Datenkommunikation und Internettechnologie WS 06/07

Lehrstuhl für Informatik 4
Dr. Dirk Thißen, Prof. Dr. Otto Spaniol

Stefan Hamacher

10. September 2007

Dies ist eine Zusammenfassung der Vorlesung, die ich zur Vorbereitung auf meine Prüfung erstellt habe. Sie beruht auf den Vorlesungsvideos vom Lehrstuhl, die ebenso wie sämtliche Folien und Übungen auf der Webseite[1] *verfügbar sind. Die Bilder sind mit freundlicher Genehmigung aus den Folien entnommen. Selbstverständlich übernehme ich keine Garantie für Richtigkeit und leider sind auch nicht alle Kapitel vollständig enthalten. Da meine Erläuterungen nicht immer ausreichend sein mögen, empfehle ich das Skript nur zur Wiederholung des Stoffes oder um sich einen groben Überblick zu verschaffen. Fehlermeldungen und/oder Kommentare sind erwünscht (`stefan.hamacher@rwth-aachen.de`).*

Inhaltsverzeichnis

Literatur

- A.S. Tanenbaum: Computer Networks. 4th Edition, Prentice Hall, 2002.
- J.F. Kurose, K.W. Ross: Computer Networking: A Top-Down Approach Featuring the Internet. Addison-Wesley, 2002.
- Cisco Systems: Internetworking Technologies Handbook. 3rd Edition, Cisco Press, 2001.

[1] `http://nets.rwth-aachen.de/content/teaching/lectures/sub/datkom/WS06-07/index.html`

1 Einführung

Datenkommunikation ist die Verarbeitung und der Transport von digitalen Daten über Verbindungen zwischen Computern und/oder anderen Geräten (normalerweise über größere Entfernungen).

1.1 Netzwerke und Netzwerktopologien

Datenkommunikation kann in zwei Bereiche unterteilt werden. In **Computernetzwerken** geht es um die Fragen:

- Wie werden mehrere Computer verbunden? (Kupferkabel, Glasfaserkabel, Funkverbindung,...)
- Wie werden die digitalen Daten transportiert? (elektrische Signale, Lichtimpulse,...)
- Wie wird der Zugriff von mehreren Computern auf das Übertragungsmedium koordiniert?

Der Bereich der **Kommunikationsprotokolle** (oder Internet Technologie/Internetprotokolle) befasst sich mit dem verlässlichen und effizienten Transfer von Dateneinheiten hauptsächlich für die Übertragung über größere Entfernungen.

Entwicklung von Datenkommunikation

Durch das Bereitstellen von Diensten auf einem Computer der mit anderen Computern verbunden ist können Kosten gespart werden. Verschiedene Institutionen können sich so teure Resourcen teilen. Für die Verbindung dieser Computer werden effiziente Mechanismen zur Datenübertragung und Interaktion benötigt. Im Laufe der Jahre wurde Computerhardware außerdem günstiger und gleichzeitig auch leistungsfähiger. Daraus entwickelte sich schließlich das **Client/Server-Prinzip** (Abb. 1).

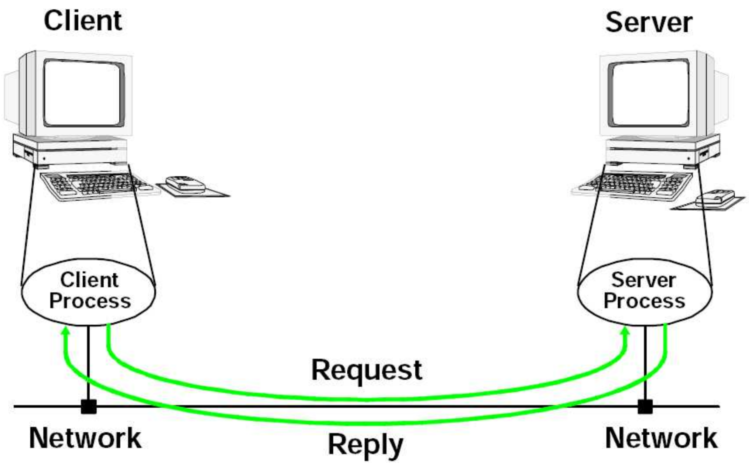


Abbildung 1: Client/Server-Prinzip

Die Vorteile des Client/Server-Prinzips sind Kostenreduktion, bessere Nutzung von Resourcen, leichtere Verwaltung durch einfaches Installieren von modularen Erweiterungen auf dem Server sowie eine höhere Verlässlichkeit durch Redundanz, da mehrere Server gleiche Dienste zur Verfügung stellen können. Das Prinzip gilt ebenfalls für viele Anwendungen wie z. B. HTTP, FTP, E-Mail-Übertragung, usw.

Ein anderes moderneres Prinzip sind **Peer-to-Peer-Systeme**, bei dem es keine festen Rollen gibt sondern jeder Host sowohl Client- als auch Serverfunktionalität übernehmen kann. Dadurch entsteht ein ganzes Netzwerk von Verbindungen. Ein Beispiel dafür ist File Sharing (Napster, Gnutella,...).
Beim schnellen und kostengünstigen Austauschen von Informationen sind auch soziale, ethnische, kulturelle, juristische, etc. Seiteneffekte zu beachten.

1.2 Kommunikationsprotokolle

Protokolle sind definiert als eine Menge von Vereinbarungen zwischen Anwendungsprozessen zur gesicherten Kommunikation. Dabei sind eine Vielzahl von Aspekten relevant:

- beide Kommunikationspartner müssen die gleichen Datenformate und Semantiken verstehen
- der Zugriff auf das Übertragungsmedium muss kontrolliert werden
- Prioritäten
- Fehlerbehandlung
- Sequenzkontrolle (Datenpakete müssen nicht zwangsweise in korrekter Reihenfolge ankommen)
- Flusskontrollmechanismen (Daten müssen in einer für beide Kommunikationspartner akzeptablen Geschwindigkeit übermittelt werden)
- Segmentierung und Erstellung langer Nachrichten
- Multiplexing (Zusammenfassen und simultanes Senden von Daten)
- Routing

Die Möglichkeit ein großes Kommunikationsprogramm zu verwenden, das alle Anforderungen des Protokolls erfüllt, bietet zwar effizienten Datenaustausch für bestimmte Anwendungen, wird allerdings trotzdem meist nicht genutzt, da sie relativ unflexibel ist. Eine implementierte Fehlerkorrektur kann bei Livestreams beispielsweise zu störenden Verzögerungen führen.
Stattdessen schreibt man heute oft mehrere kleine Programme, die für besondere Aufgaben im Verbindungsprozess benötigt und später in einer Kombination verwendet werden können. Dadurch kann z. B. der Code zur Übermittlung von Daten über ein Kupferkabel leicht ersetzt werden, falls eine optische Übertragung gewünscht wird. Nachteilig ist allerdings eine höhere Komplexität und mehr Overhead in diesen Programmen.

Das OSI-Schichtmodell

Für eine gebietsweite bis weltweite mögliche Datenübertragung sind Standardisierungen unverzichtbar. Leider sind die Prozesse bis zu einer Standardisierung häufig sehr langsam, was allerdings oft nicht-technische Gründe zur Ursache hat. Die ISO (Internationale Organisation für Normung, isos (gr.) = gleich) hat erstmalig versucht, Datenkommunikation im sog. **ISO/OSI-Schichtmodell** (OSI = Open Systems Interconnection) zu standardisieren:

1. **Datenübertragungsschicht (Physical Layer)** Auf der Datenübertragungsschicht wird festgelegt, wie Daten, die als Nullen und Einsen repräsentiert werden, über das Medium übertragen werden (elektrische Signale, optische Impulse,...)

2. **Sicherungsschicht (Data Link Layer)** Die Sicherungsschicht sorgt für eine gesicherte Übertragung der einzelnen Bits durch Gruppieren dieser in Frames und hinzufügen einer Prüfsumme. Zusätzlich wird in dieser Schicht die Flusskontrolle und der Zugriff auf das Übertragungsmedium geregelt.

3. **Vermittlungsschicht (Network Layer)** Durch Adressierung und Routing ermöglicht die Vermittlungsschicht eine Übertragung von Datenpaketen über verschiedene Stationen hinweg bis zu einem Zielrechner.

4. **Transportschicht (Transport Layer)** Damit gesichert ist, dass Daten auch vollständig und in der richtigen Reihenfolge ankommen (Fluss- und Sequenzkontrolle) wird auf der Transportschicht mit zusätzlichen Kontrollinformationen und Sequenznummern für einen netzwerkunabhängigen Datentransfer gesorgt.

5. **Sitzungsschicht (Session Layer)** Bei einem längeren Verbindungsdialog mit einem Rechner kann in der Sitzungsschicht die Verbindung durch Synchronisationspunkte wieder aufgenommen werden. Beim Download von größeren Dateien wird dadurch beispielsweise der Vorgang bei einem Abbruch fortgesetzt.

6. **Darstellungsschicht (Presentation Layer)** Auf der Darstellungsschicht wird das Datenformat für die Übertragung festgelegt.

7. **Anwendungsschicht (Application Layer)** Auf der Anwendungsschicht werden die Anwendungen definiert, die spezielle Übertragungen von Daten ermöglichen (z. B. FTP, HTTP).

Im Großen und Ganzen wird das OSI-Schichtmodell meist nur als Referenz verwendet, da viele Details oft nicht benötigt werden und eine vollständige Implementierung somit zuviel Overhead erzeugen würde.

Zusammenspiel zwischen den Schichten

Eine Schicht erweitert die zu sendenden Daten mit einem Header, der die Funktionalität der jeweiligen Schicht implementiert (z. B. eine Adresse in der Vermittlungsschicht) und sendet das Ergebnis (PDA = Protocol Data Unit) an die nächstniedrigere Schicht, die ebenfalls wieder einen Header hinzufügt (Abb. 2).

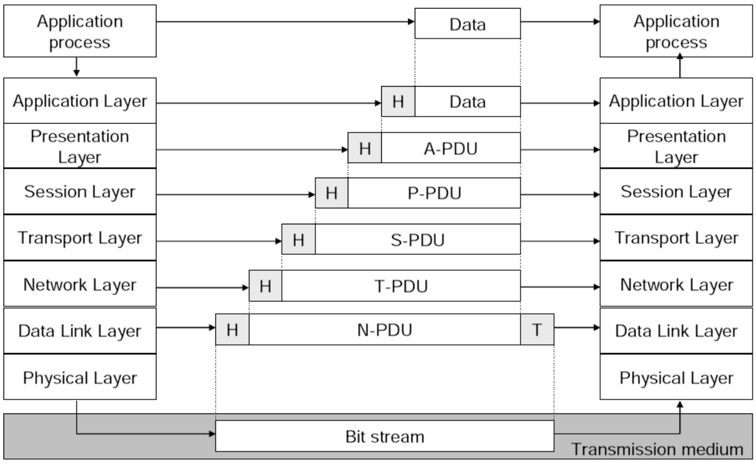


Abbildung 2: Der vollständige Kommunikationsprozess

Man beachte den Overhead, der beim Erstellen und Abarbeiten der Header entsteht. Die Pakete können außerdem so groß werden, dass sie zusätzlich gesplittet werden. In manchen Fällen, wenn eine Schicht größere Einheiten akzeptiert, werden diese auch zusammengelegt.
Abschließend sollte man feststellen, dass nur die drei untersten Schichten die tatsächliche Netzwerkübertragung und das Netzwerk selbst betreffen. Die oberen vier Schichten sind hauptsächlich anwendungsbezogen.

Computernetzwerke

Die erste Generation von Computernetzwerken bestand (z. B. in Universitäten) aus einem großen Mainframe, der von vielen Terminals aus erreichbar war und über den auch andere Peripheriegeräte angesprochen werden konnten. Um die Daten über eine größere Entfernung, z. B. zu einem anderen Gebäude, zu transportieren wurden sog. Multiplexer und Demultiplexer eingesetzt, mit denen das Bündeln mehrerer Signale möglich war. Zusätzlich war der Mainframe oft über die Telefonleitung mit dem Rest der Welt verbunden.
Da der Bedarf nach Zugriff auf zentrale Resourcen immer größer wurde, führte man später in den Gebäuden einzelne lokale Netzwerke ein, die zusätzliche lokale Resourcen (z. B. einen Drucker) beinhalten konnten und über eine Vebindung zum Mainframe verfügten. Außerdem wurden Router eingeführt, die entscheiden konnten, zu welchem Netzwerk externe Daten gesendet werden mussten.
Heute sind diese lokalen Netzwerke oft größer und bestehen aus einem eigenen Server, einem Switch und einem Router, der das ganze Netzwerk über einen Backbone mit anderen Netzwerken und einem Rechenzentrum verbindet. Ein Backbone besteht aus einer Reihe von mit hohen Bandbreiten verbundenen Routern und bezeichnet den Kernbereich eines Telekommunikationsnetzes.
Computernetzwerke können wie folgt klassifiziert werden:

- In einem **Point-to-Point Netzwerk** sind Rechner direkt miteinander verbunden.
- In einem **Broadcast Netzwerk** teilen sich alle verbundenen Rechner einen Verbindungskanal und die Daten werden grundsätzlich an alle Teilnehmer gesendet. Damit die Daten am Ziel ankommen, werden die Datenpakete mit der Unicast-Adresse des Empfängers markiert. Alle übrigen Rechner ignorieren die Daten. Zur Übermittlung von Daten an alle Stationen auf einmal gibt es eine spezielle Broadcast-Adresse.

In der Regel werden Netzwerke jedoch durch die Entfernung, die sie abdecken, klassifiziert:

∼1m	-	Personal Area Network (PAN), z. B. Handy per Bluetooth verbinden
∼10m-1km	-	Local Area Network (LAN)
∼10km-100km	-	Metropolitan Area Network (MAN)
∼100km-1000km	-	Wide Area Network (WAN), meist Point-to-Point Verbindungen
∼10000km	-	Internet

Heute ist der Bereich der LANs und WANs weiter ausgedehnt, so dass in der Regel nicht von PANs und MANs gesprochen wird.

Standards

Das *Institute of Electrical and Electronics Engineers (IEEE)* hat unter der Nummer IEEE 802.X eine Reihe von Standards für LANs (und MANs) festgelegt, z. B.:

- 802.3 CSMA/CD (Ethernet),
- 802.11 Wireless LAN,
- 802.15 Personal Area Networks (Bluetooth),
- 802.16 WirelessMAN,
- 802.17 Resilient Packet Ring,
- 802.20 Mobile Broadband Wireless Access (MBWA)

TCP/IP-Referenzmodell

Das OSI-Schichtmodell wurde nie wirklich vollständig implementiert. Stattdessen verfolgte die *Internet Engineering Task Force (IETF)* eine andere Vorgehensweise: Standardentwürfe können nur endgültige Standards werden, wenn die Implementierung erfolgreich an zwei unabhängigen Orten für wenigstens vier Monate getestet worden ist. Daraus entwickelte sich das **TCP/IP-Referenzmodell**. TCP/IP-Protokolle wurden schon benutzt bevor OSI ihre Standardisierung abgeschlossen hatte.
Das TCP/IP-Referenzmodell ist im Unterschied zum ISO/OSI-Referenzmodell wesentlich einfacher aufgebaut. Zunächst werden die beiden untersten Schichten (Data Link und Physical Layer) in einem Host-to-Network-Layer zusammengefasst, jedoch nicht tiefergehend spezifiziert. Es wird angenommen, dass die Daten irgendwie über ein Medium übertragen werden. Anstelle des Network Layers gibt es dann einen ähnlichen Internet Layer, der mit Hilfe von IP-Adressen über das IP-Protokoll das Übertragen von Datenpaketen ermöglicht. Ein erneut ähnlicher Transport Layer sorgt für das Ankommen der Pakete in korrekter Reihenfolge. Dabei wird zwischen TCP für den verlässlichen Transfer mit Kontrollinformationen und UDP für schnellen Transfer ohne Kontrollinformationen (z. B. für Audiostreaming) unterschieden. Einen Session- sowie Presentation Layer gibt es im TCP/IP-Referenzmodell nicht. Sollte derartige Funktionalität benötigt werden, so wird sie schließlich im wieder vorhandenen Application Layer (z. B. HTTP, SMTP, FTP) zusammengefasst. Durch das Vorhandensein von nur vier Schichten entfällt nun Overhead in Form von Headerinformationen und die Struktur ist wesentlich einfacher (Abb. 3).

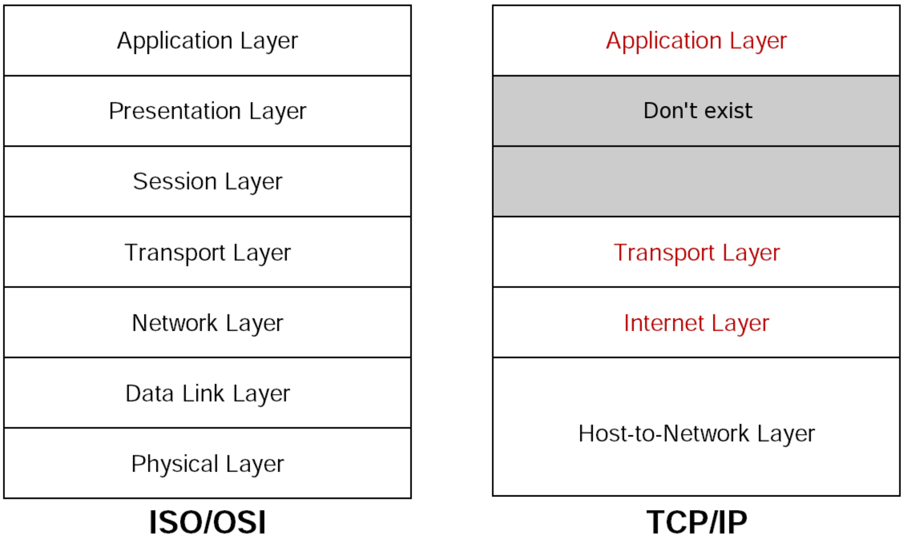


Abbildung 3: Das ISO/OSI- und TCP/IP-Referenzmodell

2 Computernetzwerke

Wie bereits gesagt dienen Local Area Networks als Kommunikationsinfrastruktur für einen begrenzten Bereich von ∼10m bis einige Kilometer. Die Übertragungskapazität ist auf max. 1000MBit/s festgesetzt (10GB/s in Vorbereitung) und die Übertragungsverzögerung (Dauer der Übertragung von Daten über das Medium) liegt im Bereich von ∼10ms. Als Topologien bieten sich an:

- Bus
- Stern
- Ring
- Baum
- Maschennetzwerk

Ähnlich dazu sind Metropolitan Area Networks aufgebaut. Bei DQDB arbeitet man beispielsweise ebenfalls mit einem Bus, der aus zwei Kabeln zum Senden und Empfangen besteht. Der Unterschied besteht in der Verwendung von **Time Slots**. Hier laufen in bestimmten Intervallen besondere Signale den Bus entlang, die festlegen wann eine Station senden darf.
Bei Wide Area Networks schließlich gibt es keine spezielle Topologie, sondern nur Routerstationen, die mit hoher Bandbreite untereinander verbunden sind und die anderen LANs und MANs mit Point-to-Point-Verknüpfungen verbinden.

2.1 Datenübertragungsschicht 1 (Physical Layer): Darstellung von digitalen Signalen

In der Datenübertragungsschicht werden alle untersten Verbindungsparameter wie mechanische (wieviele Pins?), elektrische und elektronische (welche Spannung?) sowie funktionale und prozedurale Parameter (wann kann eine Station senden? wie lange muss eine Station bis zum Senden des nächsten Bits warten?) behandelt. Es wird festgelegt, welches Übertragungsmedium genutzt werden soll, wie die Pins in Steckerverbindungen belegt sind und in welcher Form einzelne Bits über das Medium transportiert werden sollen. Daraus kann schließlich auch die Datenrate ermittelt werden. In den Bereich des Physical Layers fällt außerdem, ob die Bits seriell oder parallel, synchron oder asynchron oder im Simplex, Halb-Duplex oder Full-Duplex Übertragungsmodus gesendet werden.

Übertragungsmedien

Eine Vielzahl von Übertragungsmedien unterscheiden sich nicht nur in der Technologie und der Kapazität sondern auch in der Bitfehlerrate (BER = Bit Error Rate).

Twisted Pair Da bei einer elektrischen Übertragung andere elektromagnetische Signale die Übertragung stören können sind bei Twisted Pair Kabeln zwei isolierte Kupferkabel ineinander verdreht, was die Interferenz reduziert. Twisted Pair Kabel sind günstig, einfach und eignen sich sowohl für digitale als auch analoge Signalübertragungen. Die Bitfehlerrate liegt bei $\sim 10^{-5}$. Verschiedene Kategorien definieren verschiedene Kabel in unterschiedlicher Qualität: Cat.5 unterscheidet sich von Cat.3-Kabeln durch eine höhere Rate der Windungen und für eine bessere Isolierung wird Teflon anstatt einer Plastikhülle verwendet. Bei Cat.6 und Cat.7-Kabeln ist jedes Paar zusätzlich mit einer Silberfolie geschützt. Nur Kabel mit einer höheren Qualität eignen sich für Gigabit-Netzwerke. Für die günstigeren Kabel gibt es außerdem Varianten mit (STP = Shielded Twisted Pair) oder ohne (UTP = Unshielded Twisted Pair) eine zusätzliche Abschirmung.

Koaxialkabel Koaxialkabel besteht aus einem inneren Kupferleiter, der durch eine Isolation und einen weiteren umflochtenen Außenleiter umgeben ist. Der Außenleiter ist erneut durch eine Plastikhülle geschützt. Dadurch sind die übertragenen Signale besser als bei Twisted Pair Kabel abgeschirmt was eine höhere Signalqualität und eine größere Reichweite zur Folge hat. Die Bitfehlerrate liegt bei $\sim 10^{-9}$. Da Koaxialkabel nur einen Draht hat, ist der Aufbau eines Netzwerkes hier allerdings aufwändiger.

Glasfaserleiter Bei Glasfaserkabel werden die Informationen mit Lichtimpulsen übertragen. Theoretisch kann die Datenrate auch über größere Entfernungen dabei bis zu 50.000 GBit/s hoch sein. Es werden dabei Wellenlängen im Bereich von etwa 0,85μm, 1,3μm und 1,55μm benutzt, bei denen die Abschwächung des Signals in etwa konstant ist. Die Bitfehlerrate bei Glasfaserleitern beträgt $\sim 10^{-12}$ (Abb. 4).

Eine Glasfaser besteht aus einem sehr dünnen Kern (Core), einem Mantel (Cladding) und einer Hülle. Dadurch, dass der Kern, in dem die tatsächlichen Lichtimpulse übertragen werden, einen höheren Brechungsindex als der Mantel besitzt, verbleiben die Lichtsignale durch Totalreflexionen in der Faser. Es muss allerdings darauf geachtet werden, dass das Kabel nicht zu stark gebogen wird. Ein Glasfaserkabel besteht aus mehreren Fasern.

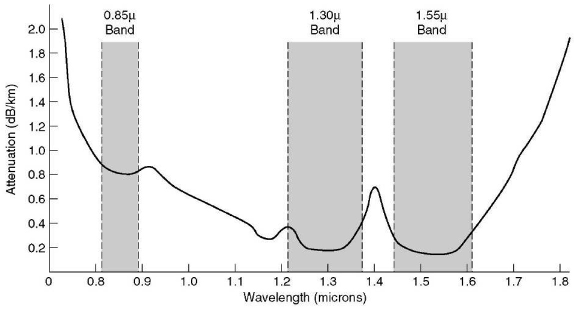


Abbildung 4: Signalabschwächung bei Glasfaserkabel

Die Nachteile von Lichtwellenleitern sind zum einen eine natürlich Absorbtion der Signale durch das Medium und andererseits eine Dispersion (Zerstreuung) des Signals, die aufgrund unterschiedlicher Ausbreitungsgeschwindigkeiten der Lichtwellen und unterschiedlicher Ausbreitung (mit mehr Brechungen) über das Medium auftritt. Dadurch können sich ursprünglich kurze elektrische Signale gegenseitig überlappen. Als Gegenmaßnahme gibt es u.a. **Monomode-Fasern**, bei der ein Signal in nur einer Wellenlänge in einem sehr dünnen Kern mit einem Durchmesser von 8-10μm übertragen wird, was die durch erhöhte Brechungen entstehende Dispersion stark vermindert.

Bei **Multimode-Fasern** beträgt der Durchmesser des Kerns etwa 50μm und es werden mehrere Lichtwellen verwendet. Die Reichweite umfasst dadurch nur etwa 2km (bei Monomode, abhängig von der Qualität des Kabels etwa 10-60km).

Eine weitere Variante sind Gradientenindexfasern, bei denen die Brechzahl in radialer Richtung nach außen hin kontinuierlich abnimmt (Gegenteil: Stufenindexfasern).

Als optische Sender können LEDs (günstig und zuverlässig aber mit einer geringen Kapazität und einem hohen Spektrum von Wellenlängen, die zu einer hohen Dispersion und einer geringen Reichweite führen) oder Laserdioden (teuer und sensibel z. B. gegen Temperaturschwankungen aber mit einer hohen Kapazität und einem geringen Spektrum von Wellenlängen) verwendet werden. Mit letzteren werden heute für eine erhöhte Kapazität mehrere Laserdioden mit unterschiedlichen Farben gleichzeitig verwendet.

Informationen codieren

Die zu übertragenen Daten werden auf mehreren Schichten codiert und übertragen. In der Datenübertragungsschicht müssen die Bits zunächst für das Medium codiert werden. Man unterscheidet dabei zwischen **Baseband-** und **Broadband-Übertragung**.

Bei Baseband werden die digitalen Daten so wie sie sind, beispielsweise als +3V für 1 und −3V für 0, übertragen.

Bei Broadband werden die Daten durch Modulation auf ein Trägersignal analog übertragen. Durch die Verwendung mehrerer Trägersignale (Frequenzen) können mehrere Informationen gleichzeitig übertragen werden.

Broadbandübertragung wird bei optischen und Funknetzwerken genutzt, sonst wird i.d.R. Baseband verwendet, da hier die Übertragung über Kupferkabel einfacher ist.

Um nun ein digitales Signal für eine Basebandübertragung zu erzeugen, wird ein Impuls aus Sinus- und Cosinusfunktionen generiert. Mehrere harmonische Schwingungen zusammen ergeben so das approximierte digitale Signal (Abb. 5).

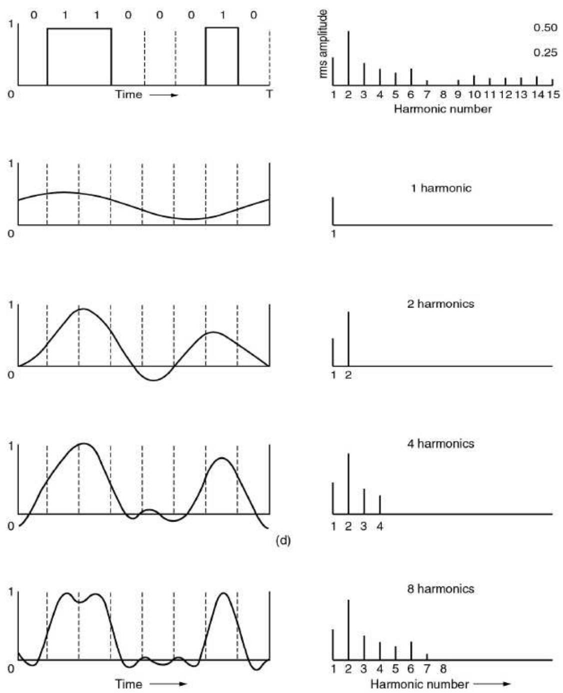


Abbildung 5: Analoge Darstellung digitaler Signale

Abgesehen von üblichen Störeinflüssen durch die Umgebung ist dabei zu beachten, dass die höheren Frequenzen über eine größere Entfernung stärker abschwächen wodurch das Signal verzerrt wird. Ein anderes Problem stellt die Synchronisation mit dem Empfänger dar, d. h. bei einer Folge von vielen identischen Bits könnte die Anzahl aufgrund einer nicht gleich laufenden Empfängeruhr falsch gezählt werden. Schließlich ändert sich bei erhöhter Übertragung von nur einem Wert auch die Spannung an den Endgeräten.

Um digitale Signale also elektrisch darzustellen sollten sie zum einen robust gegen Verzerrungen und zum anderen effizient sein. Anstelle eines Binärcodes ist es auch möglich ternären Code (z. B. +5V/0V/−5V) oder quartären Code zu verwenden, bei dem durch 4 verschiedene Zustände 2 Bits zur gleichen Zeit codiert werden. In der Regel wird allerdings Binärcode verwendet, der nun auf verschiedene Weise aufgebaut sein kann:

NRZ: Non Return to Zero Beim NRZ-Code wird 1 als positive Spannung und 0 als negative Spannung übermittelt. Das einfache Prinzip führt bei einer niedrigen Taktrate zu einer höheren Datenrate, hat aber den Verlust von Synchronisation und einer Gleichspannung bei hohen Sequenzen von 0 oder 1 zum Nachteil.

Differential NRZ Der Unterschied zu NRZ ist hier, dass 1 durch einen Spannungswechsel und 0 durch bleibende Spannung codiert wird. Dadurch können lange gleichbleibende Sequenzen nur noch durch eine Kette von Nullen entstehen.

Manchester Code Um das Synchronisations- und Spannungsproblem zu lösen, definiert der Manchester Code beide Bits als einen Spannungswechsel: Ein Wechsel vom Positiven ins Negative codiert eine 0, ein Wechsel vom Negativen ins Positive eine 1. Dadurch lassen sich außerdem leicht weitere zusätzliche Kontrollsignale (z. B. zum Starten oder Beenden einer Übertragung) in Form von einer längeren Folge positiver und/oder negativer Spannung implementieren. Der entscheidende Nachteil beim Manchester Code ist allerdings, dass hierbei die Übertragungskapazität nur zur Hälfte genutzt wird, da ein Bit letztendlich aus zwei Signalen besteht.

Differential Manchester Code Hierbei wird eine 1 so codiert, dass die Spannung zwischen diesem und dem vorherigen Bit zunächst gleich bleibt, dann aber wieder wechselt. Bei einer 0 ändert sich die Spannung abhängig von der vorherigen stattdessen sofort. Dieser Code hat allerdings die gleichen Vor- und Nachteile wie der normale Manchester Code (Abb. 6).

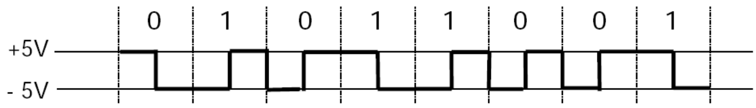


Abbildung 6: Differential Manchester Code

4B/5B Code Um nun die 50%ige Effizienz des Manchester Codes zu verbessern, verwendet man beispielsweise den 4B/5B Code, bei dem 4 Bits durch 5 Bits dargestellt werden, was die Effizient auf 80% erhöht. Als Grundlage wird Differential NRZ verwendet, mit dem in 5 Bits 32 verschiedene Sequenzen erzeugt werden können. Diesen werden nun in einer bestimmten Weise die zu codierenden 16 Zeichen der gewünschten 4 Bits zugeordnet. Zusätzlich können weitere Sequenzen für Kontrollinformationen genutzt werden. Viele andere gelten jedoch als ungültig. Durch dieses Verfahren treten maximal drei Nullen hintereinander auf. Weiterhin üblich sind 9B/10B Code und 8B/10B Code. Theoretisch kann man dies beliebig erweitern, wodurch allerdings auch der Aufwand erhöht wird.
Neben dem Codieren der Informationen werden wie schon erwähnt auch weitere Dinge für die Datenübertragungsschicht festgelegt, wie z. B. die Belegung von Pins, die Datenrate oder den Übertragungsmodus Simplex (Übertragung nur in eine Richtung), Half-Duplex (Übertragung abwechselnd in beide Richtungen) oder Full-Duplex (Übertragung gleichzeitig in beide Richtungen). Letzteres wird meistens durch zwei Kabel realisiert, möglich wäre aber auch nur ein Kabel mit zwei Übertragungskanälen oder durch Filtern der elektrischen Signale.

2.2 Sicherungsschicht 2 (Data Link Layer): Fehlerbehandlung und Zugriffskontrolle

Da durch äußere Einflüsse das in der ersten Schicht übertragene Signal trotzdem Fehler aufweisen kann, führt die Sicherungsschicht u.a. eine Fehlererkennung und -korrektur ein. Zu beachten ist, dass bestimmte zu definierende Dinge vom verwendeten Netzwerk (z. B. Token Bus, Token Ring, DQDB oder FDDI) abhängen, andere jedoch nicht. Aus diesem Grund ist Schicht 2 in 2a **Medium Access Control (MAC)** und 2b **Logical Link Control (LLC)** unterteilt. Beide Konzepte sind zusammen mit der ersten Schicht im Gerätetreiber implementiert.

Schicht 2b: Logical Link Control (LLC)

Im netzwerkunabhängigen Teil der zweiten Schicht werden die zu übermittelnden Daten in **Frames** mit einem Header für Kontrollinformationen (Adressen, Framenummern, etc.) und einer **Frame Checking Sequence (FCS)** zur Fehlererkennung und -korrektur gruppiert. Wenn eine Übertragung nicht möglich ist, muss eine Flusskontrolle die Möglichkeit schaffen, den Sender noch einmal zur Übermittlung der Daten aufzufordern sowie ggf. die Sendegeschwindigkeit zu regeln um ein Überladen des Empfängers zu vermeiden. Schließlich werden die Frames an beiden Stationen gepuffert, da erst ein vollständiger Frame anhand einer Prüfsumme verifiziert werden kann.

Fehlererkennung und -korrektur

Die einfachste Möglichkeit ist ein sog. Paritätsbit, dass bei einer ungeraden Anzahl von 1-Bits auf 1 gesetzt wird und so die Summe immer eine gerade Zahl ist. Dadurch können jedoch nur 1-Bit-Fehler erkannt werden und eine Korrektur ist nicht möglich. Außerdem können korrekt erkannte Frames trotzdem falsch sein. Durch doppelte Parität, bei der mehrere Bitblöcke mit zusätzlichen Paritätsbits geprüft werden, kann ein fehlerhaftes Bit herausgefunden werden. Mit dem Hamming-Abstand D, der die Anzahl von unterschiedlichen Bits in zwei gültigen Sequenzen angibt, kann nun angegeben werden, dass zum Korrigieren von t Fehlern, der Hamming-Abstand mind. $2 * t + 1$ betragen muss. Nur zum Erkennen von Fehlern reicht ein Hamming-Abstand von wenigstens $t + 1$. Zur Fehlerkorrektur wird also ein relativ hoher Overhead benötigt. (Bei einer bestimmten WLAN-Übertragungsmethode wird 2/3 der Daten nur für die Fehlerkorrektur gesendet.) Trotzdem kann nicht 100%ig sichergestellt werden, dass die korrigierten Daten auch die gesendeten Daten sind.

Hamming Code

Zur minimalen Fehlerkorrektur von einem Bitfehler kann der Hamming Code verwendet werden. Dabei wird das zu übertragene Wort mit zusätzlichen Paritätsbits angereichert, die an genau den Stellen stehen, deren Index eine Zweierpotenz darstellen, also an $1 = 2^0$, $2 = 2^1$, $4 = 2^2$, usw. Die Datenbits werden dann durch die Paritätsbits überprüft, deren Positionen in der Summe die Position des Datenbits ergeben, z. B. wird Position 6 durch die Paritätsbits an Positionen 2 und 4 überprüft. Die Paritätsbits werden also auf 1 gesetzt, wenn die Summe der zugehörigen Datenbits ungerade ist und beim Überprüfen werden genau die Positionen der falschen Paritätsbits addiert, was die Position des fehlerhaften Datenbits ergibt. 1-Bit-Fehler können so sicher erkannt werden, bei 2-Bit Fehlern wird entweder gar nicht oder falsch korrigiert. Mehr als 2 Fehler werden nicht erkannt.

Fehlererkennung mit zyklischen Codes

In manchen Fällen ist Fehlerkorrektur unmöglich, z. B. wenn eine ganze Reihe von Bits durch eine Störung nicht korrekt empfangen wurde. Außerdem ist es oft besser, beim Erkennen eines Fehlers den besagten Frame einfach noch einmal zu übermitteln. Die oben beschriebene Methode zur Fehlerkorrektur wird deshalb i.d.R. nur eingesetzt, wenn eine Verzögerung bei der Übertragung nicht erwünscht ist. Zur einfacheren Fehlererkennung können sog. zyklische Codes eingesetzt werden, bei denen ein k-Bit PDU $(a_{k-1}, ..., a_0)$ als Polynom $a_{k-1}x^{k-1} + ... + a_0$ mit den Koeffizienten 0 oder 1 dargestellt wird.
Der **Cyclic Redundancy Check (CRC)** arbeitet nun wie folgt: Nachdem Sender und Empfänger sich auf ein sog. Generatorpolynom $G(x)$ geeinigt haben, fügt der Sender dem Datenblock der Länge m, der als Polynom $M(x)$ interpretiert wird, zusätzliche Bits hinzu, so dass $M'(x)$ durch $G(x)$ ohne Rest teilbar ist. Der Empfänger teilt $M'(x)$ erneut durch $G(x)$ und stellt bei einem Rest von 0 fest, dass die Daten fehlerfrei empfangen wurden. Ein einzelnes Paritätsbit kann übrigens als CRC mit Generatorpolynom $x + 1$ gesehen werden.
Wenn das Generatorpolynom r-ten Grades ist (z. B. $x^4 + x + 1$ ergibt 10011), werden dem Frame zur Ermittlung der Prüfsumme r Nullen angehangen und durch das Generatorpolynom dividiert. Der Rest ist dann die Prüfsumme und wird vom Sender durch die r Nullen ersetzt. Das Verfahren lässt sich technisch sehr leicht mit XOR realisieren. Zu beachten ist, dass mit CRC theoretisch auch ein falsch übermittelter Datenframe als korrekt identifiziert werden kann, was jedoch unwahrscheinlich ist.
Gebräuchliche Generatorpolynome sind: IBM-CRC-16 $(x^{16} + x^{15} + x^2 + 1)$, CRC-CCITT (CRC-16) $(x^{16} + x^{12} + x^5 + 1)$ und beispielsweise CRC-32 für Ethernet $(x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1)$. Für 16-Bit-Generatorpolynome können alle 1-Bit bis 3-Bit-Fehler erkannt werden sowie alle Frames mit einer ungeraden Anzahl von Bitfehlern. Zusätzlich werden ganze Fehlersequenzen mit bis zu 16 Fehlern sicher erkannt und längere Sequenzen immer noch mit einer sehr hohen Wahrscheinlichkeit (∼99,997%). Mit der zusätzlichen geringen Fehlerrate von $\sim 10^{-5}$ bei Twisted Pair Kabel ergibt das eine sehr sichere Datenübertragung.
In Ausnahmefällen ist es möglich, 1-Bit-Fehler auch mit CRC zu korrigieren. So gibt es bei ATM (Asynchronous Transfer Mode) einen 5-Byte-Header, bei dem ein Byte für eine Prüfsumme gedacht ist. Die geringe Länge von also insgesamt 40 Bit führt zu nur 40 verschiedenen Nicht-Null-Resten bei 1-Bit-Fehlern, die man dann korrigieren könnte.
Man unterscheidet zwischen zwei Methoden zur Fehlersicherung:

Fehlerkorrektur: Forward Error Correction (FEC) Hierbei wird eine Fehlerkorrektur verwendet, die kleinere Fehler sofort korrigiert. Unkorrigierbare Daten werden jedoch einfach gelöscht und ignoriert. Dieses Verfahren wird verwendet, wenn keine Verzögerungen erwünscht sind sowie auch auf CDs und DVDs.

Fehlererkennung: Automatic Repeat Request (ARQ) Wenn bei diesem Verfahren Fehler mit CRC erkannt wurden, werden die entsprechenden Daten erneut angefordert. Dazu ist eine Flusskontrolle notwendig, bei der die Datenblöcke mit Nummern versehen werden.

Flusskontrolle

Send and Wait Bei ‚Send and Wait' wartet der Sender auf ein ACK des Empfängers, bevor er das nächste Datenpaket sendet. Wird nach einem Timeout keine Bestätigung empfangen, wird das Paket erneut gesendet. Nachteilig hierbei sind lange Wartezeiten zwischen dem Versenden von einzelnen Datenblöcken. Die Daten können weiterhin nicht kontinuierlich weiter versendet werden, da der Empfänger überladen werden könnte.

Sliding Window Aus den zuvor genannten Gründen einigen sich Sender und Empfänger auf ein Übertragungsfenster W, dass von der Kapazität des Übertragungsmediums und der Größe des Pufferspeichers abhängt. Dann werden die Frames sequentiell durchnummeriert, wobei $MODULUS > W$ verwendet wird, d. h. nach $MODULUS$ Frames beginnt der Zähler von vorne. Der Sender kann nun bis zu W Nachrichten verschicken, ohne ein ACK vom Empfänger erhalten zu haben. Beim Empfangen von ACK verschiebt der Sender das Fenster. Zu beachten ist, dass der $MODULUS$ echt größer als das Fenster sein muss, da bei $MODULUS = W$ bei einem ACK-Signal sonst u. U. nicht entschieden werden kann, ob das komplette Fenster entweder bestätigt oder verworfen werden soll.

Wir unterscheiden zwischen folgenden Signalen:

- ACK_j bedeutet, dass alles bis zu Datenblock j korrekt empfangen wurde.
- REJ_j / NACK_j bedeutet, dass bis zu Block $j-1$ alles korrekt, aber j fehlerhaft ist.

Go-Back-N Mit dem **Go-Back-N**-Verfahren wird der Sender nun nach einem empfangenen REJ_j zum erneuten Senden aller Daten ab Frame j aufgefordert. Das hat den Nachteil, dass dabei bereits empfangene – vermutlich korrekte – Daten unnötig neu übermittelt werden. Der Vorteil ist allerdings, dass der Empfänger nur Pufferspeicher für ein Bit haben muss.

Selective Repeat (SREPEAT) Mit der SREPEAT-Erweiterung werden empfangene Datenpakete gepuffert und bei einem fehlerhaften Paket einfach darauf gewartet, dass der Sender nach einem Timeout die Pakete nach dem letzten empfangenen ACK erneut verschickt. Wenn der Sender dann ein ACK für das letzte (gepufferte) empfangene Paket erhält, hört er auf, alte PDUs zu verschicken und fährt mit den neuen Paketen fort. Dadurch wird die Kapazität effizienter genutzt, allerdings ist ein Pufferspeicher auf Empfängerseite nötig.

Selective Reject (SREJ_j) Der Unterschied zu SREPEAT liegt bei SREJ_j darin, dass bei einem fehlerhaften oder fehlenden PDU dieses mit REJ_j verworfen wird. Der Sender verschickt daraufhin nur dieses eine Paket j wodurch die Übertragungskapazität besser genutzt wird. Als Variante kann der Empfänger auch eine Liste von fehlenden PDUs verschicken. Auch hierbei benötigt der Empfänger jedoch einen Pufferspeicher. Zu beachten ist, dass nach einem REJ_j kein ACK für andere Pakete verschickt wird, bis das zurückgewiesene Paket bestätigt wurden konnte, da sich ACK auch auf die vorhergehenden Pakete bezieht.

High-Level Data Link Control (HDLC) Das HDLC-Protokoll ist ein Beispiel für eine Implementierung der oben genannten Funktionen. Für einen korrekten Verbindungsauf- und -abbau beginnt und endet jeder Datenblock zunächst mit dem speziellen Flag 01111110. Um zu vermeiden, dass dieses Flag innerhalb der Daten auftritt, wird **bit stuffing** angewendet, d. h. nach jeweils fünf 1 wird eine 0 eingefügt, die auf der Empfängerseite wieder entfernt wird. Zwischen zwei solcher Flags steht dann zunächst die Adresse (8-Bit) und ein 8-Bit-Steuerfeld. Diesem folgt in variabler Länge das Datenfeld und eine aus 16-Bit bestehende Blockprüfung mit einer Prüfsumme (CCITT CRC-16).

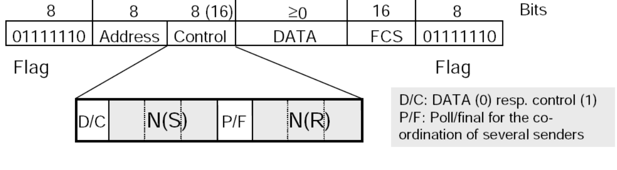


Abbildung 7: HDLC-Frame

Das erste Bit des Steuerfeldes gibt an, ob es sich um einen I-Frame (Information Frame) zur Datenübertragung (0) oder einen Steuerframe (1) handelt. Der I-Frame beinhaltet dann eine 3- oder 7-Bit lange Sende- und eine Empfangssequenznummer, die durch ein Poll/Final-Bit getrennt sind. Bei einer Duplex-Datenübertragung kann so gleichzeitig ein Datenpaket versendet, und ein empfangenes Datenpaket bestätigt werden („Piggybacking"). Handelt es sich um einen Steuerframe, bedeutet eine darauf folgende 0 ein S-Frame (Supervisory Frame) mit zwei Funktions-Bits, einem Poll/Final-Bit und drei Empfangssequenznummer-Bits zur Steuerung des Datenflusses und eine 1 ein U-Frame (Unnumbered Frame) mit drei weiteren Funktions-Bits anstelle der Empfangssequenznummer-Bits zur Steuerung der Verbindung. Die Funktions-Bits geben im ersten Fall **Receive-Ready**, **Receive-Not-Ready**, **Reject** oder **Selective-Reject** an (Abb. 7).

Das Point-to-Point Protokoll (PPP)

Heute wird für Verbindungen in Netzwerken häufig das **Point-to-Point Protokoll** verwendet, das eine Variante von HDLC für Point-to-Point Verbindungen darstellt. Es besteht ebenfalls aus einem Start- und Endflag. Das Adressfeld enthält standardmäßig den Wert 11111111 und das Steuerfeld den Wert 00000011. Danach folgen 1 oder 2 Byte mit Informationen über die verwendete Paketart in den Nutzerdaten, die in variabler Länge folgen. So kennzeichnet 00000110 z. B. das IP Protokoll für die Verarbeitung der Daten. Eine aus 2 oder 4 Byte bestehende Prüfsumme dient schließlich zur Fehlererkennung (Abb. 8).

Bytes	1	1	1	1 or 2	Variable	2 or 4	1
	Flag 01111110	Address 11111111	Control 00000011	Protocol	Payload	Checksum	Flag 01111110

Abbildung 8: PPP-Frame

Schicht 2a: Medium Access Control (MAC)

Schicht 2a wird nur für Broadcast-, nicht jedoch für Point-to-Point-Netzwerke benötigt und legt fest, wie Daten abhängig vom verwendeten Netzwerk an verschiedene Empfänger gesendet werden sollen, so dass nur ein Rechner zur gleichen Zeit Daten sendet.
Zwei einfache Möglichkeiten, mehreren Benutzern den Zugriff auf ein Übertragungsmedium zu gewähren sind TDMA und FDMA. Beim **Time Division Multiple Access**, der z. B. für Telefonnetze verwendet wird, erhält jeder Nutzer die komplette Übertragungskapazität für feste Zeitintervalle (Baseband-Übertragung), während beim **Frequency Division Multiple Access** jeder Nutzer eine bestimmte Übertragungsfrequenz erhält und die Nachrichten auf das Medium moduliert werden müssen (Broadband-Übertragung). Dies erzeugt einen hohen Overhead und wird normalerweise nur für eine Funkverbindung eingesetzt. Als Alternative gibt es deshalb sog. **reservation protocols**, bei denen in einer Reservierungsphase der Sender seinen Sendewunsch signalisiert und in einer Übertragungsphase nach erfolgreicher Reservierung die Daten versendet werden. Dabei wird die Kapazität zwar sehr effizient genutzt, derartige Verfahren führen allerdings durch das Zwei-Phasen-Schema zu Verzögerungen und häufig wird eine Zentralstation benötigt, die eine kritische Fehlerquelle darstellt. Zwei Möglichkeiten für eine einfach Reservierung ohne Zentralstation sind **explicit reservation** und **implicit reservation**.

ALOHA

Das ALOHA Protokoll wurde auf den Hawaii-Inseln entwickelt, wo mehrere Stationen nur durch Satellitenverbindungen kommunizieren konnten. Bei diesem Protokoll gibt es keine Koordination, jeder Rechner sendet seine Daten über die gleiche Frequenz und hofft, dass keine Kollision entsteht, wodurch beide Nachrichten auch schon bei minimaler Überschneidung zerstört werden können. In diesem Fall werden die Frames nach einer zufälligen Zeit einfach noch einmal gesendet. Mit ALOHA ist daher nur eine Auslastung von etwa 20% des Netzes möglich und eine Antwort, ob die Übertragung erfolgreich war, benötigt über eine Satellitenverbindung zusätzlich eine längere Zeit.
Eine Verbesserung auf 40% Auslastung brachte die Variante **Slotted ALOHA** bei der die Zeitachse in mehrere Zeitfenster unterteilt wird und jede Station die Übertragung eines Frames genau am Anfang eines Zeitfensters beginnt.
Eine andere Variante von ALOHA für Netzwerke mit geringer Reichweite wird bei Ethernet verwendet. Mit **Carrier Sense Multiple Access (CSMA)** horcht eine Station am Netzwerk und sendet, wenn keine anderen Daten versendet werden. Ansonsten wird auf eine freie Leitung gewartet. Zu beachten ist, dass dieses Prinzip nur bei einer kurzen Übertragungsverzögerung funktioniert. Außerdem besteht die Möglichkeit einer längeren Wartezeit, bevor die Übertragung beginnen kann.

Koordination mit einem Token

Eine andere Möglichkeit, den Zugriff auf das Übertragungsmedium zu regeln ist die Verwendung einer bestimmten Bitsequenz als „Token“, die von Station zu Station gereicht wird und ihnen nacheinander die Erlaubnis gibt, für eine bestimmte Zeit Daten zu senden. Diese Technik wird insbesondere in Token Ring-Netzwerken verwendet. Durch dieses Verfahren sind Kollisionen ausgeschlossen, die Netzwerkkapazität sehr gut ausgelastet und faire sowie garantierte Antwortzeiten gesichert. Es ist außerdem möglich, mehrere Token zu verwenden. Der Nachteil hierbei besteht darin, dass solche Netzwerke komplex und teuer sind, denn bei einem Verlust des Tokens z. B. durch einen Übertragungsfehler können keine Daten mehr gesendet werden.

2.3 Netzwerk Infrastruktur

Um komplexe Netzwerke aufzubauen, benötigt man einige zusätzliche Komponenten:

Repeater (Schicht 1) Ein Repeater vergrößert die Reichweite eines LANs, indem das Signal ohne Veränderung neu aufbereitet und weitergesendet wird.

Hub Ein Hub verbindet mehrere Computer eines LANs des selben Typs zu einem Broadcast-Netzwerk.

Bridge (Schicht 2) Eine Bridge verbindet LANs von verschiedenen Typen (Beispiel: WLAN Router). Sie analysiert wie ein Switch die Datenframes, kann diese für ein anderes Netzwerk umwandeln und entscheidet außerdem, ob die Daten überhaupt weitergeleitet werden müssen. Durch die Abgrenzung der Netzwerke wird deshalb auch der Traffic reduziert und es kann Datenaustausch innerhalb beider Netzwerke gleichzeitig stattfinden. Zu beachten ist, dass Bridges mehrere MAC-Adressen benötigen.
Eine **Transparent Bridge** ist für die einzelnen Stationen in den Netzwerken nicht sichtbar und enthält die Zieladressen für die Pakete in Hash-Tabellen zusammen mit dem zugehörigen Port und einer Altersangabe, so dass veraltete Einträge nach einiger Zeit (Standard: 15 Sek.) gelöscht werden können. Ist die Zieladresse unbekannt, wird der Frame in jedem Fall weitergeleitet. Zu Problemen kann es kommen, wenn zwei LANs durch zwei oder mehr Bridges verbunden sind, da dann Daten an einen nicht mehr erreichbaren Zielhost möglicherweise hin und her gesendet werden. Abhilfe schafft in diesem Fall der **Spanning Tree Algorithmus**: Zu Beginn versenden alle Bridges ein Paket mit ihrer eigenen ID (MAC-Adresse), einer Kostenangabe, einer Root-ID (initialisiert mit der eigenen ID) und dem Port. Wird ein solches Paket von einer Bridge empfangen, werden die Root-ID und die Kosten (falls vorhanden) mit kleineren Werten erneuert und es anschließend weitergeleitet. Ansonsten werden auf die Kostenangabe die eigenen Kosten aufaddiert. Zum Schluss ist die Bridge mit der niedrigsten Root-ID die Root-Bridge und die empfangenen Pakete mit den niedrigsten Kosten für die Root-Bridge geben die designierte Bridge für ein LAN und die designierten Ports an.
Im Gegensatz zu **Transparent Bridges** müssen bei **Source Routing Bridges** die einzelnen Stationen selbst wissen oder lernen, in welchem Netzwerksegment sich die Empfänger befinden, was einen größeren Aufwand bedeutet, da alle LANs und Bridges eines Pfades explizit adressiert werden müssen.
Um ein sehr großes Netzwerk aufzubauen, können allerdings trotzdem nicht ohne weiteres Bridges eingesetzt werden, die außerdem auch nur ein paar tausend Stationen unterstützen können. Zum einen ist in vielen Fällen eine bessere Trennung von LANs erwünscht (z. B. zur besseren Administration oder Fehlerbehandlung), zum anderen können Bridges auch nicht direkt mit Hosts kommunizieren (z. B. um Informationen bei Überlastung oder über zurückgewiesene Pakete zu liefern). Bridges leiten außerdem auch Broadcasts an alle teilnehmenden LANs weiter, was zu sog. „Broadcast Storms“ führen kann. Diese Mängel beseitigen **Router**.

Switch (Schicht 2) Ein Switch arbeitet wie ein Hub - jedoch ohne Broadcast - und stellt zwischen einzelnen Rechnern Point-to-Point-Verbindungen her. Er lernt dazu die MAC-Adressen der Sender und leitet die Signale an die korrekten Ports weiter. Trotzdem kann es passieren, dass zwei Rechner gleichzeitig Daten an einen anderen Rechner senden wollen und Kollisionen auftreten. Moderne Switches haben deshalb zusätzliche Pufferspeicher, in denen die Daten zwischengespeichert werden, wodurch die Gefahr von Kollisionen reduziert wird. Schließlich können durch sog. „crossbars“ alle Ausgänge mit einem Eingang und einem extra Pufferspeicher verbunden sein.
Mittlerweile gibt es auch „Layer 3-Switches“, die Funktionen der dritten Schicht übernehmen können (Routing) und „Layer 4-Switches“, die z. B. zur Lastenverteilung Informationen im TCP-Header auslesen können.

Router Ein Router verbindet mehrere LANs mit einem bestimmten Netzwerkprotokoll über größere Entfernungen.

Zum Verbinden mehrerer Rechner zu einem Netzwerk bieten sich unterschiedliche Topologien an:

Bus Ein Bus besteht aus einer durch Endwiderstände abgeschlossenen Leitung, an die alle Netzwerkteilnehmer passiv angeschlossen sind. Beim Senden von Daten erreichen diese alle Stationen, sie werden aber nur vom Zielrechner wirklich verwendet. Es handelt sich also um ein Broadcast-Netzwerk. Von Vorteil ist, dass leicht neue Rechner an den Bus angeschlossen werden können und der Ausfall eines Rechners nicht den Rest des Netzwerks stört. Der größte Nachteil eines Bus-Netzwerks ist, dass nur ein Rechner Daten gleichzeitig senden kann.

Stern Die einzelnen Rechner eines Netzwerkes können in Form eines Sterns mit einer zentralen Einheit, z. B. einem Hub oder einem Switch, untereinander verbunden werden. Diese zentrale Einheit stellt in diesem Fall jedoch eine verwundbare Stelle für das Netzwerk dar.

Baum Mehrere Bus- oder Stern-Netzwerke können zu einem Baum verbunden werden. Die verzweigenden Elemente können aktiv (Router) oder passiv (Repeater) sein. Mit einer Baum-Topologie können auf diese Weise große Entfernungen überwunden werden (Abb. 9).

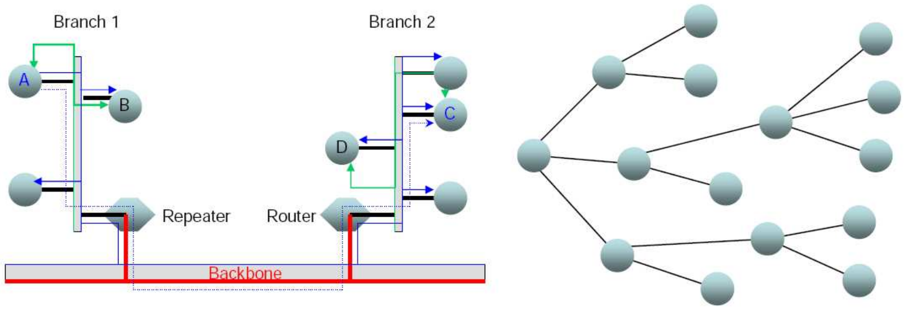


Abbildung 9: Baum-Topologie

Ring Verbindet man die Rechner des Netzwerks untereinander zu einem Ring, erhält man eine Kette von Point-to-Point-Verbindungen wobei jede Station aktiv als Repeater arbeitet und die Daten an die jeweils nächste Station weiterleitet. Prinzipiell hat man hier also auch ein Broadcast-Netzwerk. Da die Daten nur in einer Richtung versendet werden gibt es ein Problem, wenn ein Rechner ausfällt. Dies kann man durch einen bidirektinalen Ring vermeiden, bei dem für jede Richtung ein Ring vorhanden ist. Beispiele für eine Ring-Topologie sind Token Ring- und FDDI-Netzwerke.

(Vollständig) vermaschtes Netzwerk Bei einem vermaschten Netzwerk bestehen mehrere Point-to-Point-Verbindungen zwischen Stationen wodurch man viele redundante Pfade erhält, aber auch erhöhte Kosten hat.

2.4 Local Area Networks

Die bekanntesten LANs sind Ethernet (10 MBit), Fast Ethernet (100 MBit), Gigabit Ethernet und 10Gigabit Ethernet. Weniger durchgesetzt haben sich Token Bus und Token Ring-Netzwerke. FDDI (Fiber Distributed Data Interface) ist ein weiteres 100/1000 MBit-Netzwerk für LANs mit Glasfaser- oder Kupferverbindungen in einem doppelten, gegenläufigen Ring mit Token-Zugriffsmechanismus, dass für Bereiche bis 100km eingesetzt wird. Aufgrund seiner Technik zählt es ebenfalls zu Local Area Networks. WLAN-Netze haben nur eine Reichweite von bis zu ein paar 100m.
In Metropolitan Area Networks werden DQDB (Distributed Queue Dual Bus), FDDI II, das neuere von IEEE standardisierte Resilient Packet Ring und häufig auch ganz normale Gigabit Ethernet-Netzwerke verwendet.
Die in Wide Area Networks verwendeten Techniken sind Frame Relay, ATM (Asynchronous Transfer Mode) und SDH (Synchronous Digital Hierarchy).

Ethernet

Zu Beginn wurden Ethernet-Netzwerke meist in einer Bus-Topologie verwendet mit einer maximalen Segmentlänge (ohne Repeater) von 500 Metern und maximal 100 passiven Stationen. In den frühen 90er Jahren wurde die Bus-Topologie schließlich nach und nach durch die Stern-Topologie ersetzt.
Ethernet basiert auf dem Standard IEEE 802.3 „**CSMA/CD**“ (**Carrier Sense Multiple Access/Collision Detection**). Das bedeuted, wenn eine Station senden will, horcht sie zunächst auf einem Medium (Carrier Sense), ob dieses frei ist und sendet, wenn dies der Fall ist. Durch dieses einfache Verfahren ist einerseits eine gute Auslastung der Netzwerkkapazität möglich, andererseits besteht die Gefahr von längeren Wartezeiten vor dem Senden von Daten. Die Wahrscheinlichkeit einer Kollision besteht leider auch mit CSMA, wenn zwei Stationen fast gleichzeitig Daten senden. Dies wird durch folgende Kollisionserkennung (Collision Detection) erkannt: Beim Senden von Daten horcht eine Station weiterhin, ob andere Signale empfangen werden. Ist dies der Fall, wird ein Störsignal („jam“) abgeschickt, um die anderen Stationen über die fehlerhafte Nachricht zu informieren und eine zufällig gewählte Zeit gewartet, bis ein erneuter Übertragungsversuch gestartet wird. Zu beachten ist, dass in diesem Fall ein Frame eine gewisse minimale Länge haben muss, damit die Übertragung nicht abgeschlossen ist, bevor eine Kollision entdeckt werden kann. Um in diesem Zusammenhang auch Hubs oder Repeater zwischen den Stationen mit einbeziehen zu können, bezeichnet hier die **Round Trip Time (RTT)** die Zeit, die ein Datenpaket benötigt, um von einer Quelle zu einem Ziel und zurück reisen zu können. Ist die maximale Ausbreitung eines Netzwerks auf 2800m festgelegt, muss man bei einer Übertragungsrate von

10 MBit/s mindestens 64 Byte versenden, um Kollisionserkennung in dieser Weise möglich zu machen. Man kann also sagen, dass die Leistung eines Ethernet-Netzwerkes von dem Anteil eines Frames abhängt, den ein Sender übermitteln muss, bis das erste Bit das Netzwerk durchquert hat. Dabei ist es von Vorteil, wenn das Netzwerk klein, die Frames groß und die Kapazität gering ist, da durch diese Eigenschaften der angesprochene Frameanteil kleiner wird.

Datenframe für CSMA/CD

Zu Beginn einer Übertragung sendet die Station 7 Byte 10101010 zur Synchronisation (Präambel) und 1 Byte 10101011 als „start frame delimiter“. Darauf folgen jeweils 6 Byte mit der Ziel-MAC-Adresse und der Quell-MAC-Adresse. Bevor dann 0-1500 Byte Daten gesendet werden, gibt es noch 2 Byte, in denen die Länge der Daten (IEEE 802.3) oder für Ethernet eine Identifikation für das höhere Übertragungsprotokoll (z. B. „IP“) steht. Wenn die darauf folgenden Daten weniger als 64 Byte lang sind, müssen diese anschließend noch in einem „PAD-Feld“ aufgefüllt werden (0-46 Byte). Abgeschlossen werden die Daten schließlich mit einer „Frame Check Sequence“ (FCS), die eine 32-Bit-CRC-Prüfsumme für die Daten ab der Zieladresse bis zum Ende des PAD-Feldes darstellt (Abb. 10).

Preamble	SFD	DA	SA	L/T	Data	Padding	FCS

Abbildung 10: CSMA/CD-Frame

Auflösen von Kollisionen

Nachdem eine Kollision erkannt wurde, gibt es verschiedene Möglichkeiten zu reagieren: Bei einem **nonpersistent** Verhalten wird eine zufällige Zeit in einem bestimmten Zeitintervall gewartet und ein erneuter Übertragungsversuch gestartet. Nachteilig ist hier, dass das Medium ineffizient ausgelastet werden könnte. Bei einem **1-persistent** Verhalten wird der nächste Übertragungsversuch gestartet, sobald dies möglich ist. Hierbei können jedoch aufeinanderfolgende Konflikte auftreten. Der Kompromiss nennt sich ***p*-persistence** und bedeutet, dass mit einer Wahrscheinlichkeit von p ein Übertragungsversuch gestartet wird, ansonsten werden 50μs gewartet.
Der **Binary Exponential Backoff (BEB)**-Mechanismus arbeitet nun so, dass zunächst nach einer Slot-Time von 51,2μs (Ethernet-Framelänge: 512 Bits) geprüft wird, ob die Leitung frei ist und ggf. sofort ein neuer Versuch gestartet wird. Kommt es danach erneut zu einer Kollision, wird der nächste Starttermin aus 2^2 möglichen Slot-Times ausgewählt, danach aus 2^3 Möglichkeiten usw. Eine Station, die i Kollisionen hatte, wählt also einen zufälligen Wert aus dem Intervall $[0, 2^i - 1]$ aus. Bei Kollisionen 10-15 bleibt der Wert allerdings bei $[0, 2^{10} - 1]$ und nach 16 erfolglosen Versuchen mit Kollision wird die Übertragung mit einer Fehlermeldung abgebrochen.
Vorteilhaft bei BEB sind kurze Wartezeiten bei wenig Traffic und eine Verteilung der Wiederholungen bei viel Traffic. Es kann jedoch passieren, dass einzelne Stationen wiederholte Konflikte haben und sich die Wartezeit erhöht, wenn viele Stationen senden wollen.

Im folgenden nun einige Ethernet-Eigenschaften:

Parameter	**Ethernet**	**Fast Ethernet**	**Gigabit Ethernet**
Maximale Länge	bis zu 2800m	205m	200m
Kapazität	10 MBit/s	100 MBit/s	1000 MBit/s
Minimale Framelänge	64 Byte	64 Byte	520 Byte
Maximale Framelänge	1526 Byte	1526 Byte	1526 Byte
Signaldarstellung	Manchester Code	4B/5B Code, 8B/6T Code,...	8B/10B Code,...
Maximale Anzahl Repeater	5	2	1

Die Bezeichnung von Ethernet-Varianten ist wie folgt: Zunächst wird die Kapazität in MBit/s angegeben (10, 100, 1000, 10G), danach ‚Base‘ für Basebandübertragung oder ‚Broad‘ für Broadband Übertragung und schließlich ein Wert, der die maximale Länge eines Segments in 100 Metern oder das Übertragungsmedium bezeichnet.

Beispiele:

10Base-5	-	10 MBit/s, Basebandübertragung und 500m Länge für ein Segment
100Base-T2	-	100 MBit/s, Basebandübertragung über zwei Twisted Pair Kabel (zwei Leitungspaare)
1000Base-X	-	1000 MBit/s, Basebandübertragung, Glasfaserkabel

Zu beachten ist, das manche Parameter von der Variante abhängen. So beträgt die minimale Framelänge bei 1000Base-X-Netzwerken nur 416 Byte und bei 1000Base-T-Netzwerken 520 Byte.

In den Anfängen gab es 10Base-5 („Yellow Cable“) und die günstigere 10Base-2 („Cheapernet“) Ethernet-Variante mit Koaxialverbindungen. Später etablierte sich 10Base-T, bei der alle Stationen mit Twisted Pair-Kabeln in einer Sterntopologie an einen Hub angeschlossen wurden. Bei dieser Variante durfte die Länge eines Kabels zum Hub max. 100 Meter betragen. Von den 4 Kabelpaaren wurden nur 2 benutzt (zum Senden und Empfangen jeweils für Daten und Kontrollsignale).
Beim Entwickeln des schnelleren **Fast Ethernets** (100 MBit/s) sollte sichergestellt werden, dass Abwärtskompatibilität zu bereits bestehenden Ethernet-Netzwerken besteht. Es mussten also die Parameter wie z. B. die minimale Framelänge (64 Byte) beibehalten werden. Das führte allerdings dazu, dass die Kabellänge von theoretisch 2000 Meter auf 200 Meter reduziert werden musste, um Kollisionserkennung auch bei der schnelleren Übertragung von nur 64 Byte großen Frames zu ermöglichen.
Varianten von Fast Ethernet sind 100Base-T4 mit günstigem Cat.3 Twisted Pair Kabel bei dem alle 4 Kabelpaare verwendet werden und das am häufigsten verwendete 100Base-TX mit hochwertigem Cat.5 Twisted Pair Kabel bei dem nur 2 Kabelpaare verwendet werden. Bei 100Base-FX verbinden Glasfaserkabel die Stationen über Längen von bis zu 2km, weswegen hier wegen unzureichender Kollisionserkennung keine Hubs sondern nur noch Switches erlaubt sind.
Beim Sprung auf Gigabit Ethernet gibt es nun das Problem, dass zur Erhaltung der Kompatibilität mit Fast Ethernet die Kabellänge auf 20 Meter reduziert werden müsste. Aus diesem Grund wurde eine neue minimale Framelänge von 520 Byte spezifiziert. Durch die sog. **Carrier Extension** wird der Standardframe um ein „nodata“-Feld hinter der Frame Check Sequence erweitert, in dem der Frame bei Bedarf auf 520 Byte verlängert wird. Beim Transfer von einem Gigabit Ethernet in ein Fast Ethernet muss dieser Teil dann nur entfernt werden.
Eine andere Erweiterung für Gigabit Ethernet wurde eingeführt, um mehrere Frames hintereinander ohne erneutes CSMA/CD übertragen zu können. Beim **Frame Bursting** werden mehrere Frames zusammengefasst und in einem Schwung gesendet. Nach dem ersten Frame inklusive nodata-Feld folgt nach einem sog. „Interframe-Gap“ (IFG) sofort der nächste MAC-Frame. Auf diese Weise können bis zu 5,4 Frames auf einmal versendet werden.
Da aber in Gigabit Ethernet-Netzwerken in der Regel sowieso nur noch Switches eingesetzt werden und keine Kollisionen mehr auftreten können, ist CSMA/CD in gewisser Weise optional und kann deaktiviert werden.

Schließlich gibt es noch 1000Base-T (Twisted Pair Cat.5/6/7 mit 4 benutzten Kabelpaaren und einer Segmentlänge von 100m), 1000Base-CX (seltener verwendete Variante mit 2 Kabelpaaren von STP-Kabel und einer Segmentlänge von 25m). Die 1000Base-SX-Variante verwendet wieder Multimode Glasfaserkabel mit einer Länge von 550m und 1000Base-LX verwendet Mono- oder Multimode Glasfaserkabel mit einer Länge von 5000m. Mit 1000Base-LH und Monomode Glasfaserkabel sind sogar Reichweiten von bis zu 70km möglich.

Die neueste Spezifikation ist **10-Gigabit Ethernet** (IEEE 802.3ae) bei der nun vollständig auf CSMA/CD verzichtet wird, da alle Verbindungen über einen Switch laufen. Trotzdem wird aus Kompatibilitätsgründen das bekannte Frameformat beibehalten. Eine besondere Änderung besteht darin, dass es zwei Spezifikationen für den Physical Layer (PHY) gibt: Einen für LANs mit 10 GBit/s und einen für WANs mit 9,6215 GBit/s zur Kompatibilität mit SDH/SONET. Varianten von 10-Gigabit Ethernet sind:

Name	**Type**	**Wellenlänge**	**PHY**	**Codierung**	**Glasfaser**	**Reichweite**
10GBase-SR	seriell	850nm	LAN	64B/66B	Multimode	26-65m
10GBase-LR	seriell	1310nm	LAN	64B/66B	Monomode	10km
10GBase-ER	seriell	1550nm	LAN	64B/66B	Monomode	40km
10GBase-LX4	WWDM	1310nm	LAN	8B/10B	Mono-/Multimode	10km/300m
10GBase-SW	seriell	850nm	WAN	64B/66B	Multimode	26-65m
10GBase-LW	seriell	1310nm	WAN	64B/66B	Monomode	10km
10GBase-EW	seriell	1550nm	WAN	64B/66B	Monomode	40km

(**s**hort, **l**ong, **e**xtended)

Beim **(Wide) Wavelength Division Multiplexing** werden 4 Laserdioden mit 4 verschiedenen Wellenlängen/Farben benutzt, die Signale parallel über die Leitung senden. Auf der Empfängerseite wird dann ein Prisma benutzt, um die einzelnen Signale wieder zu trennen. „Wide“ bezieht sich auf einen größeren Abstand zwischen den Wellenlängen.
Eine Variante von 10-Gigabit Ethernet für etwas anderes außer Glasfaserleiter erschien bis vor ein paar Jahren noch nicht möglich. Heute gibt es jedoch Spezifikationen für Koaxialkabelverbindungen (10GBASE-CX4) über eine Länge von 15 Metern und für Twisted Pair Kabel Cat.6 (50m) und Cat.7 (100m), bei dem alle 8 Adern in beide Richtungen gleichzeitig verwendet werden. Zusätzlich sorgt eine Variante des „Pulse Amplitude Modulation“-Verfahrens (PAM) mit 16 diskreten Leveln zwischen -1 und $+1$V (PAM16) für die erhöhte Kapazität.
In der Zukunft wäre vielleicht ein Umstieg von Ethernet zur **Resilient Packet Ring**-Technologie möglich. Außerdem werden derzeit Tests mit optischen Multiplexern und Switches gemacht.

Token Bus

Die Besonderheit bei **Token Bus** oder **Token Ring** Netzwerken ist, dass jede Station gleich behandelt wird und eine garantierte Sendemöglichkeit erhält. Dazu wird ein bestimmtes Token in Form einer Bitsequenz, mit dem genau ein Frame verschickt werden kann, in der Reihenfolge der Rechneradressen weitergereicht. Verwendet wird diese Technik vorzugsweise dann, wenn vor der Datenrate vor allem die Antwortzeit ein wichtiges Kriterium darstellt. Beim Senden wird nun – vereinfacht dargestellt – entweder eine Tokennachricht mit der eigenen ID und der ID des nächsten Senders oder eine Datennachricht mit der eigenen ID und den zu übermittelnden Daten verschickt. In einem Token Bus Netzwerk hat man dadurch einen hohen Overhead, da 512 Bit alleine für die Tokennachricht verschickt werden müssen. Token Ring Netzwerke sind hier deshalb besser geeignet, da dort nur ein Bit von 0 auf 1 gesetzt werden muss.
Zwei Probleme können auftreten: Zunächst muss beim Verlassen eines logischen Rings in einem Bus die Station eine Nachricht zu seinem Vorgänger schicken und ihm den neuen Nachfolger mitteilen. Ein größeres Problem stellen jedoch neue Stationen dar, die in den logischen Ring eintreten wollen. Dazu wird in periodischen Abständen ein Fenster zwischen zwei Nachbarn geöffnet, so dass Stationen mit IDs die zwischen den IDs der beiden Nachbarn liegen eintreten können. Wenn allerdings mehrere Stationen gleichzeitig eintreten wollen, geben die letzten beiden Bits der ID an, wie lange die Station erneut ein Eintreten in den logischen Ring versuchen soll (00 kurz warten bis 11 am längsten warten). Bei gleichen Bits entscheidet das nachfolgende Bitpaar, usw. Heute sind Token Bus Netze immer mehr durch Ethernet Varianten ersetzt wie beispielsweise EtherCAT oder Ethernet Powerlink.

Token Ring

Im Gegensatz zu Token Bus kann bei **Token Ring** Netzwerken die Netzwerkkapazität sehr gut ausgelastet werden, da ein Token nur in einem Bit übertragen werden muss. Es gibt Varianten für 4, 16 und 100 MBit/s und auf unterster Ebene werden die Bits im Differential Manchester Code übertragen. Zu bemerken ist, dass alle Teilnehmer eines Token Ring Netzwerks wie Repeater die Signale aktiv empfangen und neu weiterleiten, daher kann sich ein Ring beliebig erweitern.
Daten, die nun von einer Station empfangen werden, werden in jedem Fall in den Ring weitergesendet. Sie wandern also einmal komplett von Sender zu allen Stationen, bis der Sender sie erneut empfängt und dann nicht mehr weiterleitet. Es empfängt beispielsweise eine Station ein Token (3 Byte) mit einer Übertragungserlaubnis, ändert dieses in ein belegtes Token und sendet dieses mit einem Datenframe weiter. Mehrere Frames sind möglich, solange ein Timer (Standard: 10ms) nicht überschritten wird. Anschließend wartet diese Station darauf, dass die Daten erneut ankommen, entfernt sie aus dem Ring und sendet ein freies Token weiter.
Da dieses Verfahren bei größeren Ringen zu ungenutzten Kapazitäten führen kann, gibt es die Möglichkeit eines **Multiple Token Ring**s, bei dem nach dem letzten Datenframe sofort ein neues freies Token erzeugt wird, so dass dieses von anderen Stationen zum Übertragen von Daten genutzt werden kann. In kleineren Netzen ist diese Erweiterung jedoch nicht notwendig, da die Daten schon wieder an der Sendestation ankommen, bevor das letzte Bit überhaupt gesendet wurde.

Token Ring Frame

Ein Frame beginnt mit einem Startsymbol (SD = Start Delimiter) hinter dem in einem Byte die Zugriffskontrolle (AC = Access Control) festgelegt wird (Prioritäts Bits, Token Bit, Monitor Bit und Reservierungs Bits). Nach einem Byte mit Kontrollinformationen (FC = Frame Control) folgen (wenn erforderlich) die Daten mit zugehöriger Ziel- und Quelladresse (jeweils 6 Byte). Abgeschlossen wird der Frame mit einer Frame Check Sequence (FCS) und einem Endsymbol (ED = End Delimiter) hinter dem noch ein Byte mit einem Frame Status folgt, der von dem Empfänger der Daten belegt wird: Ein Bit A (arrived) wird gesetzt, wenn die Daten angekommen sind und ein Bit C (confirmed) wird gesetzt, wenn die Daten verarbeitet wurden. Der Sender kann also sehen, ob die Station nicht vorhanden ist (00), der Frame nicht akzeptiert wurde (10) oder der Frame korrekt empfangen wurde (11). Zur Redundanz werden beide Bits doppelt gespeichert, die üebrigen 4 Bits sind 0 (Abb. 11).

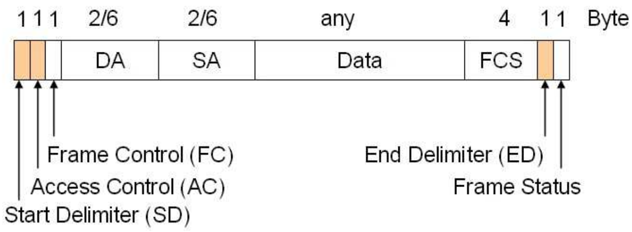


Abbildung 11: Token Ring Frame

Drei Reservierungs Bits können von einer Station mit einem Wert belegt werden, um eine erhöhte Priorität zu signalisieren. Andere Stationen können natürlich höhere Werte angeben, die dann Vorrang hätten. Bei der nächsten Tokenerstellung werden dann die drei Prioritäts Bits mit diesem Wert belegt, so dass nur die angeforderte Station dieses Token verwenden darf. Das Monitor Bit dient zur Erkennung eines erneut zirkulierenden Frames

durch eine spezielle Monitorstation im Falle einer Störung des Senders. Diese Station löscht den Frame dann und generiert ein neues freies Token. Die Monitorstation entfernt außerdem alte Framefragmente und reagiert bei einem Zusammenbrechen des Rings. Um dafür zu sorgen, dass jederzeit eine Monitorstation vorhanden ist, werden nach einer bestimmten Zeit, nachdem die Monitorstation nicht reagiert hat, sog. **CLAIM_TOKEN** ausgesendet. CLAIM_TOKEN mit einer niedrigeren ID werden gelöscht, so dass nach einem Durchlauf die Station mit der höchsten ID die neue Monitorstation wird.
Aufgrund der Komplexität von Token Ring- und Token Bus-Netzwerken werden diese heute kaum noch eingesetzt.

Fiber Distributed Data Interface (FDDI)

Die zu Token Ring ähnliche Variante bietet mit Glasfaserkabel Datenraten von 100 MBit/s über große Entfernungen von bis zu 200km und unterstützt bis zu 1000 Stationen mit einer Distanz von maximal 2km. Häufig wird ein FDDI-Ring als Backbone für kleinere LANs verwendet.
Der Nachfolger FDDI-II eignet sich auch zur Übertragung von isochroner Datenströme, bei denen immer eine feste Anzahl von Schritten zwischen zwei beliebigen Kennzeichnungspunkten liegt. Diese Übertragungsart eignet sich vor allem gut für Sprachübertragungen oder ISDN-Daten.
Eine Variante von FDDI ist CDDI (Copper Distributed Data Interface) mit einer Übertragungsrate von 100 MBit/s über Twisted Pair Kabel.
Bei FDDI kommen nun zwei Glasfaserringe mit entgegengesetzter Übertragungsrichtung zum Einsatz. Der zweite Ring ist allerdings nur ein Reservering, der bei einer Störung des anderen Rings benutzt wird. Wenn beide Ringe unterbrochen werden, können diese zu einem Ring kombiniert werden. Bei den Stationen wird unterschieden zwischen einer **Dual Attachment Station (DAS)** für einen doppelten Ring und einer günstigeren **Single Attachment Station (SAS)**, die nur an einen Ring angeschlossen werden kann. Um mehrere Ringe zu verbinden, können Konzentratoren eingesetzt werden: Ein **Dual Attachment Concentrator (DAC)** dient der Verbindung von doppelläufigen Ringen der Klasse A, ein **Single Attachment Concentrator (SAC)** dient der Verbindung von einfachen Ringen (Klasse B) und ein **Null Attachment Concentrator (NAC)** funktioniert vom Prinzip her wie zwei Repeater (Abb. 12).

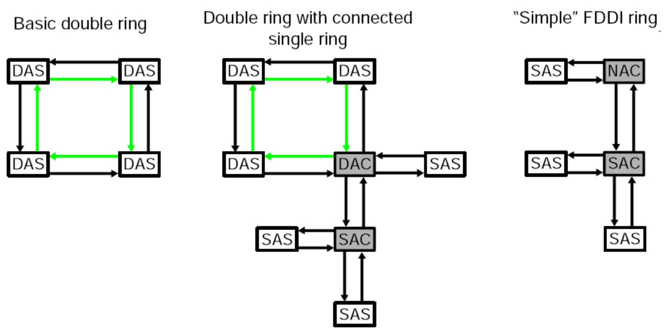


Abbildung 12: FDDI

Zur Codierung auf der ersten Schicht wird 4B/5B Code verwendet. Eine lange Präambel dient dem Empfänger zur Synchronisation des Taktgebers, der mindestens zu 0,005% stabil laufen muss. In diesem Fall können bis zu 4500 Byte Daten übertragen werden. Aufgrund der Ausbreitung von FDDI wird der **Multiple Token** Modus verwendet. Zusätzlich gibt es eine Funktion um Daten über den Reservering umzuleiten. Das Token besteht aus einer mindestens 8 Byte langen Präambel, sowie dem Startsymbol (SD) und Endsymbol (ED), zwischen denen sich **Frame Control (FC)** Bits befinden, die den Typ des Frames (Daten/Steuerung, synchron/asynchron,...) angeben. Ansonsten ist das Übertragungsprotokoll ähnlich zu IEEE 802.5 (Token Ring).
In FDDI-II können auch synchrone Frames für PCM oder ISDN-Daten übertragen werden. Dazu generiert eine Masterstation im Takt von 125μs synchrone Frames, um 8000 Samples/Sek für PCM zu erhalten. Jeder Frame besteht aus 16 Byte normalen Daten und 96 Byte für 96 PCM Kanäle pro Frame. Jede Station nutzt dann feste Slots in einem Frame und muss diese zur anderweitigen Benutzung explizit freigeben. Unbenutzte synchrone Slots werden auf Anfrage einer beliebigen Station zugewiesen.

2.5 Metropolitan Area Networks

Der Hauptunterschied zwischen einem LAN und einem MAN besteht neben der größeren Entfernung in der Verwendung eines Zeitgebers, nach dem alle Stationen synchronisiert sind. Dies soll Kollisionen vorbeugen, da bei MANs sehr viele Rechner miteinander verbunden werden sollen.

Distributed Queue Dual Bus (DQDB)

Bei DQDB sind alle Rechner an zwei gegeneinander laufende, einseitig gerichtete Busse angeschlossen, eine Station muss also eine Adresstabelle haben, um Daten in die richtige Richtung senden zu können. Für die Synchronisation gibt es dann an den Anfängen der beiden Busse jeweils eine Kopfstation, an der alle 125μs ein 53 Byte großer Frame mit einem 48 Byte leeren Datenfeld gesendet werden. In dem 5 Byte großen Header gibt es auch hier ein Byte für die Zugriffskontrolle (Access Control). Darin enthalten sind ein **Busy Bit** und ein **Request Bit**. Das Busy Bit wird von einer Station auf 1 gesetzt, wenn diese den noch freien Datenblock mit eigenen Daten füllt. Das Request Bit wird auf 1 gesetzt, wenn eine Station einen Sendewunsch auf dem anderen Bus signalisieren möchte. Zusätzlich gibt es nun für beide Senderichtungen jeweils einen **Requestcounter (RC)** und einen **Countdowncounter (CD)**, welche die Sendewünsche der anderen Stationen zählen. Ein empfangener Frame mit einem gesetzten Request Bit erhöht den Requestcounter und beim Setzen des Request Bits wird der Wert des Requestcounters in den Countdowncounter verschoben. Beim Empfangen eines freien Frames auf dem anderen Bus (mit Busy Bit auf 0) wird nun bei Sendewunsch der Countdowncounter dekrementiert bis schließlich der Frame mit den Daten belegt wird. Besteht kein Sendewunsch wird der Requestcounter dekrementiert. Auf diese Weise wird eine FIFO Warteschlange realisiert. Zu beachten ist, dass ein weiterer Request erst angefordert werden kann, wenn der aktuelle Sendewunsch ausgeführt wurde. Die Übertragungsrate von DQDB beträgt 155 MBit/s mit Glasfaserkabel und 44 MBit/s mit Koaxialkabel. Eine Reichweite von bis zu 100km sind möglich. Heute wird DQDB kaum noch eingesetzt.

Resilient Packet Ring

Diese relativ neuartige Technologie wird derzeit noch unter IEEE 802.17 standardisiert. Sie verwendet ähnlich wie FDDI einen doppelten gegenläufigen Glasfaserring, bei dem einer für den Datentransfer gedacht ist, während der andere für Kontrollinformationen genutzt wird. Außerdem gibt es eine Reihe von Funktionen zur Beseitigung von Störungen, die in einer Zeit von maximal 50ms behoben werden können.
Der Datentransfer wird im Unterschied zu FDDI nicht mit Token reguliert. Stattdessen gibt es an jedem aktiven Knoten einen Empfangspuffer, einen Sendepuffer und eine Transitqueue. An der Zielstation werden die Daten nun nicht mehr weitergeleitet sondern gelangen direkt durch den Empfangspuffer zu der Station. Über die Transitqueue werden die Daten weitergeleitet. Am Sendepuffer werden Daten solange zurückgehalten, bis die Transitqueue frei ist. Werden Daten gesendet, wird der empfangene Frame in der Transitqueue festgehalten. Gesteuert wird der Datentransfer über den zweiten Ring. Für unterschiedliche Datentypen gibt es verschiedene Klassifizierungen. Mit Traffic Shapern kann die Kapazität eingeteilt werden und es können spezielle Kontrollnachrichten verschickt werden, um andere Stationen zum Drosseln der Senderate anzufordern. Schließlich kann zur Erhöhung der Ausfallsicherheit der Datentransfer über den anderen Ring umgeleitet werden.

2.6 Wide Area Networks

Wenn sehr große Entfernungen überbrückt werden müssen und ganze Kontinente vernetzt werden sollen hat man andere Anforderungen als bei lokalen Netzwerken. Nicht der Zugriff auf ein Medium steht im Vordergrund sondern der möglichst schnelle und verlässliche Transfer von möglichst viel Daten über sehr große Entfernungen. Man spricht hier von **Wide Area Networks**. Es gibt mehrere solcher Netzwerke (z. B. das Forschungsnetz), die an bestimmten Punkten verbunden sind und gemeinsam das Internet bilden. Es gibt Point-to-Point-Verbindungen, aber keinen Broadcast.
Um nun eine solche Verbindung herzustellen gibt es drei Möglichkeiten: Eine komplette WAN Verbindung kann beispielsweise von der Telefongesellschaft gemietet werden, die dann nach Kapazität und Entfernung bezahlt wird. Eine bessere Möglichkeit bietet möglicherweise „Circuit Switching“, d. h. eine Verbindung wird nur dann hergestellt, wenn sie benötigt wird. Dies ist z. B. bei ISDN (Integrated Services Digital Network) der Fall. **Packet Switching** verbindet nun diese beiden Methoden und ein Benutzer mietet eine virtuelle Verbindung (*virtual circuit*), die mit anderen geteilt wird. Dies ist heute auch die meist genutzte Technologie in WANs. Man unterscheidet außerdem zwischen **Switched Virtual Circuits (SVC)** bei der nur sporadisch Verbindungen aufgebaut werden und **Permanent Virtual Circuits (PVC)** die für permanente Verbindungen genutzt werden.

Frame Relay

Auf Packet Switching basierend werden nun bei **Frame Relay** Datenpakete von verschiedenen Sendern mit „Statistical Multiplexing“ auf die Bandbreite verteilt, so dass am Ende statistisch gesehen alle Verbindungen gleichwertig behandelt werden. Die Bandbreite kann auf diese Weise flexibel und effizient genutzt werden. Nun können Verbindungen mit Datenraten zwischen 56 kBit/s und 45 MBit/s gemietet werden. Man unterscheidet zwischen **Data Terminal Equipment (DTE)** für Verbindungen zwischen dem LAN und dem WAN (PC, Router, Bridge, etc.) und **Data Circuit-Terminating Equipment (DCE)**, das die Übertragung innerhalb des Wide Area Networks ermöglicht.
Eine Verbindung zwischen zwei DTEs wird aufgebaut, indem jede Station für jede einzelne Verbindung einen **Data-Link Connection Identifier (DLCI)** bestimmt und die DCEs anhand dieser Bitsequenzen und den Adressen im Frameheader eine Verbindungstabelle erstellen. Nachdem dies geschehen ist, können die Pakete mit hohen Datenraten übermittelt werden. Aus diesem Grund werden auch meistens PVCs verwendet, da in diesem Fall die Verbindungstabelle weniger häufig neu erstellt werden muss.

Eine Möglichkeit zur Flusskontrolle bietet Frame Relay nicht direkt an. Im Falle der Überladung einer Leitung gibt es stattdessen zwei Benachrichtigungsmöglichkeiten, um die höheren Protokollschichten zu informieren. Bei einer **Forward-Explicit Congestion Notification (FECN)** wird in einem Frame ein FECN-Bit gesetzt, um den Empfänger über eine überladene Leitung hinzuweisen. Bei einer **Backward-Explicit Congestion Notification (BECN)** wird ein BECN-Bit in einem Frame in entgegengesetzter Richtung gesetzt, um den Sender direkt zu informieren.

Asynchronous Transfer Mode (ATM)

Aus der Idee bei Frame Relay entwickelte sich schließlich der **Asynchronous Transfer Mode (ATM)**, der versucht die zwei unterschiedlichen Transfermöglichkeiten für die Telekommunikation und die Datenkommunikation zu verbinden. In der Telekommunikation haben wir verbindungsorientierte Transfers, bei der eine Performancegarantie zum lückenlosen Übermitteln von Daten wichtig ist. Dies wird mit **Time Division Multiplexing** erreicht, wodurch allerdings auch ungenutzte Bandbreite entsteht. In der Datenkommunikation hat man hingegen verbindungslose Transfers und weist durch **Statistical Multiplexing** flexibel Ressourcen zu. Hierbei kann wiederum keine Performancegarantie gewährleistet werden.
ATM verwendet nun anstatt Frames sog. Cells und kombiniert die Vorteile von Point-to-Point-Verbindungen und Circuit Switching in Packet Switching, verwendet eine Cell-basierte Multiplexing und Switching Technologie und unterstützt PVCs, SVCs sowie verbindungslose Übertragungen. Für eine verbindungsorientierte Kommunikation werden virtuelle Verbindungen hergestellt, für die bestimmte Qualitätskriterien angefordert werden können (Bandbreite, Verzögerung, etc.). Die Datenraten betragen 34, 155 oder 622 MBit/s (Glasfaserkabel).
Das sog. „Cell Switching“ arbeitet nun mit **Asynchronous Time Multiplexing** von mehreren virtuellen Verbindungen. Ankommende Zellen werden an einer ATM Verbindung mit einem entsprechenden Header einfach hintereinander weitergeleitet und aneinandergereiht. Muss keine Zelle verschickt werden, wird stattdessen eine leere Zelle gesendet, so dass eine evtl. später ankommende Zelle erst hinter dieser angereiht wird. Bei Überlast werden entsprechende Zellen einfach gelöscht. Außerdem werden an diesen Verbindungen Zellen mit besonderen Anforderungen z. B. für Sprachübertragungen bevorzugt behandelt und entsprechend umsortiert.
Bei PCM (Pulse-Code Modulation) werden Audio-Signale wie folgt kodiert: Für eine minimal lange Schwingung müssen nach Nyquist mindestens 2 Scanwerte übermittelt werden. Bei einer Sprachübertragung, bei der man etwa 3400 Hz erreicht, hat man sich deshalb auf 8000 Hz festgelegt. Innerhalb der Quantisierungsreichweite können außerdem mit einer Genauigkeit von 8 Bit insgesamt 256 unterschiedliche y-Werte gespeichert werden. Man erhält also eine Datenrate von $8 * 8000 = 64$ kBit/s. Es wird nun nicht jede Sekunde ein PCM-Datenpaket verschickt, denn dies würde zu hohen Verzögerungen führen. Stattdessen forderten Europa und Japan nach der Möglichkeit, 32 Samples mit jeweils 8 Bit zu verschicken. Da in den USA allerdings der Wunsch nach größeren Zellen mit 64 Byte für Optimierung auf reine Datenübertragung mit weniger Overhead durch Header bestand, einigte man sich auf eine Zellengröße von 48 Byte mit einer Verzögerung von 6ms zuzüglich 5 Byte für Headerinformationen.
In einem ATM Netzwerk gibt es nun ATM Switches, an denen die Cellheader ausgewertet, angepasst und die Daten entsprechend weitergeleitet werden. Zusätzlich gibt es ATM Endpunkte, an denen mehrere unterschiedliche Netzwerke mit dem ATM Netzwerk verbunden werden. Man unterscheidet also zwischen einem „Network-Network Interface“ (NNI) und einem „User-Network-Interface“ (UNI).
Der 5-Byte-Header von ATM Zellen enthält nun 1 Byte mit einer CRC für die ersten 4 Byte, so dass 1-Bit-Fehler korrigiert werden können (HEC = Header Error Control), weiterhin 1 Bit, dass auf 1 gesetzt eine niedrigere Priorität angibt um die Zelle in Überlastsituationen zu löschen (CLP = Cell Loss Priority) und schließlich 3 Bit zur Identifizierung der Daten (PTI = Payload Type Identifier). Die übrigen 3,5 Byte werden zur Adressierung verwendet.
Für eine virtuelle Verbindung in einem ATM-Netzwerk wir nun zunächst anhand einer global eindeutigen ATM-Adresse eine Verbindung angefordert, die mit einem „OK“ bestätigt wird. Dabei wählt jeder Switch einen sog. **Connection Identifier**, der in Tabellen der Switches gespeichert wird und wohin die Daten mit welchen Qualitätsanforderungen geschickt werden müssen. Beim Weiterleiten der Zellen wird dieser Identifier erneuert. Nach dem Herstellen der Verbindung werden die ATM-Adressen nicht mehr benötigt, es werden nur noch die virtuellen Connection Identifier benutzt. In der Switchtabelle steht also der eingehende Port mit einem Header, der dazu gehörige ausgehende Port und der neue Header, da sich ja auch die Prüfsumme ändert. Nun wird zwischen **Virtual Path Identifier** und **Virtual Channel Identifier** unterschieden, die beide im Header gespeichert werden. Wenn nun mehrere virtuelle Verbindungen über eine bestimmte Strecke über den gleichen Pfad laufen, können die Switches dafür die gleichen Virtual Path Identifier benutzen, der Virtual Channel Identifier bleibt unangetastet. An den Switches, wo sich die Verbindungen trennen und unterschiedliche Ports benutzen, werden dann neben den Virtual Path Identifiers auch die Virtual Channel Identifier gesetzt. Dadurch kann die Größe der Tabellen reduziert werden.
Auch bei ATM wird zwischen verschiedenen Schichten unterschieden. Wie bekannt werden auf der **Bitübertragungsschicht (Physical Layer)** die ATM Zellen über das Medium transferiert wobei Prüfsummen zum

Verifizieren der Bits dienen. Auf der **ATM-Schicht (ATM Layer)** werden von der Sendestation Header generiert und auf Empfängerseite die Inhalte extrahiert (ausgenommen der Prüfsumme). Diese Schicht ist auch für die Connection Identifier zuständig. In der **ATM-Anpassungsschicht (ATM Adaptation Layer = AAL)** werden nun außerdem bestimmte Anforderungen der in den Anwendungen liegenden höheren Schichten festgelegt und größere Nachrichten segmentiert bzw. zusammengesetzt. Für diese Anforderungen gibt es vier verschiedene Klassen:

AAL 1: Constant Bit Rate (CBR) Anfordern einer Spitzenrate (PCR = Peak Cell Rate), die dann garantiert wird. Beispiel: PCM-Daten

AAL 2: Variable Bit Rate (VBR) Anfordern einer durchschnittlichen Zellenrate, die für eine bestimmte Zeit um einen bestimmten Betrag überschritten werden kann. Beispiel: Komprimierte Daten. Parameter: Peak Cell Rate (PCR), Sustainable Cell Rate (SCR), Maximum Burst Size

AAL 3: Available Bit Rate (ABR) Die Senderate wird dynamisch anhand der freien Bandbreite geregelt. Parameter: Peak Cell Rate (PCR), Minimum Cell Rate

AAL 4: Unspecified Bit Rate (UBR) Dieser verbindungslose Modus verwendet einfach die Bandbreite, die noch frei ist. Eine Übertragungsgarantie besteht nicht. Parameter: Peak Cell Rate (PCR)

Dieses komplexe Netzwerk benötigt außerdem weitere Kontrollmechanismen:

Connection Admission Control (CAC) Beim Herstellen der Verbindung werden die angeforderten Ressourcen überprüft und anschließend reserviert bzw. zurückgewiesen.

Usage Parameter Control/Network Parameter Control Die angeforderte Übertragungsrate wird überprüft, und eingehalten (Generic Cell Rate Algorithm/Leaky Bucket Algorithm)

Switch Congestion Control Um ständig die reservierten Anforderungen erfüllen zu können, werden selektiv bestimmte Zellen (hauptsächlich in AAL 4, UBR) bei einer Überlast gelöscht.

Flow Control In AAL 3 (ABR) gibt es Mechanismen zur Flusskontrolle, um verfügbare Bandbreite angemessen anzupassen.

Zusammenfassend kann man feststellen, dass die Besonderheit von ATM neben dem TCP-ähnlichen Interface zu höheren Schichten **QoS (Quality of Service)** Garantien gewährleistet. Während seiner Einführung gab es allerdings zu wenig Anwendungen, die direkt auf ATM aufsetzten. Zu dieser Zeit war TCP/IP bereits Standard. Ohne eine TCP/IP Anbindung konnte ATM nicht verkauft werden. Heute wird ATM noch in wenigen Gebieten benutzt, allerdings wird für WAN meistens SDH eingesetzt.

Synchronous Digital Hierarchy (SDH)

SDH Technologie hat in den letzten Jahren beispielsweise das deutsche B-WIN (ATM) mit dem G-WIN (Gigabit-Wissenschaftsnetz) abgelöst. Seit 2006 wurde mit X-WIN die Topologie komplett erneuert und mit der Integration von DWDM (Dense Wavelength Division Multiplexing) sind mit bis zu 160 parallelen Übertragungen eine Kapazität von 1,6 TBit/s möglich. Die in den USA analoge Technologie nennt sich **Synchronous Optical Network (SONET)**.
SDH Netzwerke sind hierarchisch aufgebaut, d. h. es gibt Regionen mit beispielsweise einem beliebigen 155 MBit/s Netzwerk (oft Ring-Topologie), an welches dann entweder lokale Netzwerke (oder auch MANs) oder überregionale größere SDH Netze mit einer höheren Kapazität angeschlossen werden. Zum Verbinden dieser Netzwerke werden **SDH Cross Connect** bzw. **Add/Drop Multiplexer** eingesetzt. Die höchste mit SDH mögliche Datenrate beträgt etwa 40 GBit/s. Auf diese Weise können bis zu vier Datenströme zuzüglich Kontrollinformationen mit der nächsthöheren Ebene verbunden werden und die Verbindungen eignen sich außerdem gut für Sprachübertragungen. Zu beachten ist, dass Verbindungen nicht nur über die nächsthöheren bzw. -niedrigeren Ebenen erfolgen müssen. Außerdem ist eine Synchronisation sehr wichtig, um die korrekten Daten im Datenstrom wiederfinden zu können. Dafür hat man einen direkten Zugriff auf die jeweiligen Frames die nicht erst vollständig empfangen werden müssen wodurch es nur zu kurzen Verzögerungen beim Einfügen und Extrahieren von Signalen kommt.
Die Datenpakete bei SDH werden „Container“ oder „Transport Module“ genannt. Für die verschiedenen Bandbreiten gibt es jeweils ein **Synchronous Transport Module** in entsprechender Größe, das sich STM-N nennt, wobei $N = 1$,4,16 oder 64 beträgt. Jeder Container besteht aus 9 Zeilen mit jeweils $270 * N$ Byte, wobei $9 * N$ Byte für Kontrollinformationen und $261 * N$ Byte für den Payload verwendet werden. Ein STM-1 Container kann in $125\mu s$ übertragen werden. In den Kontrolldaten enthalten ist ein **Regenerator Section Overhead (RSOH)** und ein **Multiplex Section Overhead (MSOH)** in denen Informationen die Verbindung zwischen

zwei Repeatern bzw. zwei Add/Drop Multiplexern betreffend enthalten sind. In letzteren müssen zusätzlich Routinginformationen gespeichert werden. Weiterhin gibt es **Administrative Unit Pointers** mit denen im Falle einer fehlerhaften Synchronisation der direkte Zugriff auf die Daten im Payload möglich gemacht werden kann. Um die Synchronisation zu korrigieren, werden die Container dazu in jedem Fall zum richtigen Zeitpunkt abgeschickt und im Falle von zu spät ankommenden Daten der Payload später versetzt angehangen bzw. bei zu früh ankommenden Daten bereits in das AUP-Feld geschrieben.

Der Payload enthält außerdem 9 Byte mit einem **Path Overhead (POH)**, der zusätzliche Kontrolldaten wie z. B. Informationen über Alternativrouten, über die Übertragungsqualität und Kommunikationskanäle zu Wartungszwecken enthält. Durch das Hinzufügen dieser POH-Bytes wird der Container zu einem sog. **Virtual Container**. Mehrere Container werden dann in größeren STM-Payloads zu **Tributary Unit Groups** zusammengefasst und durch Hinzufügen von Administrative Unit Pointers wird eine Tributary Unit Group wiederum zu einer **Administrative Unit (AU)**. Schließlich werden noch die bereits angesprochenen SOH Bytes (RSOH und MSOH) ergänzt, die beispielsweise Bits zur Synchronisation, Fehlererkennung, Steuerung von alternativen Pfaden, uvm. enthalten. Eine STM-4 kann also aus 4 STM-1 bestehen usw. Das Multiplexen geschieht dabei auch für die Kontrolldaten Byte für Byte, wobei jedes Byte eine Datenrate von 64 kBit/s für Sprachübertragung hat (Abb. 13).

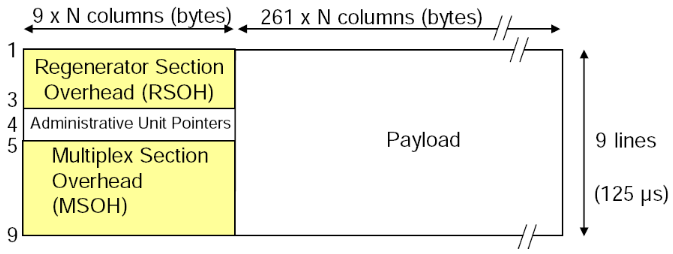


Abbildung 13: SDH-Container

2.7 Das Letzte Meile Problem

Beim Problem der letzten Meile geht es darum, wie der Endnutzer schlussendlich an das öffentliche Internet angeschlossen werden kann. Dies geschieht bisher durch Nutzen der bereits vorhandenen Telefonleitungen, die dann für beliebigen Datenverkehr eingesetzt werden.

Modem

Zunächst erläutern wir das Problem, wie die zu sendenden digitalen Daten über ein analoges Medium übertragen werden können. Dabei ist zu beachten, dass von den Vermittlungsstellen der Telefongesellschaft nur die für Sprachübertragung verwendeten Frequenzen im Bereich von 300 bis 3400 Hz verarbeitet werden. Das **Modem** (**Mo**dulator - **Dem**odulator) ist nun dazu da, die digitalen Informationen in verwertbare analoge Daten in beide Richtungen umzuwandeln. Dabei ist die Fehleranfälligkeit leider sehr hoch. Datenraten bis zu 56 kBit/s sind möglich (Abb. 14).

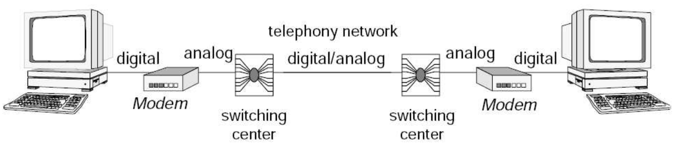


Abbildung 14: Modemverbindung

Um Bits auf eine Trägerwelle zu modulieren, gibt es eine Vielzahl von Varianten. Man kann die Codierung z. B. durch unterschiedliche Amplituden (**Amplitude Shift Keying (ASK)**) oder durch verschiedene Frequenzen (**Frequency Shift Keying (FSK)**) oder auch durch unterschiedliche Phasen (**Phase Shift Keying (PSK)**) festlegen. Beim **Quaternary Phase Shift Keying (QPSK)** werden beispielsweise mit vier verschiedenen Phasen vier unterschiedliche Zustände codiert, so dass für beide Übertragungsrichtungen zwei zur Verfügung stehen.

Für 56 kBit/s Modems gibt es 128-PAM (**Pulse Amplitude Modulation**) bei der alle 125μs 128 unterschiedliche Spannungslevel erkannt werden können was vom Prinzip her wieder den Kabelcodes ähnelt.

Bei **Quadrature Amplitude Modulation (QAM)** werden Amplituden- und Phasenmodulation kombiniert. Dieses Verfahren ist von 4-QAM (wie QPSK) bis zu 64-QAM erweiterbar, wobei jeweils mehr Bits codiert werden.

Integrated Services Digital Network (ISDN)

Wie man es von ATM her kennt, ist es möglich, Sprache digital zu übertragen wodurch man auf ein Analogisieren von digitalen Daten verzichten kann. Bei **ISDN** werden nun mehrere verschiedene Kommunikationsdienste wie z. B. für Sprache, Fax oder Daten integriert und digital übermittelt. Dadurch sind nicht nur höhere Übertragungsraten sondern auch zusätzliche Funktionen (beispielsweise mehrere Telefonnummern, Rufnummernübermittlung, Weiterleitung, Konferenzen, uvm.) möglich. Insgesamt beträgt die Übertragungsrate für einen ISDN-Basisanschluss 144 kBit/s und besteht aus zwei unabhängigen B-Kanälen (64 kBit/s) und einem D-Kanal (16 kBit/s) für zusätzliche Datenübertragung. Diese drei Kanäle werden dann mit Hilfe von **Time Multiplexing** übermittelt. Ein ISDN-Primärmultiplexanschluss bietet 30 B-Kanäle und einen D-Kanal sowie einen zusätzlichen Kanal für Synchronisation und Wartung mit jeweils 64 kBit/s und insgesamt 2 MBit/s an. Geplant war ein B-ISDN (Broadband-ISDN) Anschluss mit einer höheren Bandbreite, der aber aufgrund zuvieler Probleme nicht erfolgreich war.

Digital Subscriber Line (DSL)

Vorteilhaft bei DSL ist auch hier, dass die bestehenden Telefonleitungen zur Übertragung genutzt werden können. Es werden die oberhalb von Sprachtelefonie (bis 3,4 kHz) und ISDN (bis 20 kHz) genutzten Frequenzen (40 kHz bis 1,1 MHz bei ADSL, bis zu 2,2 MHz bei ADSL2+ und bis zu 30 MHz bei VDSL2) genutzt um Übertragungsraten bis zu 50 MBit/s zu ermöglichen. Dabei ist zu beachten, dass mit steigender Entfernung zur Vermittlungsstelle das Signal schwächer wird und besonders höhere Frequenzen nicht mehr korrekt interpretiert werden können. Die Übertragungsraten sind bei unterschiedlichen Entfernungen wie folgt:

Entfernung	**Downstream**	**Upstream**
1,4 km	12,96 MBit/s	1,5 MBit/s
0,9 km	25,86 MBit/s	2,3 MBit/s
0,3 km	51,85 MBit/s	13 MBit/s

Aus diesem Grund wird die Übertragungsrate bei Verzerrungen entsprechend angepasst. Aufgrund der verschiedenen Varianten von DSL (HDSL, SDSL, ADSL, VDSL) gibt es die allgemeine Bezeichnung xDSL. Bei ADSL beträgt die Senderate 16-640 kBit/s und die Empfangsrate 1,5-9 MBit/s.
Die zur Verfügung stehenden Frequenzen werden nun durch **Discrete Multitone Modulation (DMT)** in mehrere Kanäle mit jeweils ~4 kHz Bandbreite eingeteilt wobei üblicherweise für den Downstream mehr als für den Upstream zugewiesen werden. Jeder dieser Kanäle verwendet dann ein passendes Modulationsverfahren von QPSK für höhere und schwächere Frequenzen bis zu 64-QAM für niedrige Frequenzen.
Ein **Splitter** sorgt dann an der Endverbindung mit Hoch- und Tiefpassfiltern dafür, dass die niedrigen Frequenzen an ISDN-Geräte bzw. analoge Telefone und die hohen Frequenzen an das **DSL-Modem** weitergeleitet werden. Letzteres verarbeitet schließlich die verschiedenen DSL-Kanäle und sendet einen verwertbaren Datenstrom an den Computer.
An der Vermittlungsstelle werden die DSL-Daten von einem **DSL Access Multiplexer (DSLAM)** empfangen, der mehrere DSL-Leitungen zusammenführt, überwacht (beispielsweise um den Traffic eines Benutzers zu zählen) und an ein WAN (meist SDH) weiterleitet.

3 Protokolle und Dienste

3.1 Vermittlungsschicht 3 (Network Layer)

Während die ersten beiden Schichten prinzipiell nur für die Übermittlung von Daten zwischen zwei angrenzenden Computern sorgen, stellt die Vermittlungsschicht nun die Grenze zwischen dem Netzwerk und dem Benutzer dar und ist für das Verbinden von Sub-Netzwerken, eine globale Adressierung, das Routen von Paketen, das Management eines gesamten Netzwerks und eine globale Flusskontrolle zuständig. In Routern ist die dritte Schicht die oberste Schicht und es werden dort die Headerinformationen ausgewertet, die nächste Verbindung ausgewählt und das Paket mit einem neuen Header entsprechend weitergeleitet.
Zum Übertragen dieser Pakete gibt es nun zwei Möglichkeiten: In einer **verbindungslosen Kommunikation** werden die Daten spontan ohne Reservierungen von Station zu Station weitergeleitet. Dies ist zwar sehr leicht zu implementieren, allerdings besteht die Gefahr, dass Pakete unterschiedliche Pfade entlang laufen und in der falschen Reihenfolge am Empfänger ankommen (siehe auch *Packet Switching*).
Bei einer **verbindungsorientierten Kommunikation** wird zwischen beiden Stationen eine feste Verbindung aufgebaut über die dann der Transfer läuft. Hierbei kann eine Kapazität (z. B. in einer virtuellen Verbindung) reserviert werden und die Flusskontrolle ist gesichert (siehe auch *Circuit Switching*).
Auch wenn nun im Internet teilweise Daten verbindungsorientiert übertragen werden, wird die globale Kommunikation verbindungslos durchgeführt.

Routing

Die wichtigste Funktion der Vermittlungsschicht betrifft das Adressieren und Routen. Jeder Computer kann weltweit durch eine einzigartige Adresse identifiziert werden und jeder Router verwaltet eine Tabelle, in der die ausgehenden Verbindungen für bestimmte Ziele gespeichert werden. Bei verbindungslosen Übertragungen muss

dabei nur das Ziel, bei verbindungsorientierter Übertragung auch die Sendestation gespeichert werden wobei hier zusätzlich Identifikationsnummern verwendet werden, wodurch auch die Tabellen größer werden und bei einer Unterbrechung eine komplett neue Verbindung aufgebaut werden muss. Verbindungsorientierte Kommunikation wird z. B. für Sprachübertragungen verwendet.
Die bevorzugte Route kann so gewählt werden, dass kurze Antwortzeiten, ein hoher Datendurchsatz oder ein kürzester Pfad erreicht werden. Es können dabei auch Überlastsituationen oder Sicherheitskriterien eine Rolle spielen. Letztendlich ist es jedoch nicht möglich, eine optimale Route zu entscheiden da ein Router keine vollständige Informationen über das Netzwerk hat und jede Entscheidung das Netzwerk für einen bestimmten Zeitrahmen beeinflusst.
Bei Überlastsituationen kann sich die Leistung des Netzwerks rapide verschlechtern, da Pakete bei Fehlern in der Übertragung erneut gesendet werden müssen. Daher ist es erforderlich, dieses frühzeitig zu erkennen und durch **Traffic Shaping** die Sender zum Anpassen der Datenrate anzuweisen. Überlast kann durch mehrere Faktoren erkannt werden, beispielsweise mit Hilfe der durchschnittlichen Länge der Routerwarteschlange, in der alle Pakete der eingehenden Leitungen zwischengespeichert werden. Andere Möglichkeiten wären die Anzahl der verworfenen Pakete oder der wiederholten Übertragungen von Paketen zu beobachten.

3.2 Das Internet Protokoll (IP)

Der Vorgänger des Internets war das ARPANET (Advanced Research Project Agency Network) in dem mehrere Universitäten zu einem Netzwerk zusammengeschlossen wurden. Ein Subnetz bestand damals aus mehreren **Interface Message Processors (IMP)** die eine hohe Konnektivität und eine hohe Verlässlichkeit sicherstellen sollten. Für den Transfer IMP-IMP oder Host-IMP gab es unterschiedliche Protokolle und das Integrieren weiterer Netzwerke gestaltete sich aufgrund verschiedener Protokolle schwierig. Aus diesem Grund wurden einheitliche Protokolle für die Transport- und Vermittlungsschicht entwickelt und es entstand 1974 das **Transmission Control Protocol/Internet Protocol** welches eine hohe Fehlertoleranz, maximal mögliche Zuverlässigkeit und Verfügbarkeit und hohe Flexibilität für Anwendungen mit unterschiedlichen Anforderungen ermöglichen sollte und 1983 das offizielle Protokoll des ARPANETs wurde. Nach und nach wurde das Netzwerk immer mehr erweitert bis hin zu Interkontinentalverbindungen und nannte sich schließlich das „Internet“. Heute geschieht auch firmeninterne Kommunikation in einem **Intranet** häufig mit Hilfe von TCP/IP-Protokollen, da heterogene Netzwerkstrukturen dort leicht integriert werden können.
Das Netzwerkprotokoll **IP** für die dritte Schicht verpackt die Daten in Pakete und sendet sie verbindungslos und unabhängig voneinander durch das Netz. Zusätzlich gibt es deshalb zwei weitere End-zu-End-Protokolle, das **Transmission Control Protocol (TCP)** für einen verlässlichen verbindungsorientierten Transfer und das **User Datagram Protocol (UDP)** für einen verbindungslosen Transfer mit weniger Overhead. Im Unterschied zum OSI-Referenzmodell gibt es beim TCP/IP-Referenzmodell einer nicht weiter ausgeführten **Netzzugangsschicht (Host-to-Network Layer)** der eine Art Platzhalter für verschiedene Datenübertragungstechniken ist und im OSI-Referenzmodell der ersten und zweiten Schicht entspricht. Anschließend folgt eine **Internetschicht (Internet Layer)** (IP) und eine **Transportschicht (Transport Layer)** (TCP/UDP). Ganz oben arbeiten die Anwendungsprogramme auf einer **Anwendungsschicht (Application Layer)** mit diesen Protokollen zusammen. Es handelt sich also um eine ganze Familie von Protokollen und das IP-Protokoll bildet ihren Kern. Wegen der kleinen Anzahl von zentralen Protokollen wird der TCP/IP-Protokollstack auch mit dem sog. **Sanduhrmodell** beschrieben (Abb. 15).
Die Internetschicht hat drei Aufgaben, die in unterschiedliche Protokolle aufgeteilt sind: Die Adressierung wird über das Transferprotokoll IPv4 oder IPv6 geregelt. Routingentscheidungen werden nur an Routern benötigt und sind in Routing Protokollen festgelegt. Schließlich gibt es Kontrollprotokolle wie beispielsweise ICMP, ARP, RARP und IGMP.

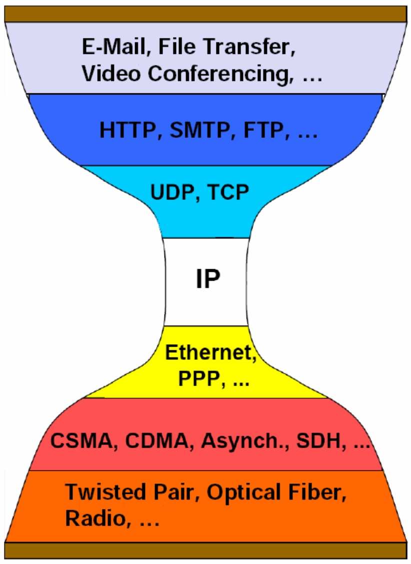


Abbildung 15: Sanduhrmodell

Internet Protokoll (IP)

Das Internet Protokoll sorgt für eine verbindungslose Übertragung von Datenpaketen bis zu einer Größe von maximal 64 kByte (normalerweise 1500 Byte) zwischen Sender und Empfänger. Die Kommunikation ist transparent, da die Netzwerkinfrastruktur vor den oberen Protokollen versteckt wird. Die Adressierung geschieht mit IPv4 über logische 32 Bit-Adressen, die hierarchisch verwendet werden. Auf diese Weise können bestimmte Subnetze aufgebaut werden und Routingentscheidungen vereinfacht werden. Es gibt ferner 3 Netzwerkklassen und 4 Adressformate (einschließlich Multicast). Für den Fall, dass die Paketgröße für ein bestimmtes Netzwerk zu groß ist, sorgt IP außerdem für das Fragmentieren und Zusammensetzen dieser Pakete.

Ein IP-Paket besteht nun aus einem normalerweise 20 Byte großem IP-Header und Nutzdaten in variabler Größe (Abb. 16). Zu Beginn geben 4 Bit Auskunft über die **Version** des Pakets (binär codierte 4 für IPv4). Darauf folgen weitere 4 Bit mit der **IP Header Length**, die in einem Vielfachen von 32 Bit die Größe des Headers angibt. Bei einem leeren Optionsfeld würde hier also für 20 Byte = 160 Bit eine 5 stehen. Das 8 Bit große Feld **Type of Service (TOS)** sollte ursprünglich für Prioritätsangaben verwendet werden. Diese Funktionalität wurde aber wegen zuviel Rechenaufwand in Routern bisher nicht eingebaut. Es folgt das Feld **Total Length** in der in 16 Bit die Gesamtlänge des IP-Pakets gespeichert ist (maximal 64 kB).

In den nun folgenden 32 Bit wird das Fragmentieren und Zusammensetzen von Paketen geregelt. Das 16 Bit große **Identification**-Feld gibt eine Kennung des Pakets an, die bei allen Teilpaketen gleich ist. Es folgt ein reserviertes Bit, ein **Don't Fragment (DF)**-Bit das angibt, wenn ein Paket nicht fragmentiert werden darf (es wird im Zweifelsfall verworfen) und ein **More Fragments (MF)**-Bit das weitere Teilpakete ankündigt. In den übrigen 13 Bit wird in 64 Bit bzw. 8 Byte Schritten die Position des Fragments angegeben.

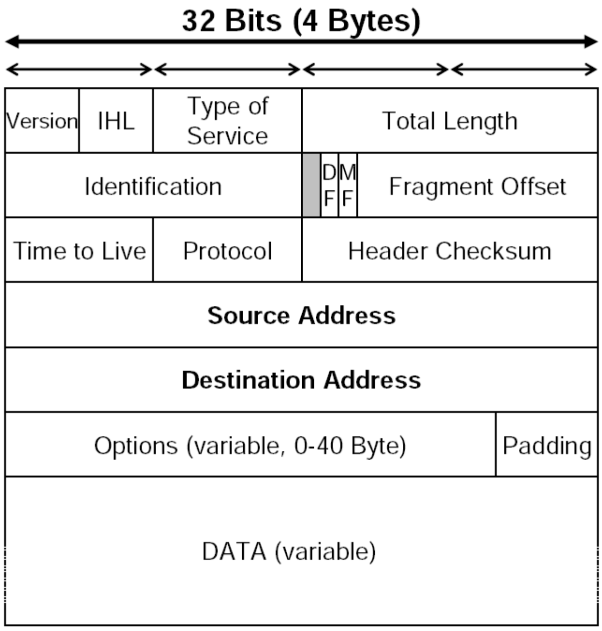


Abbildung 16: IP Header

Damit Pakete bei einer Fehlkonfiguration nicht unendlich lange weiterverschickt werden, gibt es im IP-Header 8 Bit, in der ein **Time to Live (TTL)**-Wert eingetragen wird, der von jedem Router dekrementiert wird. Erreicht der Wert 0, wird das Paket gelöscht. Nun folgen 8 Bit für ein **Protocol**-Feld, in welchem das Folgeprotokoll spezifiziert wird. Für ein TCP-Paket steht hier z. B. der Wert 0x06, für ein UDP-Paket der Wert 0x11. Der **Header Checksum** gibt in 16 Bit eine Prüfsumme für den IP-Header an, die sich durch das Dekrementieren der TTL auch bei jedem Weiterleiten durch einen Router ändern müsste. Das Berechnen der Prüfsumme kostet allerdings verhältnismäßig viel Zeit, so dass dies aus Performancegründen meist nicht durchgeführt wird. Schließlich ist in den letzten $2 * 32$ Bit die Sendeadresse und die Zieladresse codiert.

Nachfolgende 0-40 Byte können für weitere Optionen verwendet werden, wobei die derzeit 5 spezifizierten Erweiterungen (**Security**, **Strict Source Routing**, **Loose Source Routing**, **Record Route**, **Time Stamp**) praktisch nicht von den gängigen Routern unterstützt werden, da sie u.A. Rechenzeit kosten und in manchen Fällen die 40 Byte auch nicht ausreichend sind. Die Optionen müssen ein Vielfaches von 32 Bit = 4 Byte betragen. Sind sie das nicht, muss mit 0-Bits aufgefüllt werden (Padding).

Netzklassen

IP Adressen sind nun hierarchisch strukturiert und in sog. **Netzklassen** unterteilt. Beginnt die Adresse mit einer 0 handelt es sich um ein Klasse-A-Netzwerk, das durch die anschließenden 7 Bit festgelegt wird. Es besteht aus (abzüglich 2 für besondere Zwecke) 126 Netzwerken mit jeweils 2^{24} Hosts (beginnend mit 1.0.0.0). Ein Klasse-B-Netzwerk wird durch die Bits 10 identifiziert und die folgenden 14 Bits geben das Netzwerk an. Ein Klasse-B-Netzwerk kann also 2^{16} Hosts besitzen (beginnend mit 128.0.0.0). Schließlich gibt es 2^{21} Klasse-C-Netzwerke, die durch 110 identifiziert werden und 256 Hosts besitzen (beginnend mit 192.0.0.0). Klasse-D-Netzwerke (beginnend mit 1110) definieren sog. **Multicast**-Adressen und Klasse-E-Netzwerke (beginnend mit 1111) wurden für eine spätere Verwendung reserviert (Abb. 17).

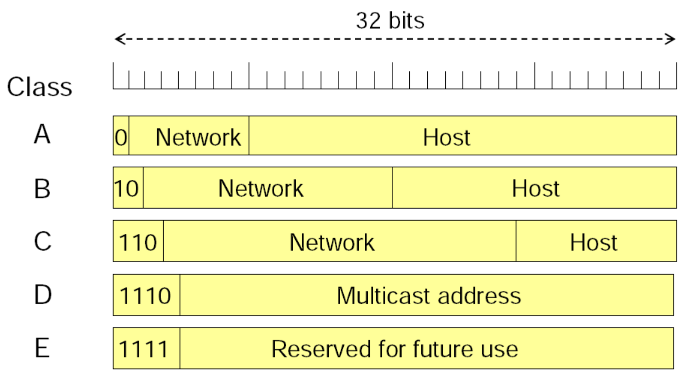


Abbildung 17: Netzklassen

Zu beachten ist, dass Router für jedes Netzwerk eine eigene IP Adresse besitzen müssen. Sie lesen in einem IP-Paket die ersten Bits der Zieladresse und können über ihre intern verwaltete Tabelle ablesen, an welchem Port dieses weitergeleitet werden muss. Für alle übrigen Pakete gibt es eine Default-Route. Innerhalb eines Netzes können schließlich eigens definierte Subnetze gebildet werden, so dass insbesondere größere Netze mit einer großen Anzahl von IP Adressen nochmal aufgeteilt werden können. Dies geschieht mit einer **Subnetzmaske** in Form einer IP-Adresse, die durch die Anzahl der 1-Bits den Hostteil vom Netzwerkteil trennt (Abb. 18). Um anzugeben, in welchem Subnetz sich eine Adresse befindet, verwendet man die Schreibweise 137.226.12.221/24, wobei die letzte Zahl die Anzahl der Bits für die Netzwerkadresse angibt.

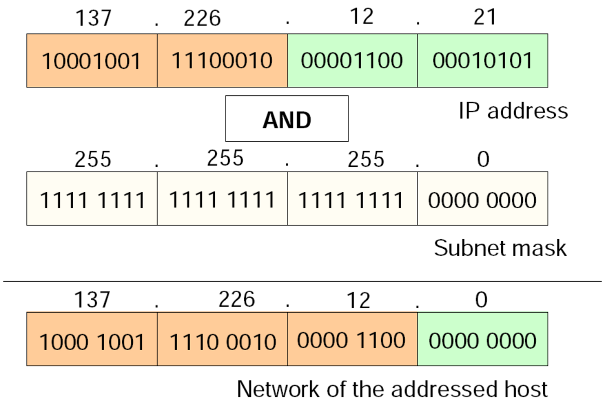


Abbildung 18: Netzmaske

Einige besondere Adressen sind:

0.0.0.0	-	Aktueller Host
0.0.HOST	-	Host im aktuellen Netzwerk
255.255.255.255	-	Broadcast ins aktuelle lokale Netzwerk
NETWORK.0.0	-	Reserviert für das (Sub)netz
NETWORK.255.255	-	Broadcast in ein anderes Netzwerk
127.x.x.x	-	Loop; nicht ins Netzwerk senden (localhost)

Außerdem gibt es einige reservierte Adressblöcke, die zur privaten Nutzung im eigenen Netzwerk verwendet werden können, da derartige Pakete nicht ins Internet verschickt werden. Dazu gehören die Bereiche 10.0.0.0 - 10.255.255.255, 172.16.0.0 - 172.31.255.255 und 192.168.0.0 - 192.168.255.255.
Durch die Einteilung von IP Adressen in Netzklassen besteht nun das Problem, dass diese ineffizient an verschiedene Organisationen und Firmen verteilt wurden. Ein großes Klasse-A-Netzwerk bieten normalerweise zu viele Adressen und ein kleines Netz ist möglicherweise nicht ausreichend. Heute sind die verfügbaren IP Adressen knapp geworden weshalb die neue Version **IPv6** entwickelt wurde, mit der 128 Bit für Adressen zur Verfügung stehen. Damit wurde der Adressraum von ∼4,3 Milliarden auf ∼340 Sextillionen vergrößert. Aufgrund von Interoperabilität, Kosten und Migrationsproblemen gestaltet sich die Einführung von IPv6 allerdings schwierig.

Classless Inter-Domain Routing (CIDR)

Das unflexible Netzklassenkonzept wird heute in Routern nicht mehr verwendet. Falls beispielsweise ein einzelnes Klasse-C-Netzwerk nicht ausreicht und deshalb mehrere von diesen verwendet werden, müsste ein Router für diese verschiedenen Netze auch mehrere Einträge haben, obwohl sie alle zu einer Firma oder Organisation gehören. Stattdessen verwendet man Netzmasken, um mehr Netzwerke in unterschiedlichen Größen definieren zu können (Abb. 19). Die Router identifizieren das Netz dann nicht mehr an den ersten Bits sondern an der zusätzlich gespeicherten Netzmaske (z. B. 137.250.3/17). Backbone Router, beispielsweise an transatlantischen Verbindungen, benötigen deshalb nur kleine Routingtabellen, da sie nur etwa die ersten 13 Bits betrachten müssen.

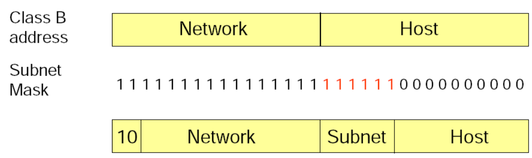


Abbildung 19: Klasse-B-Netz und Subnetzmaske

Network Address Translation (NAT)

In verschiedenen Fällen wird vom Router beim Versenden von Paketen ins Internet durch eine **NAT Box** eine Übersetzung der lokalen IP Adressen (meist aus einem privaten Adressblock) in eine oder mehrere globale IP Adressen vorgenommen. Dadurch kann die Anzahl der IP Adressen künstlich erweitert werden und die interne Netzwerkstruktur nach aussen hin versteckt. In der einfachsten Variante **Basic NAT/Static NAT** hat der Router einen Vorrat an IP Adressen und weist diese den lokal verwendeten Adressen zu wodurch jedoch für jeden Rechner wieder eine Adresse benötigt wird. Alternativ dazu können IP Adressen auch bei Bedarf zugewiesen werden, aber auch hier werden in Extremfällen eine große Anzahl an Adressen benötigt.
Bei der heute meist verwendeten Variante **Hiding NAT** oder auch **NAPT (Network Address Port Translation)** wird nun jede lokale IP Adresse auf die gleiche externe Adresse übersetzt. Hier entsteht das Problem, dass eingehende Pakete zunächst nicht eindeutig einem lokalen Host zugeordnet werden können. Daher wird an dieser Stelle im Payload der Daten ein globaler Port festgelegt, der beim Empfänger ausgelesen wird. Zu beachten ist, dass Ports Teil des TCP-Protokolls sind.
Für vollständig eingehende Verbindungsanfragen funktioniert dies jedoch nicht. Besitzt das Netzwerk beispielsweise einen Webserver, so muss im Router manuell Port 80 auf die entsprechende lokale IP Adresse eingetragen werden. Aufwändigere Szenarios mit z. B. mehreren Webservern sind mit NAPT nicht möglich.

IPv6

Nicht nur die geringe Anzahl an IP Adressen sondern auch der Bedarf nach weiteren Verbesserungen hat dazu geführt, dass 1995 **IPv6** unter RFC 1883 spezifiziert wurde. Es bietet eine einfachere Headerstruktur, mehr Automatisierungen, einfachere Konfigurationen, Leistungsverbesserungen, mehr Sicherheit und einen sehr großen Adressbereich.
[.. Siehe Folie 10, S.40ff. ..]

3.3 Routing

Der zweite Bestandteil der Internetschicht betrifft Routingentscheidungen. Router müssen Tabellen führen, in denen Zielnetzwerke und die entsprechende nächste erreichbare Station gespeichert sind. Genaueres Wissen über die dort vorhandenen Netztopologien muss dieser Router nicht haben, da das weitere Routing dann lokal vorgenommen wird. Weiterhin gibt es eine **Default Route** an die alle Pakete geleitet werden, für die keine genaueren Informationen in der Tabelle gespeichert sind.
Um nun diese Tabellen aufzubauen gibt es mehrere Protokolle. Jedes einzelne Netzwerk arbeitet autonom und kann ein anderes **Interior Gateway Protocol (IGP)** benutzen. Diese Arten von Protokollen zeichnen sich durch besondere Fähigkeiten im Umgang mit komplizierten Netzwerktopologien aus. Die hier besprochenen Protokolle sind das veraltete **Routing Information Protocol (RIP)** und **Open Shortest Path First (OSPF)**.
Exterior Gateway Protocols (EGP) dagegen dienen dazu, autonome Systeme zu verbinden und Informationen über ihre Erreichbarkeit auszutauschen. Dazu gehört das heute auch eingesetzte **Border Gateway Protocol (BGP)**.
In einem Subnetz können nun mehrere Router auf verschiedene Weise miteinander verbunden sein, wobei sich die Verbindungen durch unterschiedliche Charakteristika unterscheiden und redundant sein können. Ziel ist es, möglichst kurze und effiziente Routen zu erhalten. Das Ergebnis einer solchen Optimierung kann ein **Sink Tree** sein, der keine Schleifen enthält und als Indikator für die Qualität eines Routingalgorithmus verwendet werden kann. Zu beachten ist jedoch, dass sich ein Sink Tree durch Ausfall eines Routers oder Wegfall einer Verbindung ändern kann.

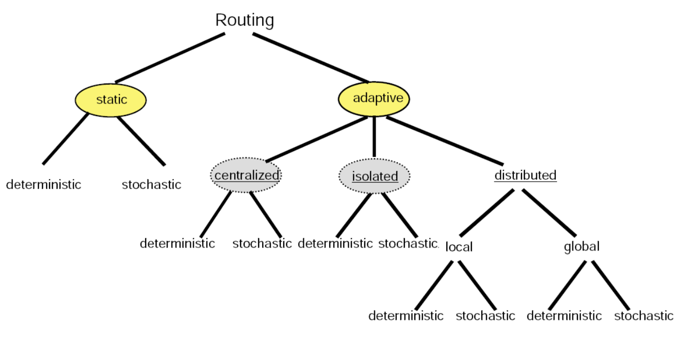


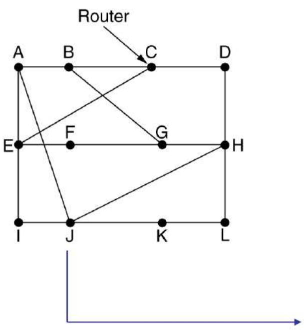
Abbildung 20: Routing-Algorithmen

Man unterscheidet zwischen vier verschiedenen Routingalgorithmen. Bei **statischem Routing** werden die Tabellen von Hand festgelegt und ändern sich nicht. Durch *Source Routing* kann die Route alternativ vom Sender festgelegt und in den IP Header geschrieben werden. Das *Internal Routing*, bei dem die Tabellen im Router manuell eingetragen werden, gestaltet sich einfach, kann jedoch bei Fehlern im Netz zum Zusammenbrechen von ganzen Verbindungen führen. Statisches Routing wird deshalb heute nur noch in lokalen Netzwerken verwendet.
Ein anderer Algorithmus, der heute noch in manchen Fällen verwendet wird ist das **Flooding** bei dem das Paket an allen anderen ausgehenden Ports

weitergeleitet wird. Dieses Verfahren bietet eine hohe Verlässlichkeit, führt aber zu einer hohen Auslastung des Netzes. Bei Schleifen zirkuliert das Paket bis zum Ablaufen der TTL. Es kann jedoch zum Messen einer Referenzverzögerung für andere Routingalgorithmen verwendet werden, da der optimale Pfad auf jedenfall durchlaufen wird. Flooding wird außerdem zur Unterstützung von anderen Routingalgorithmen (z. B. OSPF) verwendet.
Beim **adaptiven Routing** unterscheidet man zwischen **zentralisierten**, **isolierten** und **verteilten** Routingalgorithmen. Beim zentralisierten Routing gibt es ein **Network Control Center (NCC)**, das Informationen über den Netzwerkstatus sammelt und auswertet. Alle betroffenen Router werden dann von diesem regelmäßig mit Routingstrategien und Informationen über das Netzwerk versorgt, die endgültige Routingentscheidung wird jedoch lokal von den Routern durchgeführt. Dieses Prinzip funktioniert verständlicherweise nur in sehr kleinen Netzwerken, da das Sammeln und Übertragen dieser Informationen Zeit kostet und die Daten schnell veraltet sein können.
Ein anderes Verfahren nennt sich **Local Estimation Procedure**. Hier werden die Übertragungsverzögerungen auf allen Ports gemessen und die Tabellen entsprechend der schnellsten Verbindung gefüllt. Der Nachteil ist, dass hier jeder Router für sich selbst die Schätzungen vornimmt und im Extremfall ein Paket zwischen zwei Routern hin- und hergeschickt wird.
Eine weitere Möglichkeit besteht darin, das Paket an den Port mit der geringsten Auslastung weiterzuleiten. Dieses Verfahren nennt sich „Hot Potato“ und funktioniert eher schlecht, da nicht garantiert ist, dass ein Paket beim Empfänger ankommt.
Zusätzlich können alle Verfahren nicht nur deterministisch sondern auch stochastisch implementiert werden, wodurch mehrere ausgehende Ports mit einer bestimmten Wahrscheinlichkeit ausgewählt werden.

Kürzester Pfad, Dijkstra-Algorithmus

Angenommen, jeder Router besitzt nun ausreichend Wissen über den Aufbau des Netzwerks und die Kosten jeder Verbindung. Hiermit können nun dann mit Hilfe des Dijkstra-Algorithmus die kürzesten Pfade zu den Zielknoten berechnet werden. In der adaptiven Variante werden dann dynamische Routingalgorithmen verwendet, mit denen dann die Routingtabellen regelmäßig erneuert werden, da sich die Metrik des Netzwerkes natürlich fortwährend verändert (neue oder wegfallende Knoten sowie verändernde Kosten für Verbindungen).
Der Dijkstra-Algorithmus funktioniert wie folgt: Ausgehend von einem „Arbeitsknoten“ werden jedem erreichbaren Knoten die Verbindungskosten und der Vorgängerknoten zugewiesen. Der Knoten mit dem kürzesten Pfad wird als „permanent“ markiert und ausgehend von diesem werden wieder alle von dort erreichbaren Knoten ausgewählt und dort ihre Kosten sowie Vorgängerknoten gespeichert. Gegebenenfalls wird ein Eintrag überschrieben, falls ein kürzerer Pfad gefunden wurde. Der Algorithmus endet, wenn von allen Knoten die ausgehenden Verbindungen geprüft wurden.



To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

Vectors received from J's four neighbors

Abbildung 21: Distance Vector Routing

Die adaptive Variante zum **Shortest Path Routing** nennt sich **Distance Vector Routing**. Hierbei verwaltet jeder Router eine Tabelle mit gewussten oder geschätzten Entfernungen/Kosten zu jedem Ziel und dessen zugewiesenen Verbindungen zu den Nachbarknoten. Dabei wird angenommen, dass jeder Router die Kosten zu seinen Nachbarn kennt. In regelmäßigen Abständen werden diese Informationen dann zu den Nachbarknoten gesendet wo der Router seine Tabelle dann (ohne Beachtung seiner alten Informationen) erneut berechnet. Hierbei werden dann die Kosten der direkten Verbindungen neu ermittelt. In Abb. 21 wird beispielhaft gezeigt, wie Router J mit den empfangenen Vektoren von seinen Nachbarn seinen neuen Distanzvektor berechnet.
Ganz zu Beginn einer Verbindung sendet jeder Router lediglich einen Distanzvektor mit Informationen über sich selbst, also etwa $V_A = ((A = 0))$. Die Router, die diesen Vektor empfangen addieren die Kosten der jeweiligen Verbindung und fügen Einträge über diesen Router in ihren Tabellen hinzu. Außerdem senden sie ihre eigenen Informationen an den neuen Router, der wiederum seine Tabelle füllen kann.
Wichtig bei diesem Verfahren ist, dass die Informationen verlässlich sein müssen. Beim sog. „Christmas Deadlock“ sendete ein Router beispielsweise einen Distanzvektor, in dem er fälschlicherweise ankündigte, jeden Router mit Kosten 0 erreichen zu können, woraufhin fast der gesamte Datenverkehr über ihn umgeleitet wurde und das Netz zusammenbrach. Andere Nachteile sind ein zusätzlicher Overhead und das Problem, dass es eine gewisse Zeit dauert, bis die nötigen Informationen im Netz verteilt wurden. Aufgrund dieser Tatsache eignet sich

Distance Vector Routing nicht für größere Netzwerke oder sogar das ganze Internet. Außerdem wird die Last in vielen Fällen nicht günstig verteilt. Bei zwei Verbindungen mit ähnlichen Kosten wird z. B. zunächst alles über die eine Verbindung geroutet die also stärker belastet ist während die andere wenig genutzt wird. Dies wird festgestellt woraufhin sich die Tabelle derartig ändert, dass nun sämtliche Daten über die freie Leitung verschickt werden, usw.
Weitere Probleme ergeben sich durch das Propagieren der Distanzvektoren: Wenn eine Verbindung ausfällt wird an den entsprechenden Knoten an dieser Stelle der Wert *inf* eingetragen. Diese Information wird wiederum weitergeleitet und von benachbarten Routern angenommen, dass eine bestimmte Verbindung nun nicht mehr besteht. Bis jedoch eine mögliche Alternativroute gespeichert wird, vergehen mehrere Schritte. Außerdem kann ein „Bouncing Effect“ eintreten, wenn beim Ausfall einer Verbindung ein anderer Router bereits seine (dann nicht mehr gültige) Routingtabelle an den Router übergibt, der gerade erst den (korrekten) Wert *inf* eingetragen hat. Der Wert des anderen Routers ist natürlich in dem Moment niedriger, wird daher angenommen und erneut weitergeleitet wobei die Empfänger ihre Werte erneut anpassen, usw. Im schlimmsten Fall wird bei einem derartigen Problem unendlich oft weitergezählt („Count-to-Infinity-Effekt“), beispielsweise wenn in einer Reihe ein Router hinzugefügt und wieder entfernt wird.
Abhilfe schafft hier der **Split-Horizon Algorithmus** der einem Router verbietet, die Kosten für eine Verbindung über die selbe Verbindung (zurück) zu schicken. Aber auch hier gibt es Fälle, in denen die (fehlerhaften) Informationen über einen Umweg zu dem problematischen Router geschickt werden und wiederum unendlich lange gezählt wird.

Routing Information Protocol (RIP)

Das **Routing Information Protocol (RIP)** ist nun ein Internal Gateway Routing Protocol, das im Internet verwendet wird und auf dem Distanzvektorprotokoll basiert: RIP Nachrichten werden alle 30 Sekunden als UDP Pakete verschickt und als Metrik wird die Anzahl der Hops verwendet, die allerdings auf 15 begrenzt ist (wobei 15 für *inf* steht). Neben dieser Begrenzung können weiterhin nur bis zu 25 Einträge der Routingtabelle verschickt werden. Daraus resultiert, dass dieses Protokoll nur langsam konvergiert und sich deshalb nur für kleine Systeme eignet. Außerdem besteht die Möglichkeit des Count-to-Infinity-Effekts und es werden keine Subnetze beachtet.
Die erweiterte Version **RIPv2** unterstützt deshalb Subnetze, Authentifizierung, Multicast und weitere Funktionen, die Anzahl der Hops ist jedoch immer noch auf 15 begrenzt.
Von Cisco wurde in dieser Zeit das **Interior Gateway Routing Protocol (IGRP)** mit einer erweiterten Metrik (Beachten der Übertragungszeit), Aufteilen der Last auf zwei Leitungen und einem effizienteren Paketformat entwickelt. Dieses konnte sich jedoch nicht durchsetzen, da es sich um Cisco-spezifische Implementierungen handelte.

Link-State-Routing-Protokoll

Während beim Distanzvektorprotokoll globale Informationen nur lokal mit den direkten Nachbarn ausgetauscht wird, werden beim **Link-State-Routing** sog. **Link-State-Advertisements (LSA)** per Multicast global an alle Router ausgetauscht. Dies geschieht folgendermaßen: Zunächst ermittelt ein Router mit sog. „HELLO Paketen“ seine direkten Nachbarn und ihre Adressen. Mit „ECHO Paketen“, auf welche Router sofort antworten müssen, wird außerdem die Zeitverzögerung gemessen. Die auf diese Weise ermittelten Informationen werden dann durch **Flooding** in LSAs per Multicast an alle anderen Router verschickt wo jeder Router mit den Informationen über das gesamte Netzwerk die kürzesten Pfade berechnet (meistens mit Dijkstra-Algorithmus). Das ganze wird regelmäßig wiederholt.
Für den Fall, dass mehrere Router in einem Broadcast-Netzwerk untereinander verbunden sind, wird ein neuer virtueller Knoten eingeführt, der diese Router verbindet.
Ein LSA enthält nun neben der Senderidentifikation und den Verbindungskosten zu den benachbarten Knoten außerdem eine Sequenznummer und ein Alter, die Probleme durch Flooding vermeiden sollen. Das Alter (angegeben in Sekunden) hat die gleiche Funktion wie die TTL beim Internetprotokoll. Ihr Wert wird kontinuierlich inkrementiert und die Informationen bei einem Maximalwert nicht mehr verwendet. Die Sequenznummer wird mit jeder neuen Version eines LSA heraufgezählt, so dass ältere Versionen ebenfalls identifiziert und gelöscht werden können. Beim Erreichen der größten Sequenznummer wird das LSA mit einem maximalen Alterswert in das Netz verschickt, was zum Löschen der Informationen führt. Anschließend kann von vorne gezählt werden.
Weiterhin gibt es **Transmission Flags**, die von einer Station gesetzt werden, wenn ein LSA zu der zugehörigen Station geschickt wird und **Confirmation Flags** die bestätigen, dass ein LSA empfangen wurde. Beispielsweise wird eine LSA einer Station A an eine Station B verschickt und dann weiter nach C und F sowie zurück nach A geschickt. Die Transmission Flags sind dann 011 und die Confirmation Flags 100 für ACF. Wenn ein LSA von zwei Stationen kommt, wird es entsprechend nur an die übrigen weitergeleitet während die anderen eine Bestätigung erhalten.

Probleme beim Link-State-Routing sind zum einen, dass bei n Routern und m Nachbarn nm Tabelleneinträge benötigt werden und große Tabellen benötigen viel Speicher, Rechenzeit und Übertragungskapazität. Zum anderen sind bei Routerausfällen sofort alle Netzwerkgraphen nicht mehr aktuell und Tabelleneinträge sind höchstwahrscheinlich fehlerhaft. Der wichtigste Punkt ist jedoch, dass dieses Verfahren sehr anfällig gegen Attacken ist, da leicht fehlerhafte Informationen verteilt werden können.
Das Problem der großen Routingtabellen wird durch hierarchisches Routing gelöst indem das Netzwerk in Regionen eingeteilt wird. Hierdurch ist es zwar möglich, dass die Pfadlänge nicht mehr ganz optimal ist, jedoch werden in einer Routingtabelle eines Routers nur noch die Stationen innerhalb der selben Region sowie zusammenfassend nur noch alle übrigen Regionen gespeichert. Soll beispielsweise ein Paket in eine Station in Region 3 verschickt werden, so wird nur festgehalten, dass Region 3 über Router 1C mit 2 Hops erreicht werden kann.

Internal Gateway Routing Protocol: OSPF

1990 wurde das Link-State-Routing-Protokoll **Open Shortest Path First** von der IETF (Internet Engineering Task Force) standardisiert. Es unterstützt eine Vielzahl an Metriken (Entfernung, Verzögerung, etc.), Lastenverteilung zwischen redundanten Verbindungen, Sicherheitsmechanismen gegen falsche Routinginformationen oder Attacken und das zuvor beschriebene hierarchische Routing. Es werden drei Verbindungstypen unterstützt: Point-to-Point-Verbindungen zwischen Routern, Broadcast-Netzwerke sowie WAN-Netzwerke ohne Broadcasting.
Das Internet ist nun aufgeteilt in sog. **autonome Systeme (AS)** und große autonome Systeme erneut in einzelne Bereiche (Area). Jedes autonome System ist über ein **Backbone** mit allen Teilen des AS verbunden und jeder Router, der zu zwei oder mehr Bereichen gehört, ist Teil dieses Backbones. Innerhalb dieser Bereiche arbeitet jeder Router mit der gleichen Link-State-Datenbank und muss den gleichen Algorithmus für den kürzesten Pfad verwenden.

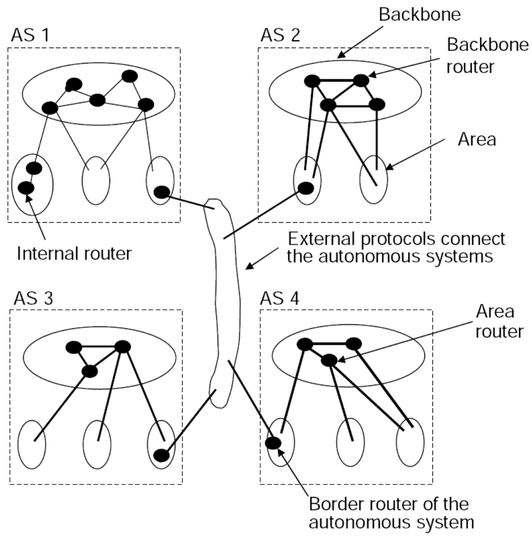


Abbildung 22: Routerklassen

OSPF unterscheidet also zwischen 4 Routerklassen (Abb. 22):

- **Interne Router** gehören nur zu einem Bereich
- **Bereichsrouter** verbinden zwei oder mehr Bereiche
- **Backbone Router** gehören zum Backbone
- **AS Grenzrouter** vermitteln zwischen mehreren autonomen Systemen

External Gateway Protocol: BGP

Während der Schwerpunkt von Internal Gateway Protocols auf Effizienz liegt, werden bei **External Gateway Protocols** eher politische oder ökonomische Richtlinien betrachtet. Das **Border Gateway Protocol (BGP)** funktioniert ähnlich wie das Distanzvektorprotokoll, es werden jedoch vollständige Pfade ausgetauscht, die sich an Sicherheits- und politischen Aspekten orientieren. Zum Austausch der Daten wird TCP verwendet.

3.4 Hilfsprotokolle

Zusätzlich zum Übertragungsprotokoll IPv4 bzw. IPv6 und Routingprotokollen gibt es eine Vielzahl von zusätzlichen Kontrollprotokollen: Beispielsweise das **Address Resolution Protocol (ARP)**, das **Reverse Address Resolution Protocol (RARP)**, das **Internet Control Message Protocol (ICMP)** sowie das **Internet Group Management Protocol**.

Address Resolution Protocol (ARP)

Da innerhalb eines lokalen Netzwerks die Hardware auf den unteren Schichten keine IP-Adressen kennt, muss der Sender die MAC-Adresse (48 Bit) des Empfängers kennen. Dazu wird bei Bedarf ein **ARP Request** über die lokale Broadcast-Adresse verschickt, welcher die benötigte IP Adresse enthält. Der Rechner mit der korrekten IP Adresse antwortet dann mit einer **ARP Response**, welche die angeforderte MAC Adresse enthält. Die ermittelten Zuordnungen von IP und MAC Adresse werden eine gewisse Zeit in einer lokalen Tabelle gespeichert, es wird also nicht für jedes Paket eine neue Anfrage verschickt. Das Verfahren kann optimiert werden, indem jeder Rechner gelegentlich ARP Requests an seine eigene IP Adresse verschickt. Da jedes ARP Paket auch die eigene IP und MAC Adresse enthält können so alle Empfänger die Zuordnungen im lokalen ARP Cache speichern.

Reverse Address Resolution Protocol (RARP)

Mithilfe von **Reverse ARP** können bekannte MAC-Adressen auf einem RARP Server gespeichert werden. So kann mit einem **RARP Request** zu einer bestimmten MAC Adresse eine (möglicherweise fest zugewiesene) IP Adresse angefragt werden. Problem hierbei ist, dass diese Anfragen über die Broadcast-Adresse verschickt werden und diese nicht von Routern weitergeleitet werden. Alternativ bietet sich deshalb ein **DHCP Server** an. Mit **DHCP Discover** Paketen (Broadcast), die von einem **DHCP Relay** auch über andere Netzwerke verschickt werden können, kann so nicht nur die IP Adresse, sondern auch Subnetzmaske, Domainnamen und andere Informationen übertragen werden. DHCP eignet sich deshalb besonders für die vollständige Hostkonfiguration eines Rechners.

Internet Control Message Protocol (ICMP)

Das **Internet Control Message Protocol (ICMP)** gehört zu Schicht 3, obwohl es eigentlich auf IP aufbaut. Es wird benutzt, um Kontrollnachrichten in IP Paketen zu verschicken. Im IP-Header ist der Servicetyp dann immer 0 und die Protokollnummer immer 1. Zum einen können beispielsweise im Falle eines Fehlers von Routern ICMP Nachrichten erstellt werden, die dann den Absender darüber informieren, zum anderen werden auch aktiv – beispielsweise von dem Programm *ping* – ICMP Nachrichten verschickt. Im Payload des IP Pakets gibt es für eine ICMP Nachricht deshalb 16 Bit für einen Typ sowie einen Code, in dem die Art der Nachricht spezifiziert wird. Weiterhin befindet sich dort eine Prüfsumme und je nach ICMP Nachricht zusätzliche Daten. Neben Fehlermeldungen, dass z. B. das Ziel nicht erreicht werden konnte und dem angesprochenen Echo Request/Reply für *ping* wird ICMP beispielsweise auch für das Programm *traceroute* verwendet.

Multicast

Während mit Unicast eine Übertragung zwischen genau zwei Hosts bezeichnet wird und bei Broadcast ein Sender Daten an alle Teilnehmer des Netzwerks verschickt, besteht in manchen Fällen der Bedarf, eine bestimmte Gruppe von Rechnern zu adressieren. Dies geschieht mit **IP Multicast** für das es den reservierten Adress-Bereich 224.0.0.0 bis 239.255.255.255 (Klasse D) gibt. Spezielle Multicast Adressen sind außerdem reserviert. Mithilfe des **Internet Group Management Protocols (IGMP)** werden IGMP Nachrichten in IP Paketen verschickt, mit denen ein Host einem Router mitteilen kann, dass er zu einer bestimmten Multicast Gruppe gehören möchte. Die Router verwalten dann separate Tabellen für diese Multicast Gruppen und tauschen untereinander diese Informationen beispielsweise über das **Distance Vector Multicast Routing Protocol (DVMRP)** aus. Zu beachten ist, dass Router nicht zwangsweise Multicast unterstützen müssen. In diesem Fall müssen die IP Multicast Pakete in normalen IP Paketen verpackt werden und durch IP Tunneling per Unicast zu anderen Multicast Routern verschickt werden.

3.5 Transportschicht 4 (Transport Layer)

Der eigentliche Kern der Protokollhierarchie, die **Transportschicht**, verbindet die Anwendungen mit der Vermittlungsschicht und sorgt für einen verlässlichen und verbindungsorientierten oder verbindungslosen Datentransfer. Es muss ein Kommunikationsprozess am Rechner zugewiesen werden, Fehler müssen erkannt und behandelt werden, eine Flusskontrolle muss gegeben sein und es sollten mehrere Verbindungen zusammengefügt (bei ungenügender Kapazität seltener auch aufgeteilt) werden können ack/seq ∈ {0, …, 15}

Timeout for TPDU 2 (rows 8–9)

	A	Message	B	
1	→	<request 8 buffers>	→	A asks for 8 places in B's buffer
2	←	<ack = 15, buf = 4>	←	B grants 4 buffer places for A and waits for TPDU 0
3	→	<seq = 0, data = m0>	→	A sends TPDU 0
4	→	<seq = 1, data = m1>	→	A sends TPDU 1
5	→	<seq = 2, data = m2>		A sends TPDU 2, TPDU is lost
6	←	<ack = 1, buf = 3>	←	B acknowledges TPDU 0 and 1, buffer places are reduced to 3
7	→	<seq = 3, data = m3>	→	A sends TPDU 3
8	→	<seq = 4, data = m4>	→	A sends TPDU 4
9	→	<seq = 2, data = m2>	→	A repeats to send TPDU 2
10	←	<ack = 4, buf = 0>	←	B acknowledges TPDU 2, 3, 4 - sender is stopped (no buffer)
11	←	<ack = 4, buf = 1>	←	B informs A about 1 buffer place
12	←	<ack = 4, buf = 2>	←	B informs A about 2 buffer places
13	→	<seq = 5, data = m5>	→	A sends TPDU 5
14	→	<seq = 6, data = m6>	→	A sends TPDU 6; A is blocked (no more buffer)
15	←	<ack = 6, buf = 0>	←	B acknowledges TPDU 5 and 6, but sender remains blocked
16		<ack = 6, buf = 4>	←	B informs A about 4 buffer places

Abbildung 23: Verbindungsvorgang

In einem vereinfachten Transportprotokoll werden die eigentlichen Daten in eine **Transport Protocol Data Unit (TPDU)** verpackt, diese in einem IP Paket und dieses wiederum im Frame Payload von Schicht 2 verschickt. Eine verbindungsorientierte Übertragung kann nun beispielsweise hergestellt werden, indem die Sende- und Empfangsstationen 5 verschiedene Operationen verstehen: Der Empfänger signalisiert durch LISTEN die Empfangsbereitschaft, ein Sender fordert eine Verbindung mit CONNECT an, die auf Empfängerseite bestätigt wird. Anschließend können mit SEND und RECEIVE die gewünschten Daten übertragen werden. Zum Schluss wird die Verbindung durch DISCONNECT beendet.

Damit beim Verbindungsaufbau keine Pakete verloren oder auch dupliziert beim Empfänger eintreffen, wird beim Verbindungsaufbau eine Sequenznummer verwendet und ein **Three-Way Handshake** durchgeführt, wobei jeweils Bezug auf die Sequenznummer genommen wird: 1. Verbindungsanfrage, 2. Bestätigen und akzeptieren des Empfängers, 3. Bestätigung des Senders, dass die Bestätigung des Empfängers angekommen ist und die Sendeanfrage noch gültig ist. Anschließend kann die eigentliche Datenübertragung beginnen. Ein ähnliches Vorgehen wird auch beim Verbindungsabbruch verwendet. Nach Timeouts werden ggf. erneute Anfragen bzw. Bestätigungen gesendet.

Gewöhnlich müssen eine große Anzahl von Verbindungen behandelt werden und Datenpakete können verloren gehen, in falscher Reihenfolge oder sogar doppelt am Sender ankommen. Eine funktionierende Flusskontrolle ist somit wichtig. Das Sliding Window Prinzip wird hier aufgrund der unterschiedlichen Verbindungen mit einer variablen Fenstergröße eingesetzt und Pufferspeicher wird dynamisch zugewiesen (Abb. 23).

In manchen Fällen können wichtige Nachrichten verloren gehen, so dass beide Stationen in ein Deadlock geraten und unendlich lange warten würden. Aus diesem Grund werden regelmäßig Kontroll-TPDUs verschickt.

Ein derartiges Transportprotokoll, bei dem jede Station bestimmte Zustände haben kann und Anfragen sowie Bestätigungen gesendet werden, kann auch als endlicher Automat dargestellt werden.

3.6 Transportprotokolle TCP und UDP

Die zwei verwendeten Protokolle der Transportschicht sind das **Transmission Control Protocol (TCP)**, das eine verlässliche und verbindungsorientierte Übertragung ermöglicht sowie das **User Datagram Protocol (UDP)**, das verbindungslos und ohne Flusskontrolle arbeitet und für den schnellen Datentransfer (beispielsweise bei Audiostreaming) eingesetzt wird. Diese Transportprotokolle werden letztendlich von den Anwendungen zur Kommunikation benutzt.

Transmission Control Protocol (TCP)

Vom **Transmission Control Protocol (TCP)** wird ein zu sendender Bytestream in Segmente von maximal 64 kB eingeteilt. Typischerweise hat ein TCP-Segment eine Größe von 1500 Bytes, so dass es in ein IP-Paket passt, das aufgrund der Größe eines Ethernetframes von 1500 Bytes ebenfalls eine entsprechende Größe besitzt. Die Header eines TCP- und eines IP-Pakets sind jeweils 20 Byte groß, nach Abzug von 8 Byte für den Header des meistens eingesetzten Point-to-Point Protokolls (PPP) verbleiben also 1452 Bytes für Nutzdaten in einem Paket. Beim Senden besteht die Möglichkeit, dringende Nachrichten auch außerhalb der Flusskontrolle übermitteln zu können.

Die gesonderte Adressierung einer Anwendung geschieht dann durch **Ports**. Im TCP-Header wird ein Quellport und ein Zielport eingetragen, die auf der Empfängerseite gelesen und an die korrekte Anwendung weitergeleitet werden. Auf diese Weise wird eine logische Verbindung zwischen zwei **Sockets** hergestellt. Ein Socket besteht aus der IP-Adresse und einer 16 Bit-Portnummer was 48 Bit für die Adressinformation ergibt. Zu beachten ist, dass ein Socket für mehrere Verbindungen gleichzeitig genutzt werden kann und TCP-Verbindungen immer Full-Duplex und Point-to-Point-Verbindungen sind. Eine TPDU wird auch **Segment** genannt und diese werden neben dem Datentransfer auch für den Auf- und Abbau einer Verbindung, das Einigen auf eine Fenstergröße sowie für Bestätigungen verwendet.
Für bekannte Dienste sind bestimmte Ports von 0 bis 1024 fest definiert. Ports von 1025 bis 65535 können beliebig von jedem Host spezifiziert werden.

TCP Header

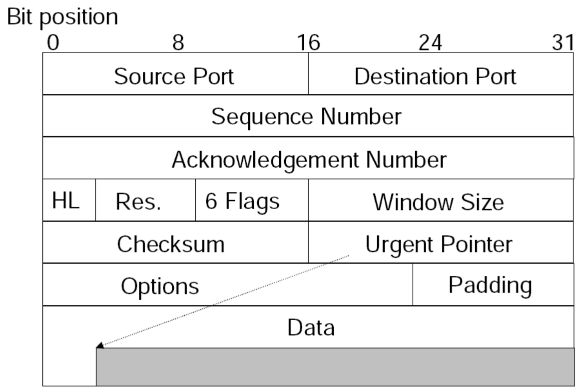


Abbildung 24: TCP Header

Der 20 Byte (+Optionen) große TCP Header (Abb. 24) enthält zunächst den Quell- und Zielport, sowie eine Sequenznummer (SEQ) und eine Bestätigungsnummer (ACK). Wie im IP Header wird anschließend auch hier die Länge des Headers in 32-Bit-Blöcken angegeben (4 Bit). Es folgen 6 reservierte Bits, die Null sein müssen. Danach geben 6 Kontrollbits mögliche Zustände an: URG (Daten, auf die ein Urgent Pointer zeigt, werden sofort von der Anwendung bearbeitet), ACK, PSH (Daten sofort an Anwendung weiterleiten), RST (Reset, Verbindung abbrechen), SYN (Verbindung initiieren) und FIN (Verbindung freigeben). Nun folgt ein 16 Bit Feld mit einer Fenstergröße, indem die Anzahl der Bytes angegeben wird, die der Sender des Paketes bereit ist zu empfangen. Dieser Wert kann durch einen Eintrag im Optionsfeld multipliziert werden. Im Prüfsummenfeld wird eine Prüfsumme, die einen **TCP-Pseudo-Header** und die Daten mit einbezieht, eingetragen. Vor möglichen Optionen folgt schließlich noch das Feld für einen Urgent Pointer. Der TCP-Pseudo-Header wird nicht mitgesendet, aber von Sende- und Empfangsstation zur Berechnung der Prüfsumme mit einbezogen und dient somit zur Erhöhung der Zuverlässigkeit. Er enthält die IP-Adresse des Absenders und des Empfängers, füllenden Nullen, der Protokollnummer (6 für TCP) und die Länge des TCP-Segments in Bytes (Abb. 25).

Source address (IP)		
Destination address (IP)		
00000000	Protocol = 6	Length of the TCP segment

Abbildung 25: TCP-Pseudo-Header

TCP Verbindungsmanagement

Zum Verbindungsaufbau führ ein Client eine CONNECT-Operation auf einen Server durch, der sich im LISTEN-Modus befindet. Dies geschieht durch ein SYN-Paket mit einer beliebigen Sequenznummer x. Wenn die Verbindung angenommen wird, sendet der Server ebenfalls ein SYN-Paket mit einer Sequenznummer y und antwortet gleichzeitig auf die Anfrage mit ACK und der Bestätigungsnummer $x + 1$. Im dritten Schritt bestätigt der Client dies mit ACK und der Bestätigungsnummer $y + 1$ sowie der Sequenznummer $x + 1$. Sollten zwei Rechner gleichzeitig eine Verbindung aufbauen wollen, wird trotzdem nur eine erstellt.
Bei der Datenübertragung (Full-Duplex) werden empfangene Bytes mit ACK bestätigt und als Bestätigungsnummer das nächste benötigte Byte angegeben. Werden beispielsweise 10 Byte Daten mit Sequenznummer 101 und Bestätigungsnummer 201 gesendet, wird auf dieses Paket mit Sequenznummer 201 und Bestätigungsnummer 111 geantwortet. Wenn keine Daten gesendet werden, erhöht sich die Bestätigungsnummer nicht! Tritt ein Timeout auf, werden die Daten erneut gesendet. Üblicherweise wird das Verfahren Go-Back-N oder Selective Repeat eingesetzt.
Beim Verbindungsabbruch wird ein FIN-Paket mit einer Sequenznummer x und einer Bestätigungsnummer y verschickt. Darauf wird erneut mit FIN und der Sequenznummer y sowie der Bestätigungsnummer $x + 1$

geantwortet. Im dritten Schritt wird wieder mit der Sequenznummer $x + 1$ und Bestätigungsnummer $y + 1$ geantwortet.
Die Fenstergröße, die dynamisch vom Sender in jedem Paket erneut angegeben werden kann, zählt die Bytes, die im Pufferspeicher noch zur Verfügung stehen. Wie es dann beim Sliding Window Verfahren üblich ist, wird nach dem Empfang von ACK mit einer Bestätigungsnummer n das Fenster um n Byte verschoben. Wird als Fenstergröße 0 angegeben, muss die Sendestation warten, bis Daten im Empfangspuffer verarbeitet wurden und eine neue Meldung über ein größeres freies Fenster eintrifft. Damit dabei nicht das **Silly Window Syndrom** auftritt bei dem immer nur sehr kleine Datenmengen von beispielsweise 1 Byte in einem Paket mit 20 Byte Header übertragen werden, sollte der Empfänger die nächste Fensteraktualisierung erst ab einem bestimmten freien Platz verschicken.

Überlaststeuerung (Congestion Control)

Um eine Überlast am Empfänger zu vermeiden gibt es das Flusskontrollfenster. Es besteht allerdings auch die Möglichkeit, dass Pakete aufgrund von fehlender Netzwerkkapazität verloren gehen. Um dies zu minimieren gibt es eine **Überlaststeuerung**. Dabei wird angenommen, dass Paketverlust nicht durch Übertragungsfehler sondern durch die angesprochenen Überlastsituationen auftritt. Jeder Sender speichert dazu neben dem Flusskontrollfenster noch ein **Congestion Window (cwnd)**, in dem eine vom Netzwerk mögliche Kapazität festgehalten wird. Das Minimum beider Fenster ergibt dann die Anzahl der Bytes, die maximal gesendet werden dürfen. Dieses wird zunächst mit der maximalen Segmentgröße (MSS) initialisiert.
Beim **Slow Start Algorithmus** wird das Congestion Window nun mit jeder innerhalb eines Timeouts eintreffenden Bestätigung (ACK) verdoppelt, bis sie den Wert des Flusskontrollfensters erreicht. Erfolgt bis zum Timeout keine Bestätigung, wird das Fenster auf den Initialwert zurückgesetzt. Zusätzlich wird ab einem Schwellwert (**ssthresh**), der zu Beginn 64 kByte beträgt, das Fenster nur noch linear um eine maximale Segmentgröße erhöht (**Congestion Avoidance**). Bei einem Timeout wird der Schwellwert auf die Hälfte des zuvor erreichten Fensters gesetzt.
Das Verfahren kann verbessert werden, indem der Sender nach drei doppelt erhaltenen Bestätigungen (Dup-Acks) und damit drei wiederholten Anfragen für ein fehlerhaftes Paket dieses sofort – und hoffentlich bevor ein Timeout eingetroffen ist – nocheinmal verschickt (**Fast Retransmit**). Der Schwellwert wird in diesem Fall nur halbiert (mindestens jedoch auf doppelte MSS) und die Fenstergröße auf $ssthresh + 3 * MSS$ gesetzt, da drei empfangene Dup-Acks für eine gute Verbindung stehen. Für jedes weitere Dup-Ack wird die Fenstergröße entsprechend weiter erhöht (**Fast Recovery**). Nach dem Empfang der nächsten normalen Bestätigung werden Fenstergröße und Schwellwert normal fortgesetzt.

Timermanagement bei TCP

Auf der Sicherungsschicht kann der Timeout-Timer leicht ermittelt werden, da die Round Trip Time (RTT) relativ stabil ist. Da die Wartezeiten bei TCP Verbindungen wesentlich variabler sind, werden hier deshalb mehrere Timer eingesetzt, die dynamisch angepasst werden. Wählt man nun die RTT zu klein, werden zu viele unnötigen wiederholten Übertragungen durchgeführt. Ist der Wert zu groß wird zu lange bei Paketverlust gewartet. Nach dem Algorithmus von Jacobson (1988) wird die RTT erneuert, wenn eine Bestätigung vor Ablaufen des Timers t eintrifft: $RTT = \alpha * RTT + (1 - \alpha) * t$. α ist ein Glättungsfaktor, der gewöhnlich $0,875$ beträgt. Der Timeout wird dann berechnet mit $Timeout = \beta * RTT$.
Zunächst wurde β auf 2 gesetzt, was sich allerdings als zu unflexibel erwies, da die Werte mehr oder weniger stark gestreut sein können. Daher wurde von Jacobson β proportional zur Standardabweichung der Ankunftszeit der Bestätigungen gesetzt: $\beta = \alpha * \beta + (1 - \alpha) * |RTT - t|$. Der Timeout beträgt hiermit: $Timeout = RTT + 4 * \beta$.
Als sehr einfache Alternative wird in vielen TCP Implementierungen der Timeout jedoch bei rechtzeitig eintreffenden Bestätigungen lediglich verdoppelt.
Weitere verwendete Timer sind beispielsweise ein **Persistence Timer**, der nach einer Meldung eines leeren Pufferspeichers gestartet wird und nach Ablaufen zum Verschicken eines Testsegments sorgt, ein **Keep-Alive Timer**, der nach längerer Inaktivität zur Überprüfung der Verbindung sorgt und ein **Time Wait Timer**, der beim Verbindungsabbruch gestartet wird.

User Datagram Protocol (UDP)

Im Gegensatz zu TCP sorgt das **User Datagram Protocol (UDP)** für eine verbindungslose und unverlässliche Verbindung zwischen zwei Anwendungen. Der Vorteil liegt jedoch in einem schnellen Austausch von Informationen mit wenig Overhead, da der Header nur 8 Byte groß ist. Es wird außerdem kein Three-Way-Handshake durchgeführt, keine Bestätigungen verschickt und es besteht die Gefahr, dass Pakete verloren gehen, doppelt oder in falscher Reihenfolge übermittelt werden. Es besteht jedoch die Möglichkeit, dass die Anwendungsprogramme diese Funktionalität implementieren.

UDP Header

Im 8 Byte Header eines UDP Pakets wird der Quell- und Zielport, die Gesamtlänge von Header und Daten in 32 Bit Worten und eine optionale Prüfsumme angegeben (zur Berechnung wird ebenfalls ein Pseudo-Header wie bei TCP benutzt). Alle Werte haben eine Größe von 16 Bit.

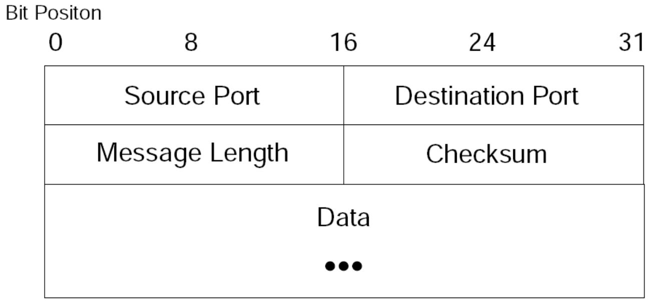


Abbildung 26: UDP Header

Anwendungen, die UDP verwenden, sind beispielsweise DNS (Port 53), BOOTP (Port 67/68), TFTP (Port 69), SNMP (Port 161/162) und RIP (Port 520).

3.7 Firewalls

[. . .]

4 Anwendungsprotokolle

4.1 Sitzungsschicht (5), Darstellungsschicht (6), Anwendungsschicht (7)

Sitzungsschicht 5 (Session Layer)

Auf der Sitzungsschicht werden an bestimmten Stellen einer Verbindung Kontrollpunkte erstellt, an denen die Sitzung nach einem Ausfall wieder synchronisiert werden kann. Dadurch wird vermieden, dass bei Übertragungsfehlern wieder von vorne begonnen werden muss. Die 5. Schicht sorgt außerdem durch Verwendung von Token zur Authorisierung während des Dialogmanagements.

Darstellungsschicht 6 (Presentation Layer)

Auf der Darstellungsschicht wird dafür gesorgt, dass die gesendeten Daten auf Sender- und Empfängerseite die gleichen Bedeutungen haben. Dazu werden Daten in einer systemabhängigen Darstellung wie z. B. ASCII (American Standard Code for Information Interchange, 7 Bit) oder EBCDIC (Extended Binary Coded Digital Interchange Code, 8 Bit) in ein für beide Systeme verständliches Format **ASN.1 (Abstract Syntax Notation One)** gebracht. Die Darstellungsschicht übernimmt außerdem Aufgaben wie Datenkompression und Verschlüsselung.

[Informationstheorie. . .]

[Huffman Code. . .]

Anwendungsschicht 7 (Application Layer)

In der obersten Schicht sind eine Reihe von oft benötigten Kommunikationsfunktionen implementiert, welche die Anwendungen dann benutzen können. Dazu gehören z. B. HTTP, FTP, Telnet, SMTP, DNS, SNMP und TFTP.
Die Anwendungsprotokolle der Anwendungsschicht sind für besondere Zwecke definiert und spezifizieren die Typen der gesendeten Nachrichten sowie deren Syntax und Semantik. Außerdem definieren sie Regeln, wann und wie eine Anwendung Nachrichten senden bzw. auf eine Nachricht antworten kann. Gewöhnlich handelt es sich um eine Client/Server-Struktur und die Prozesse verwenden TCP(UDP)/IP-Sockets.

4.2 DNS

Das **Domain Name System (DNS)** sorgt für die Auflösung einer Internetadresse in Form eines symbolischen Namens (z. B. `www.google.de`) zu einer IP-Adresse. Bei einer Anfrage an eine solche symbolische Adresse stellt ein **Resolver** eine Anfrage an einen **Nameserver**, dessen Aufgabe es ist, die passende IP-Adresse zu ermitteln. Dazu kann ein lokaler Nameserver bereits die benötigten Informationen enthalten, ansonsten wird die Anfrage an entfernte Nameserver weitergereicht.
Zur besseren Strukturierung ist der **Domain-Namespace** wie ein Baum aufgebaut, dessen maximale Tiefe auf 127 Ebenen beschränkt ist. Eine **Domain** kann bis zu 63 Zeichen lang sein und bezeichnet den Unterbaum eines Knotens. Jede Domain kann weiterhin in **Sub-Domains** aufgeteilt werden. An oberster Stelle gibt es den Root-Knoten der mit „“ (leeres Label) bezeichnet wird. Die Namen einer Domain bestehen aus einer Sequenz von Labels, die durch „`.`“ getrennt sind. Im Blattknoten ist schließlich die IP-Adresse (und oft auch Hardwareinformationen, Mail-Routing-Informationen, etc.) zu dem entsprechenden Namen gespeichert. Ein vollständiger Name wird als **Fully Qualified Domain Name (FQDN)** bezeichnet, z. B. `infomatik.rwth-aachen.de.` (der letzte Punkt wird üblicherweise weggelassen). Domainnamen, die sich nicht auf den Root-Knoten sondern auf eine andere Domain beziehen nennt man **relative Domainnamen**. Weiterhin bezeichnet man alle Kinder des obersten Wurzelknotens als **Top-Level Domain**, deren Kinder als **Second Level Domain** usw. Ursprünglich wurde der Namespace in 7 Top-Level Domains (`com`, `edu`, `gov`, `mil`, `net`, `org`, `int`) zuzüglich Domains für jedes Land eingeteilt. Mittlerweile wurden diesen aber weitere hinzugefügt. Innerhalb mancher Länder wurden weitere Konventionen zur Namensstrukturierung getroffen, z. B. `co.uk` (commercial organizations) und `ac.uk` (academic organizations) in Großbritannien; Deutschland verblieb unstrukturiert.
Jede Domain kann nun von einer eigenen Organisation verwaltet werden und entsprechende Sub-Domains besitzen. Um auf einen Host zu verweisen, dessen Position sich ändert, können Rechnern weitere **Domain Name Aliases** zugeordnet werden, die auf einen anderen Domainnamen verweisen. Ändert sich dann der Host, muss nur der Eintrag dieses Aliases geändert werden.
Den Bereich, den ein Nameserver für einen bestimmten Teil eines Namensraums abdeckt, nennt man nun **Zone** und Informationen über eine Zone kann der Nameserver entweder von einer lokalen Datei oder einem anderen Nameserver beziehen, der dann die Authorität über diese Zone besitzt. Zu beachten ist, dass ein Nameserver auch für mehrere Zonen verantwortlich sein kann. Der Vorteil von diesen Zonen ist, dass sie üblicherweise kleiner als eine ganze Domain sind und entsprechende Server einen kleineren Teil des Namensraums verwalten müssen. Beispielsweise hat die Top-Level Domain `ca` (Canada) die Sub-Domains `ab.ca` (Alberta), `on.ca` (Ontario) und `qc.ca` (Quebec). Die Domain `ca` deckt alle Daten in `ca`, sowie `ab.ca`, `on.ca` und `qc.ca` ab, die Zone `ca` enthält jedoch nur die benötigten Informationen für `.ca`, die hauptsächlich aus Zeigern zu den Sub-Domains bestehen. Ein Nameserver lädt also nur die jeweiligen Zonendaten anstelle sämtlicher Informationen der ganzen Domain.
Zur Redundanz gibt es neben einem **primären Nameserver** einer Zone in der Regel weitere **sekundäre Nameserver**. Letztere empfangen die Daten von einem anderen authoritativen Nameserver (meist dem primären Nameserver), für den die Daten von einem Administrator manuell eingetragen werden. Sowohl primäre als auch sekundäre Nameserver sind somit authoritativ für die entsprechende Zone.
Beim Auflösen eines Namens gibt es nun die rekursive und die iterative Methode: Falls ein Nameserver einen Namen nicht auflösen kann fragt dieser bei einer **rekursiven Namensauflösung** direkt den nächsten Nameserver, der ihm die gesuchte IP-Adresse zuschickt. Bei einer **iterativen Namensauflösung** würde nur die Adresse des übergeordneten Nameservers zurückgeliefert werden und die anfragende Station müsste die Anfrage selbst erneut durchführen. In der Praxis werden beide Methoden verwendet: Der lokale Nameserver fragt rekursiv den Rootserver nach der Adresse und andere Nameserver liefern iterativ die entsprechenden Adressen der passenden Nameserver zurück. Auf einer unteren Ebene befindet sich möglicherweise ein lokal verwalteter Bereich, deren untergeordneten Nameserver nach aussen hin nicht sichtbar sein sollen und der übergeordnete Nameserver fragt diese deshalb selbst mit der rekursiven Methode.
Um das Verfahren weiterhin zu optimieren, speichern die Nameserver einmal angefragte Adressen in einem lokalen Cache, für den sie in einem vom angefragten Nameserver vorgegebenen Zeitraum gespeichert werden.
Da es häufig Anfragen an den Root-Nameserver gibt, sollte dieser möglichst jederzeit verfügbar sein. Daher existieren derzeit von diesem 13 Instanzen, die mehr oder weniger auf der Erde verteilt sind.
Die Auflösung von IP-Adressen zu Namen ist schwieriger zu realisieren, es gibt jedoch den Bereich `in-addr.arpa` im Namespace, der Adressen als Label besitzt, beispielsweise `152.192.16.15.in-addr.arpa` für die IP-Adresse 15.16.192.152.

Resource Records (RR)

Die Einträge in den Zonendateien werden **Resource Records (RR)** genannt. Ihre Struktur ist (label, ttl, class, type, value). Das „label“ bezeichnet den Domänennamen, „ttl“ ist die bereits angesprochene Time-to-Life, welche die Gültigkeit der Resource in Sekunden angibt. Der Wert für „class“ bezeichnet die Protokollgruppe und beträgt fast immer „IN“ (Internet). Die wichtigsten Werte sind schließlich der Typ („type“) und sein zugehöriger Wert

(„value“) des Records. So gibt beispielsweise ein RR vom Typ „A“ die IP-Adresse eines Hosts an, mit „MX“ wird ein Mailserver der Domain angegeben, „NS“ gibt einen Nameserver für eine Zone an und „TXT“ speichert einen beliebigen Textkommentar. Ein Beispieleintrag ist „`pinkje.ripe.net 3600 IN A 193.0.1.162`“.
Zum Steuern eines Zonentransfers ist ein RR mit einem SOA-Eintrag (Start of Authority) für den authoritativen Server zwingend erforderlich. Hier wird der primäre Masterserver für diese Zone, eine E-Mail-Adresse des Verantwortlichen (mit „`.`“ anstelle des „`@`“), einer manuell gesetzten Seriennummer sowie Refresh-, Retry-, Expire- und Negativ-Caching-TTL-Zeiten gespeichert. Ein solcher Eintrag ist beispielsweise:
`bsp.de. 800 IN SOA master.bsp.de. hostmaster.bsp.de. 2007061501 3600 1800 604800 1800`

DNS Protokoll Header

Sowohl für Anfragen als auch für Antworten gibt es bei DNS nur ein Protokollformat. Zur Zuordnung von Anfragen und Antworten gibt es zunächst eine 16 Bit Identifikationsnummer. Dieser folgen 4 Bit mit Flags, die kennzeichnen, ob es sich um eine Anfrage/Antwort, authoritative/nicht authoritative Informationen, iterative/rekursive Anfragen handelt, sowie ein Bit, dass angibt, ob Rekursion möglich ist. Nun folgen 4 Felder, in denen jeweils die Anzal der darauffolgenden Felder angegeben ist. Diese Felder, die schließlich die relevanten Informationen enthalten, sind: (1) Anfragen von Namen, die aufgelöst werden sollen, (2) Antworten zu gestellten Anfragen, (3) Identifizierungen über die verwendeten verantwortlichen Nameserver und (4) zusätzliche Daten zu der Anfrage wie z. B. der RR des korrekten Namens eines Aliases.

4.3 Das WWW und E-Mail

[. . .]