnetSession ist dies allerdings nicht zu empfehlen.

Delayed ACK und Bulk Transfer:
Während die Daten der Reihe nach ohne Warten auf ACKs (aber im Rahmen des Sliding Windows) verschickt werden antwortet der Client immer nach einer bestimmten Zeit (max. 500 ms) oder einer bestimmten Anzahl von eingetroffenen Segmenten.

Problem: Silly Window Syndrome:
Der Empfänger gibt ständig kleine Windows frei die vom Sender sofort wieder aufgefüllt werden wodurch sehr viele Tinygrams verschickt werden. Eine Lösung wäre es die Window-Size nur dann zurückzugeben wenn ein bestimmter Wert an freiem Platz erreicht ist (etwa 50% des Gesamtrahmens oder ein Segment >= MSS Platz hat). Auch der Sender soll diese Bedingungen beachten und nur senden wenn sie erfüllt sind.

MSS = Maximum Segment Size

Path MTU Discovery:
Das Ziel dieser Vorgehensweise ist möglichst große Pakete zu verschicken ohne dass sie auf dem Weg fragmentiert sind. Dazu vergleichen Sender und Empfänger ihre MSS und verschicken ein Paket mit der Größe des Minimuns der beiden MSS und einem DF-Flag. Wenn keine ICMP-Fehlermeldung auftritt passt alles, ansonsten wird die MTU nach unten angepasst. Da es sein kann, dass Router ausfallen/neu dazukommen wird dieser Vorgang alle 10 Minuten wiederholt.

Jacobson Retransmission Timeout:
Wenn ein Segment verschickt wird startet ein Timer, läuft der Timer ab bevor das passende ACK angekommen ist wird das Segment nochmals verschickt. Die Schwierigkeit ist dabei die richtige Timeoutzeit zu finden (Gbit bis 9600K-Handy). Eine einfache Variante ist die RTT Roundtriptime mal mindestens 2 zu nehmen. Mit der TCP-Option Timestamp kann man die RTT genauer bestimmen und in eine komplexere Formel auch die Varianz der RTT mit einbeziehen.

Fast Retransmit:
Der Sender sollte verlorengegangen Segmente möglichst schnell wiederholen. Wenn dem Empfänger ein spezielles Paket fehlt schickt er oft ein ACK des vorigen Pakets, der Sender erkennt dies und schickt die Daten ab dem fehlenden Paket nochmals.

SACK (Selected Acknowledge):
Ein verlorenes Segment soll nicht dazu führen, dass alle danach erfolgreich versendeten Segmente ebenfalls nochmals geschickt werden müssen. Für diesen Zweck kann man die Möglichkeit einführen ACKs auch in Ranges anzugeben, z.B.: ACK of Data 1; SACK of Data 3-6.

Maximaler Durchsatz:
- Congestion Bit: (Congestion = Stau/Überfüllung)
  Wenn die Queuelänge des Routers bestimmte Zeit > 1 ist wird ein sticky 'congestion bit' im IP-header des ACK gesetzt das veranlasst, dass der Sender der Daten seine Window-Size ändert.
- Slow Start:
  Wenn ein Server zu Beginn mit voller Leistung sendet wird es Ausfälle geben, darum gibt es ein Congestion-Window das aussagt wie viele Segmente der Sender schicken darf ohne dass ein ACK zurück kommt. Dabei wird mit jedem erhaltenen ACK die Congestion-Window-Size erhöht. (Trotz des Namens relativ flott..)

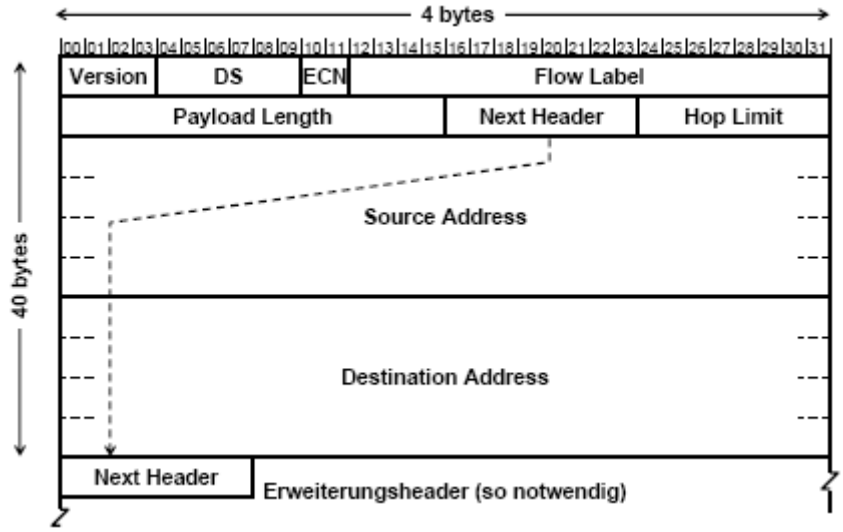
restliche Timings & Algorithmen:
- Verbindungsaufbau: Timeout & Retransmit wenn kein SYN-Paket ankommt
- Retransmission Timer: wann werden Pakete erneut gesendet wenn kein ACK kommt.
- Persist Timer: Wenn Windowsize = 0 war und das Paket mit größerer Windowsize verloren geht würde der Sender theoretisch nie mehr Senden --> Timeout mit Windowsizeabfrage
- Keepalive-Timer: (nicht Bestandteil der Spezifikation)

IPv6:

Designziele des IPv6 sind unter anderem
- unterstützen von Milliarden von Hosts, auch bei ineffizienter Adresszuteilung
- Vereinfachung der Protokolle und damit schnelleres Routen
- mehr Sicherheit und Datenschutz (z.B. IPSec)
- mehr Gewicht auf Dienstarten (Echtzeitanwendungen,..)

Allerdings ist IPv6 in keiner Weise abwärtskompatibel da es komplett unterschiedlich augebaut ist. Kompatibilität wird nur durch Tunneling erzielt, z.B.: mit 6 to 4 Nodes die IPv4 Pakete in IPv4 einpacken und über IPv4-Infrastruktur verschicken. 6 to 4 Relay Router erkennen automatisch, dass sie ein IPv6 Paket bekommen und packen es selbstständig wieder aus, wenn der Empfangsrouter dies ebenfalls kann merken Sender und Empfänger nichts vom Tunneling da es von alleine geschieht.
Eine andere Methode sind IPv4-Mapped Adressen die folgendermaßen aussehen: ::FFFF:IPv4. Ein IPv4/6 Client kann mit dieser Adresse arbeiten, beim Senden von Paketen wird IPv4 mit der eingebetteten Adresse verwendet.

IPv6-Header:



DS: Differentiated Services
ECN: Explicit Congestion Notification
Flow Label: Definiert eine Art Verbindungs-ID zwischen zwei Endpunkten. Router können anhand dieser Informationen Pakete, die zu einer Verbindung gehören, direkt übermitteln, ohne die übrigen Header-Informationen zu analysieren. Diese Funktion ist vor allem für Multimedia-Anwendungen interessant.
Source / Destination Address: haben jetzt jeweils 16 Byte zur Verfügung.

Vergleich zu IPv4:
- Payload Length nur 2 Byte, durch Erweiterungsheader aber mehr möglich
- Fragmentierung nur mehr mit Erweiterungsheader
- Hop-Count gleich geblieben.. 1 Byte ist nicht viel.
- keine Prüfsumme im Header
- Routing-Header als Erweiterungsheader

durch den Routing-Header kann z.B. Loose Routing verwirklicht werden, dabei wird nicht nur Source und Destination angegeben sondern auch bestimmte Zwischenstationen über die das Paket wander soll.

Fragmentierung:
Zum nicht-fragmentierbaren Teil gehört der IP-Header, Hop-by-Hop-Header, Routing-Header, Destination Options-Header (auf ihn folgt immer der Routing-H.);

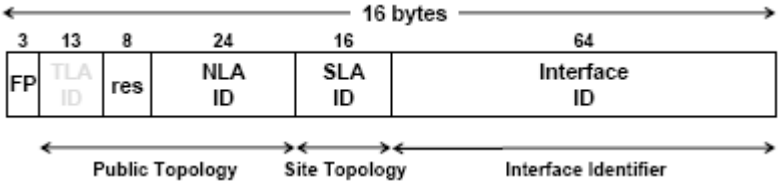
Schreibweise von IPv6-Adressen:
generell besteht die Adresse aus 8 durch einen Doppelpunkt getrennten 16-Bit Blöcken in Hexadezimalschreibweise, wobei führende Nullen in einem 16-Bit Block entfallen können und beim Auftreten von vielen Nullen in Serie kann dies mit einem :: abgekürzt werden.

Abbildung der MAC-Adresse (EUI-48) auf EUI-64 auf IPv6-Interface-ID:
Die Konvertierung der MAC-Adresse mit 48 Bit in die EUI-64-Form geschieht durch das Einfügen von FFFE (16 Bit) zwischen der OUI und der herstellerspezifischen Adresse. Für die IPv6-Interface-ID wird dann noch das 2. Bit (locally/globally) invertiert, so dass globally=1/locally=0. Damit kann man die Adressen leicht in der Kurzform schreiben: z.B.: statt 200::5 schreibt man ::5.
Die Interface-ID stellt die letzte Hälfte einer IPv6-Adresse dar.

IPv6 Link Local Unicast Address:
= FE80::/64 = FE80::<Interface-ID> .. wobei das 2.Bit der IID 1 ist! locally!!

IPv6 Site Local Unicast Address:
= FEC0::/48 = FEC0::<Subnetmask (16Bit)>:<Interface-ID>
Wird verwendet für Kommunikation innerhalb einer Site die nicht nach außen gelangen darf.

IPv6 Global Unicast Address:



FP = Format Prefix, in diesem Fall 001
TLA = Top Level Aggrevation ID
Res = reserved
NLA = Next Level Aggrevation ID
SLA = Site Level Aggrevation ID
Die Aggrevation IDs werden für Routing und Adresshierarchien verwendet.

Multicast Adressen:
= FF00::/8
Für Multicast-Adressen werden nur die letzten 32 Bit der MAC-Adresse übernommen.
Bei Solicited-Adressen (z.B. für Router Solicitation) werden nur die letzten 24 Bit übernommen.

Anycast:
Mittels Anycast kann ein Set von Routern die einer bestimmten Organisation gehören bestimmt werden, dadurch kann man z.B.: die Pakete durch einen bestimmten Serviceprovider verschicken.

ein IPv6-Node horch auf:
- Link-Lokale Unicast Adresse
- zugewiesene Unicast Adressen
- Loopback
- All-Nodes Unicast
- pro vorhandener Unicast und Anycast-Adresse: Solicited Node Multicast Address
- 'zugewiesene' Multicast Adressen

ICMPv6:

ICMPv6 fasst ARP, ICMPv4 und IGMP zusammen.

Beispiel Neighbor Discovery:
- Wenn Knoten A noch keine IP-Adresse hat schickt er eine Neighbor-Solicitation an seine gewünschte zukünftige Solicited Node Unicast Address
- Wenn die Adresse schon jemand hat kommt ein Neighbor Advertisement zurück, die Adresse muss noch auf Gleichheit geprüft werden. (gesendet an FF02::1, da A ja noch keine IP-Adresse hat!). Wenn nicht nimmt sich A die Adresse.
- Knoten A will jetzt mit B kommunizieren und stellt fest, dass er mit B im selben Netzwerksegment ist.
- A sendet eine Neighbor-Solicitation an die Solicited Node Unicast Address von B
- B antwortet direkt an A mit einem Neighbor Advertisement (IP und MAC enthalten)

Beispiel Router Solicitation & Advertisement
- Router senden periodisch ein Router Advertisement welches die nötige Information für Clients enthält um sich die richtige IP zu geben (Prefix, Netmask, Router-IP & MAC, Max. Hop Limit, Timings,..)
- wenn ein Client neu ins Netz kommt will er aber nicht so lange warten und fragt mittels Router Solicitation alle Router um ein Advertisement an, woraufhin sie antworten.

Beispiel Stateless Autoconfiguration:
- Rechner generiert aus MAC-Adressse seine IP
- fragt mittels Neighbor Solicitation nach der eigenen Adresse
- Wenn eine Antwort vom Doppelgänger kommt, andere Adresse generieren.
- Router Solicitation an alle Router schicken
- mit den Informationen selbst konfigurieren

Angabe IPv6-Adressen in HTTP:

Beispiel: http://[2002:8C4E:6482::8C4E:6482]:8080/top.htm
die eckigen Klammern müssen verwendet werden, da der Doppelpunkt schon für Ports verwendet wird.

Verteilte Systeme:

Ein Verteiltes System ist ein Zusammenschluss unabhängiger Computer, welcher sich für den Benutzer als ein einzelnes System präsentiert.

2 Aspekte:
- Hardware: die Maschinen sind autonom
- Software: für den Benutzer wirkt das System wie ein Rechner

Data Migration:
User an System A möchte Daten die auf B liegen. Jetzt kann man entweder das ganze File nach A transportieren oder nur genau die Daten die benötigt werden. Zu lösen sind Locks auf die Daten und die Möglichkeit, dass die Daten anders codiert sind als erwartet (charsets,..)

Computation Migration:
Ein effizienterer Ansatz als der erste, man überträgt die Berechnungsvorschrift hin und und lässt sich das Ergebnis schicken, das kann man entweder per Remote Procedure Call (RPC) oder message passing (A sendet Nachricht an B, neuer Prozess wird erzeugt) realisieren.

Process Migration:
ist eine Weiterführung der Computation Migration, Themen sind load balancing (Workload auf das Netzwerk aufteilen), computation speed up, hardware / software preference (manches ist nur in/auf bestimmten Rechnern verfügbar, manche Hard/Software ist für bestimmte Berechnungen besser geeignet, Lizenzen,..) und data access für Sicherheitsfragen.

Wichtig für das Design eines verteilten Systems sind:
- resource sharing: sowohl Hardware, Software als auch Daten (Groupware!) werden geteilt
- openness: öffentliche Interfaces für Schlüsselstellen in der Software
- concurrency (Gleichzeitigkeit): mehrere Aufgaben gleichzeitig bewältigen
- scalability: das System holt sich dynamisch so viele Ressourcen wie es braucht
- transparency: in Zusammenhang mit scalability, der user merkt nichts von der Anpassung
  - access: verstecke unterschiedliche Datenrepräsentationen
  - location: verstecke wo die Daten liegen
  - migrateion: verstecke Änderungen in der Datenlokalisation
  - relocation: verstecke Änderungen in der Datenlokalisation während die Daten benutzt werden
  - replication: verstecke Redundanzen
  - concurrency: verstecke, dass eine Ressource geteilt werden kann
  - failure: verstecke Fehler und Wiederherstellung
  - persistence: verstecke ob die Software/Daten im Speicher oder auf einer Platte liegen

Fehlertoleranz! Ein verteiltes System soll möglichst immer weiterarbeiten können, möglicherweise weniger effizient, aber nicht ganz stoppen. --> Hardware Redundanzen, Software Rollbacks bei Fehlern
Um diese Eigenschaften zu erreichen wird Middleware verwendet, sie setzt über dem lokalen OS auf und erst **über** der Middleware laufen verteilte Programme.
Verteilte Betriebssysteme: Multiprozessor / Multicomputer. Ziel: verstecke & manage Ressourcen.

Bei Multiprozessorsystemen kann man die Synchronisation mittels Semaphoren realisieren, oft werden Spin Locks (busy waiting) verwendet.
Multicomputersysteme haben keinen gemeinsamen Speicher, Synchronisation geschieht über message passing. Entweder Sender oder Empfänger muss einen Buffer haben, die Kommunikation muss verlässlich sein.

Network Operating Systems (NOS):
Ein NOS ist ein Betriebssystem das unter anderem printer sharing, gemeinsames Filesystem, Database Sharing, Security,.. zur Verfügung stellt. Es ist designed um Workstations, PCs und Terminals an einem LAN zu verbinden. NOS kann man noch vor der Middleware auf jedem Rechner der am verteilten System teilnimmt laufen lassen.

Zeit in verteilten Systemen:

Zeit ist ein großes Problem, denn wie schafft man es auf unterschiedlichen Rechnern eine gemeinsame Zeit zu halten, Uhren sind unterschiedlich präzis und weichen nach einiger Zeit voneinander ab.

UCT (Universal Coordinated Time) basiert auf der International Atomic Time und wird regelmäßig über Land und von Satteliten gebroadcastet. Allerdings braucht es auch wieder bis die Radiosignale beim Empfänger sind --> es ist ziemlich unmöglich eine 'common pysical time' zu realisieren.

Das ist aber auch gar nicht nötig, man kann eine künstliche Zeit einführen und diese innerhalb des Systems synchronisieren. Dies kann man mittels passiver oder aktiver Timeserver machen, wichtig ist, dass **nie** Zeit zurückgestellt wird und Änderungen langsam erfolgen (z.B.: Timer Interrupts dezent variieren)

Jeder Event wird mit einer Timestamp versehen, Events mit einer älteren (kleineren) Timestamp werden vor jüngeren bearbeitet. Zwei Events können gleichzeitig bearbeitet werden wenn keine "<" zwischen ihnen herrscht.

IPC (Inter Process Communication):

Kann entweder stream oder message orientiert sein, beim Streamen wird meistens eine Pipe (= serieller Datenstrom) geöffnet, beim Messaging werden einzelne Messages mit fixer Länge verschickt. Die Kommunikation kann auch entweder synchron oder asynchron stattfinden.
IPC-Prozesse laufen jeder in einem eigenen Speicherbereichen, shared memory kann aber für den Datenaustausch untereinander eingerichtet werden. Bei shared memory müssen allerdings wieder Synchronisationsmechanismen eingeführt werden (Semaphore, Spin-Locks,..)

IPC by Ports:
Ports sind in etwa kleine Mailboxen die exakt einen Empfänger aber mehrere Absender akzeptiert.

IPC by RPC (Remote Procedure Call)
A ruft auf B einen Prozess auf und wartet bis er das Ergebnis geliefert bekommt. Es gibt auch asynchrone RPCs bei denen der Client nach dem Prozessaufruf bei B nicht blockiert.
Grundprinzip: der Stub des Clients packt Parameter in eine Message (=Marshalling. Achtung: keine Referenzen sondern Kopien schicken. betrifft z.B. Arrays), der Kernel schickt diese zum Server, der

Server empfängt sie, der Server Stub holt die Parameter aus ihr, die Server Prozedur bearbeitet die Anfrage und das Ergebnis geht den selben Weg wieder zurück. In der Zwischenzeit ist der Client 'auf Eis gelegt'.

DCE (Distributed Computing Environment):
von den big players entwickelt um eine nahtlose Umgebung für verteilte Systeme zu schaffen, das Ziel war, dass man Rechner nimmt, dort DCE-Software laufen lässt und sofort verteilte Applikationen laufen lassen kann.

Synchronisation:

Kann zentralisiert erfolgen indem man eigene Prozesse für Semaphore oder Monitore verwendet, allerdings gibt es dann einen Single Point of Failure. Es wird immer ein Prozess als Coordinator gewählt, wenn P eine Critical Region betreten will muss er sich an den Coordinator wenden.
Um den Coordinator festzulegen kann man die Bully Method anwenden, der größte gewinnt:

- Election funktioniert folgendermaßen:
  - P schickt eine Election an alle höheren Prozesse
  - antwortet niemand ist er der Master und schickt eine Coordinatormessage an alle Niedrigeren.
  - ist ein Q größer als P (welcher die Election gestartet hat), sendet Q ein OK an P und startet selbst eine Election
  - wenn P down war und wieder online kommt startet er eine Election.

Der verteilte Ansatz setzt voraus, dass eine totale Ordnung aller Events im System herrscht, jeder Prozess weiß über die anderen bescheit soweit sie ihn betreffen und Messages die im System verschickt werden kommen sicher an. Will P eine CR betreten schickt er eine Message die den Namen der CR, die aktuelle Zeit und seine ProzessNr beinhaltet an alle Prozesse im System, inklusive sich selbst.
Will der Empfänger nicht in die CR und ist auch nicht drinnen schickt er ein OK:
Wenn der Empfänger bereits in der CR ist antwortet er **nicht** auf die Nachricht sondern reiht den Request in die entsprechende Queue ein. Nach Verlassen der Queue wird eine OK Message an alle gequeueten Prozesse geschickt.
Wenn der Empfänger in die CR will aber noch nicht drinnen ist vergleicht er die Timestamps, die ältere gewinnt, wenn der Sender die ältere hat bekommt er eine OK-Message, wenn nicht wird die Anfrage gequeued.
Wichtig: Nachdem man eine Anfrage ausgeschickt hat wartet man bis **jeder andere Prozess** mit einem OK geantwortet hat.

Eine dritte Methode ist Token Ring, das Token zirkuliert durch alle Prozesse, wer es hat darf eine CR betreten und gibt erst nachher den Token weiter.

Deadlocks:

3 notwendige Kriterien: Mutual Exclusion, Hold and Wait, No Preemption
ausreichende Kritieren: obige 3 + Circular Wait

Entweder man vermeidet Deadlocks oder man erkennt sie und löst sie auf.
Circular Wait kann man durch Reihung der Ressourcen verhindern (R1 < R2 <....), ist ein gutes globales Konzept.

Banker's algorithm:
Die Ressourcen gliedern sich in gesamte Ressourcen und verfügbare Ressourcen. Die Prozesse erhalten ebenfalls zwei Eigenschaften: Zum einen die Ressourcen, die bereits besetzt werden, zum anderen die noch benötigten Ressourcen.
Dann werden alle Prozesse - sofern möglich - nacheinander abgearbeitet und die belegten zu den verfügbaren Ressourcen zugeführt. Nach Ausführung des Algorithmus steht fest, ob ein Deadlock vermeidbar ist oder nicht. Kommt der Banker Algorithmus zu einem erfolgreichen Ende, kann unter Umständen durch "unbedachte" Ausführung der Prozesse trotzdem ein Deadlock entstehen.

Resource Preemption and Rollback: Bei Deadlock werden Ressoucen entzogen und ein Rollback durchgeführt.

Prevention using Priorities:
jeder Prozess bekommt eine Priorität. Wenn P jünger als Q ist und Ressourcen von Q braucht darf P warten bis Q fertig ist. Wenn allerdings P älter als Q ist findet ein Rollback von P statt.

Wound-Wait-Schema:
Wenn P eine Ressource braucht die von Q besetzt ist und P jünger als Q ist darf er warten, ist P älter als Q --> Rollback von Q und Entzug der Ressource von Q. (Q is wounded by P)

Wait-for-Graph:
Mittels diesem Graphen kann ein Deadlock aufgelöst werden, allerdings verbraucht es sehr viele Ressourcen und verlangt globale Kenntnis des Netzes.