

Einleitung und Überblick über das Internet

Rechnernetze II **(Internet-Dienste und -Anwendungen)**

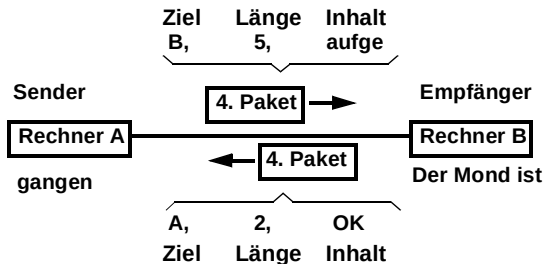
Das Internet

- Internet = sehr viele miteinander gekoppelter Teilnetze, die zu einem weltumspannenden Netz verbunden sind
- Teilnetze (sog. Intranets) gehören Firmen, Organisationen Behörden, Universitäten, ... , sowie den Telekom-Unternehmen der Welt
- Intranets sind im inneren Aufbau und den verwendeten Protokollen oft zueinander inkompatibel
- Inkompatible Intranets werden durch sog. Gateway-Rechner und dem IP-Protokoll gekoppelt (IP = internet protocol)
- In Deutschland: >30 Millionen Rechner mit Internetzugang, in Europa: >150 Millionen Rechner, weltweit: >500 Millionen Rechner (jeweils geschätzt)
- Ursprünglich wurde das Internet in den späten 1960er-Jahren für das US Department of Defense zur sicheren Kommunikation zwischen Atombunkern entwickelt
- Es wurde seit den frühen 1970er-Jahren von amerikanischen Behörden, Universitäten und Forschungseinrichtungen für verschiedene Dienste genutzt:
 - Zugriff auf entfernte Dateien (FTP)
 - Login auf entfernte Rechner (Remote Login, Telnet)
 - Elektronische Post (email)
- Heutzutage wird das Internet hauptsächlich durch Dienste wie NEWS oder www sowie email dominiert

- Alle Internet-Dienste und -Anwendungen beruhen auf Datenübertragung in Rechnernetzen
- Die Datenübertragung beruht auf Nachrichtenaustausch mittels Paketen
- Längere Nachrichten werden in kleinere Pakete zerlegt
- Beispielnachricht:

Der	Mond	ist	aufge	gangen
1. Paket	2. Paket	3. Paket	4. Paket	5. Paket

- Beispiel-Nachrichtenaustausch:

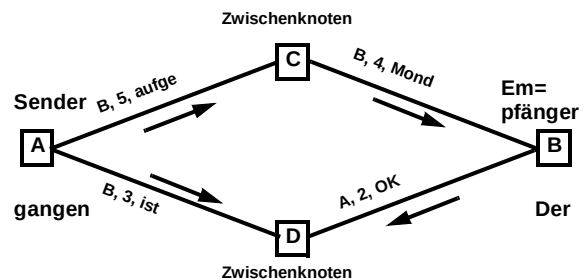


- Pakete können verfälscht werden
- => Verwendung von Prüfsummen (CRC)
- Im Fehlerfall können Pakete wiederholt werden
 - Pakete können verloren gehen:

=> Für jedes empfangene Paket, das zusätzlich korrekt ist, wird eine Quittung „OK“ zurückgeschickt

Wegwahl (Routing) in Rechnernetzen

- Der Datentransport verläuft über Zwischenknoten:

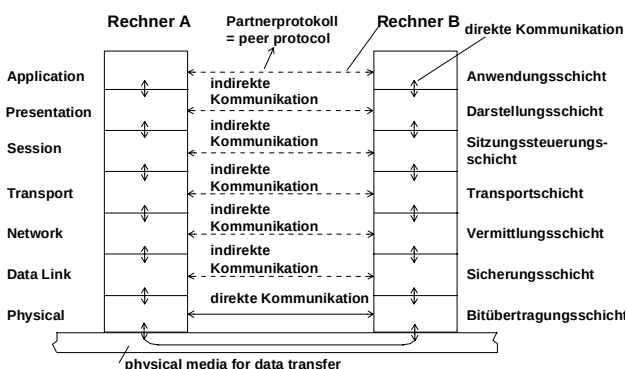


- Die Wegwahl zwischen den Knoten bzw. Intranets erfolgt über das IP-Protokoll (IP = Internet Protokoll)
 - Pakete werden als autonome Einheiten versendet
 - Zwischenknoten müssen Pakete zwischenspeichern ("store-and-forward"-Prinzip)
 - Pakete gehen unterschiedliche Wege
- => Pakete können sich überholen
- => Empfänger muss Paketreihenfolge wiederherstellen

- Sender muss Pakete im für den Empfänger richtigen Tempo senden => Flusssteuerung

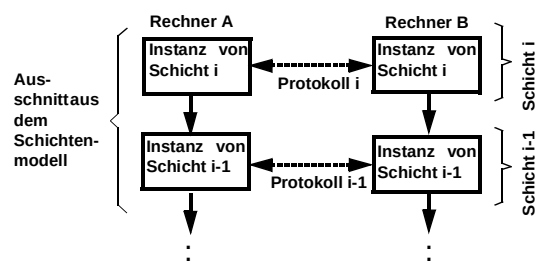
Das ISO-7-Schichten-Modell

Die ISO (International Standardization Organization) hat die Aufgaben, die beim Transport von Daten zwischen Rechnern anfallen, in einem Schichtenmodell gegliedert



- Jede Schicht erbringt einen speziellen Dienst; d.h., sie bietet an ihrer Schnittstelle nach oben einen wohldefinierten Funktionsumfang an
- Die Implementierung einer Schicht heisst Instanz

- Die Instanz einer Schicht auf einem Rechner kommuniziert nur mit einer Instanz derselben Schichthöhe
- Instanzen gleicher Höhe auf verschiedenen Rechnern heissen peers (Partnerinstanzen)
- Die Kommunikationsregeln zwischen peers werden durch ein Protokoll festgelegt
- Für die Abwicklung des Protokolls greift jede Instanz auf die Funktionen der unmittelbar darunterliegenden Schicht zurück (=> 7-stufige Rekursion)



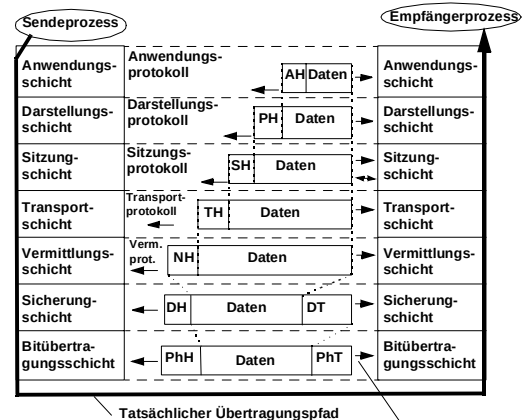
- Nur auf Ebene 1 werden tatsächlich Bits übertragen
- Die Ebenen 2-7 haben andere Aufgaben

Indirekte Kommunikation im ISO-Referenzmodell/Partnerprotokolle

- Daten werden innerhalb eines Rechners von Schicht zu Schicht transportiert
- Zwischen 2 Rechnern können Daten nur auf der Schicht 1 transportiert werden
- Kommunikation zwischen Instanzen höher als Schicht 1 werden indirekt (virtuell) abgewickelt
- Die indirekte Kommunikation wird mittels eines Protokolls zwischen Partnerinstanzen durchgeführt, so dass beide Partnerinstanzen „sich verstehen“ können (Protokoll = gemeinsame Sprache)

Aufbau eines Nachrichtenpakets im Netz

- Von Schicht zu Schicht kommt ein Paket-Header und evtl. ein Paket-Trailer hinzu
- Einige der Nachrichtenköpfe (Header) bzw. Trailer können im Allgemeinfall auch leer sein
- Wird die zu übertragende Information zu lang, wird sie segmentiert

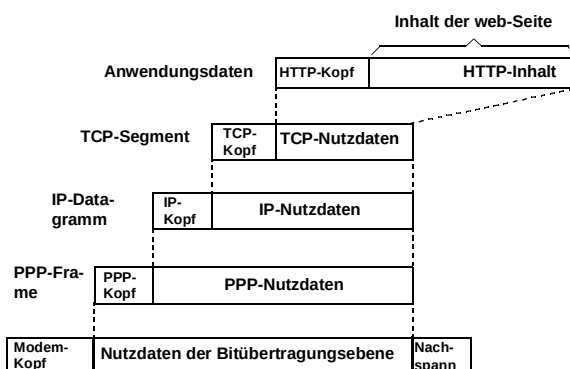


AH = application header
 PH = presentation header
 SH = session header
 TH = transport header
 NH = network header
 DH, DT = data link header/trailer
 PhH, PhT = physical link header/trailer

tatsächlich übertragene Bits

Beispiel: Surfen im Internet über Modem

- Verpackung einer web-Seite



PPP = Point-to-Point Protocol = Protokoll der Sicherungsschicht
 IP = Internet Protocol = Protokoll der Vermittlungsschicht
 TCP = Transmission Control Protocol = Protokoll der Transportschicht
 HTTP = Hypertext Transport Protocol = Protokoll der Anwendungsschicht

Die einzelnen ISO-Schichten

Bitübertragungsschicht

- Übertragungsmedien:
 - elektrische Kabel (Zweidrahtleitung, Koaxialkabel, ...)
 - elektromagnetische Wellen (Richtfunk, Satelliten, ...)
 - Glasfaser (Monomode-Faser, Multimode-Faser)
- Art der Übertragung:
 - synchron oder asynchron zu einem Taktsignal
 - bitseriell oder bitparallel
 - unkodiert (rein binär) oder kodiert (Manchester-Code, differentieller Manchester-Code, ...)
- Übertragungsverfahren:
 - nicht moduliert (= „Basisband“)
 - moduliert (Frequenz, Amplitude, Phase)
 - nicht gemultiplext
 - gemultiplext (Zeitmultiplex, Frequenzmultiplex)

Datenraten verschiedener Netz-Technologien

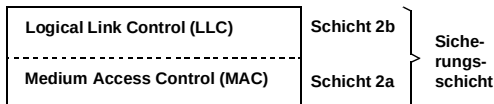
- Bei lokalen Netzen - Local Area Networks (LANs)
 - Ethernet 10/100/1000 Mbit/s
 - Tokenring 4/16 Mbit/s
 - FDDI 100 Mbit/s
 - Funk-LANs 0.2..40 Mbit/s
- Bei Fernnetzen - Wide Area Networks (WANs)
 - Telefonnetz 56 kbit/s (über Modem)
 - ISDN 64 kbit/s bzw. 128 kbit/s

- xDSL 1,44 .. 10 Mbit/s
- Mobilfunk 64 kbit/s ... 2 Mbit/s
- Mietleitung einer Telekom „E1“ mit 2048 kbit/s
- ATM 34 ... 622 Mbit/s ... 2,4 Gbit/s

ATM = Asynchronous Transfer Mode
DSL = Digital Subscriber Line

Sicherungsschicht (ISO-Schicht 2)

- Ist meist unterteilt in zwei Unterschichten, z.B. bei den IEEE 802.x-Normen
- IEEE = Institute of Electrical and Electronics Engineers = Berufsverband, sehr aktiv im Bereich der Normung



- Die LLC-Schicht ist für die eigentlichen ISO-Schicht-2-Aufgaben der Sicherung zuständig
- Die MAC-Schicht regelt den Zugang mehrerer Rechner zu einem gemeinsam genutzten Übertragungsmedium (Kabel, Funk)

Unterschicht LLC der Sicherungsschicht

- Aufgaben von LLC (logical link control):
- 1.) Portionierung des Bitstroms in "Häppchen" (Frames) = Portionierung auf der Schicht 2
- 2.) Anhängen einer Prüfsumme an die Daten (CRC, Cyclic Redundancy Check)
- 3.) Erneute Übertragung bei Fehler oder Verlust eines Pakets zwischen je zwei benachbarten Knoten
- 4.) Senden eines Quittungspakets durch den empfangenden Nachbarknoten, um fehlerhafte oder verlorene Pakete beim Sender anzuzeigen
- 5.) Timeout zum Erkennen von Verlust von Quittungen zwischen benachbarten Knoten
- 6.) Flußsteuerung zwischen benachbarten Knoten = Bremsen des Senders zur Verhinderung von Pufferüberlauf beim Empfänger
- 7.) Vollduplex- oder Halbduplexbetrieb auf der Schicht 2 (Vollduplex = Sender und Empfänger können gleichzeitig senden)
- Beispiel Ethernet: keine Neuübertragung, kein Timeout, keine Flußsteuerung, nur Halbduplex

Unterschicht MAC der Sicherungsschicht (Medium Access Control = Medienzugang)

1.) Kollisionsbehaftete Protokolle

- Jeder Rechner überträgt, ohne sich mit anderen abzusprechen (Bsp. Ethernet auf gemeinsamem Kabel)
- Dadurch kommt es zu Kollisionen (= Paketverluste durch gleichzeitige Übertragung verschiedener Pakete)
- Reduzierung der Paketverluste durch:
 - Abhören des Mediums vor dem Senden, ob Medium frei ist
 - Zufallsgesteuerte Zeiterzögerung vor erneutem Paketsenden nach Auftreten einer Kollision
 - Prinzip heisst CSMA/CD (carrier sense for multiple access with collision detection)

2.) Kollisionsfreie Protokolle

- Es gibt eine „Sendeberechtigungsmarke“ (Token), die in einem ringförmigen Netz von Sender zu Sender weitergereicht wird
- Nur wer das Token besitzt, darf senden
- Da es nur ein Token gibt, ist der exklusive Zugriff gesichert
- Nach dem Senden eines Paketes muß das Token zum nächsten Nachbarn weitergereicht werden
- Beispiel: Token-Ring (4 bzw. 16 Mbps), FDDI (Fiber Distributed Data Interface, 100 Mbps)

Vermittlungsschicht (ISO-Schicht 3)

- Aufgaben:

1.) Unterscheiden der Verbindungsart

- Verbindungslos (Datagramm)
 - Nachrichtenpakete enthalten Ziel- und Quelladresse
 - Sie werden unabhängig voneinander durch das Netz geschickt, d.h. sie gehen unterschiedliche Wege
- Verbindungsorientiert
 - Feste Verbindung vom Sender zum Empfänger wird aufgebaut, Pakete gehen stets denselben Weg

2.) Wegewahl (Routing)

- Wegewahl ist adaptiv/nicht adaptiv (adaptiv heisst, dass die Verkehrsbelastung der Zwischenknoten berücksichtigt und danach die Route ausgewählt wird)
- Verbindung ist verklemmungsfrei oder nicht verklemmungsfrei (Deadlocks)
- Verbindung nutzt gleichzeitig mehrere Pfade durch das Netz oder nicht (dient zur Bandbreiteerhöhung und Fehlertoleranz)
- 3.) Verbindung von Netzen verschiedener Betreiber (Inter-networking z.B. durch das IP-Protokoll)

- Das Schicht-3-Protokoll im Internet ist IP (internet protocol)

- 4.) Stauregelung zur Vermeidung von überlasteten Zwischenknoten (gibt es nicht im Internet)

Transportschicht (ISO-Schicht 4)

- Aufgaben:

- Bei verbindungsorientiertem Datenaustausch wird eine feste Verbindung zwischen Sender und Empfänger über Zwischenknoten quer durch das Netz auf- und wieder abgebaut
- Grosse Datenmengen werden in kleinere Portionen zerteilt (Segmentierung auf Schicht 4)
- Wiederherstellen der Paketreihenfolge bei Datagrammen
- Erneute Übertragung eines Pakets bei Fehler oder Verlust, nicht zwischen Nachbarknoten, sondern über alle Zwischenknoten quer durch das Netz
- Flusssteuerung zwischen Sender und Empfänger quer durch das ganze Netz
- Multiplexen bei verbindungsorientiertem Datenaustausch (Multiplexen = Mehrfachnutzung ein- und derselben Verbindung, um Bandbreite effizient zu nützen)
- Beispiele für Transportprotokolle im Internet: TCP (transmission control protocol) und UDP (user datagram protocol)

Sitzungsschicht (ISO-Schicht 5)

- Synchronisierung bei Client/Server-Kommunikation bzgl.:
 - Beginn einer „Sitzung“ (login) und Ende (logout)
 - half duplex/full duplex auf Schicht 5
- Automatisches Wiederaufsetzen einer gestörten Verbindung an der Stelle der Unterbrechung

- Beispiele Anwendungen: Video-on-Demand, web-TV, Real-Audio, ecommerce, NFS (SUN network file system)

Klassifizierung nach Dienste und Anwendungen

- **Definition Dienste:** Programme und ihre Protokolle, die allgemein im Internet verfügbar sind und deren Datenaustauschformat in einer Norm oder einer sog. RFC festgelegt ist.

- RFC = Request for Comment; siehe dazu <http://www.rfc.net>

- Beispiel: email ist über die RFC 822 festgelegt

- **Definition Anwendungen:** individuelle Programme von Firmen, Institutionen oder Einzelpersonen, die aufgrund ihrer Client/Server-Struktur ein Rechnernetz brauchen und die entweder proprietär oder für einen begrenzten öffentlichen Benutzerkreis geschrieben sind.

- Weitere Beispiele für Dienste: news, Zeitverteilung NTP, Ressourcenreservierung RSVP

- Weitere Beispiele für Anwendungen: Suchmaschinen, airline reservation systems (Flugbuchungen), Video-Konferenzen, gemeinsame Dokumentenbearbeitung (CSCW, computer supported work)

- Selbst geschriebene Programme, die ein Rechnernetz benutzen, sind zunächst einmal Anwendungen.

- Schicht 5 gibt es nicht im Internet

- Schicht 5-Funktionalität wird in Einzelfällen von Schicht 7-Anwendungen nachgebildet

Darstellungsschicht (ISO-Schicht 6)

- Aufgaben:

- Anpassung der unterschiedlichen Datenkodierungen und -Formate zwischen den Rechnern, z.B. EBCDIC <-> ASCII, ASCII <-> ISO-Code, <CR><LF> in <LF><CR>, u.s.w.
- Anpassung der unterschiedlichen Zahlenrepräsentationen zwischen den Rechnern, z.B. 32-Bit-Integer in 64-Bit-Integer u.s.w.
- Datenverschlüsselung
- Datenkompression

- Schicht 6 gibt es nicht im Internet. Schicht 6-Funktionalität wird in Einzelfällen von Schicht 7-Anwendungen nachgebildet

- => Schicht 7 kann im Internet nur auf den Funktionen der Transportschicht (Schicht 4) aufsetzen.

Anwendungsschicht (ISO-Schicht 7)

- Hier sind alle Internet-Anwendungen sowie die meisten Internet-Dienste lokalisiert

- Beispiele Dienste: www (world wide web), ftp, rlogin, email, Namensdienst DNS

- Sie verwenden im Internet Funktionen der Schicht 4 (Transportschicht).

- Ist eine Internet-Anwendung weit verbreitet und von allgemeinem Interesse, kann im Laufe der Zeit eine öffentliche RFC-Definition daraus entstehen.

- Aus der Anwendung wird dann ein Internet-Dienst. Bsp.: NFS von SUN.

Liste von Protokollen für Dienste

- FTP = file transfer protocol: zum Dateitransfer
- TELNET = „teletype networking“: für Einloggen auf entferntem Rechner mittels zeichenorientierten Terminals
- SMTP = simple mail transfer protocol: für email
- MIME = multipurpose internet mail extensions: für email-Erweiterungen
- HTTP = hypertext transport protocol: zum Übertragen von Texten und Graphiken mit Link-Verknüpfungen
- NSP = name server protocol: zur Kommunikation zwischen einzelnen sog. Name Server-Rechern
- NTP = network time protocol: zur Verteilung der Uhrzeit
- NNTP = network news transport protocol. Dient zur Implementierung von USENET.
- RSVP = resource reservation protocol: zur Reservierung von Bandbreite und Pufferplatz auf einem Server-Rechner; mit RSVP kann eine sog. Quality of Service (Garantie einer bestimmten Qualität bei der Übertragung) erreicht werden

- **SNMP** = simple network management protocol: dient zur Verwaltung von Netzwerk-Komponenten wie Router oder Switches; basiert auf UDP
- **DNS** = domain name system: dient zur Abbildung von symbolischen Rechnernamen auf IP-Adressen; basiert auf UDP
- **NFS** = network file system: dient zum Aufbau eines netzwerkweiten Dateisystems; wurde ursprünglich von SUN für UNIX-Rechner entwickelt; basiert auf UDP
- **ping**: dient zum Testen, ob ein bestimmter Rechner am Netz hängt; basiert auf ICMP
- Alle diese Dienste gehören zur Schicht 7
- Daneben gibt es noch verschiedene Steuerungsprotokolle, die der Vermittlungsschicht zugerechnet werden
- Ohne diese Steuerungsprotokolle würde das Internet nicht funktionieren
- Beispiele von Steuerungsprotokollen:
- **IP** = internet protocol: für die Wegewahl im Internet
- **ICMP** = internet control message protocol: dient zur Kommunikation zwischen Routern
- **ARP** = address resolution protocol: dient zur Kommunikation zwischen Name Server und Rechnern, um die Abbildung IP-Adresse -> MAC-Adresse vorzunehmen
- **RARP** = reverse ARP: dient zur Abbildung MAC-Adresse-> IP-Adresse

Verzeichnisdienste (synonym: Namensdienste)

- Beispiele für Verzeichnisdienste: Domain Name System (DNS) im Internet, sowie X.500 in ISO-7-Schichten-Netzen
- Verzeichnisdienste bilden die Basis vieler Internet-Dienste und -Anwendungen
- Ein Namensdienst leistet für ein Rechnernetz ähnliches wie ein Telefonbuch für das Telefonnetz: Die Abbildung von Rechnernamen auf Netzadressen
- Weitere Aufgaben von Namensdiensten wie z.B. DNS:
 - inverse Abbildung von Netzadressen auf Rechnernamen
 - korrekte Zustellung von email
 - Auskünfte über die lokal verfügbaren Rechner
- Namensdienste sind als weltweit verteilte Datenbank implementiert
- Die Abfrage und das Management der Datenbank erfolgt dezentral an vielen Stellen gleichzeitig
- Zur Zuverlässigkeitssteigerung wird manchmal eine Verdopplung oder Verdreifachung der jeweiligen lokalen Datenbank-Server vorgenommen
- Zur Leistungssteigerung werden frühere Abfrageergebnisse in lokalen Caches gespeichert

- **BOOTP** = boot protocol: dient zum Herunterladen eines Betriebssystems von einem Server auf einen sog. discless client
- Schliesslich existieren im Internet auch Protokolle auf der Sicherungsschicht, die teilweise in Hardware (Schicht 2a), teilweise in Software (Schicht 2b) realisiert sind
- Beispiele von Schicht 2b-Protokollen:
- **PPP** = point-to-point protocol sowie **SLIP** = subscriber line interface: dient zur gesicherten Datenübertragung, wenn Modems als Schicht 1 verwendet werden
- **PPPoE** = PPP over Ethernet: dient zur gesicherten Datenübertragung, falls Ethernet als Schicht 1 und 2a eingesetzt wird

Internet und ISO-7-Schichten-Modell

- Im Internet gibt es keine Schicht 5 und 6

Schicht 7	Anwendung, z.B.: www, ftp, email
Schicht 4	Transport: TCP, UDP
Schicht 3	Vermittlung: IP
Schicht 2	Sicherung, z.B.: SLIP, PPP, PPPoE, Ethernet, FDDI
Schicht 1	Bitübertragung, z.B.: RS 232, V24, Ethernet, FDDI

Domain Name System (DNS)

- Im Internet werden alle Rechner über ihre IP-Adresse angesprochen
- Jeder Rechner hat eine weltweit eindeutige 4-Byte IP-Adresse (=32 Bit => 2^{32} verschiedene IP-Adressen)
- Bsp.: 198.41.0.4 = IP-Adresse des .edu-Name Servers
- Jeder Rechnername ist ebenfalls weltweit eindeutig
- Wegen der Vielzahl existierender Rechner kann keine zentrale Instanz alle IP-Adressen und Namen verwalten
- **Problem:** Wie kann man, ohne globalen Überblick, erreichen, dass es keine 2 gleichen Adressen od. Namen gibt?
- **Lösung:** die Vergabe von Namen und Adressen erfolgt modular und hierarchisch
- Das graphische Äquivalent dazu ist ein Baum
- Die Hierarchieebenen im Baum entsprechen der hierarchischen Namensvergabe
- Die dezentralen, d.h. modularen Stellen der Namensvergabe entsprechen den Baumknoten
- Die eindeutigen Namen/Adressen entsprechen den Baumblättern

=> Jeder Knoten, der nicht Blatt ist, ist für einen ganzen Bereich von Namen zuständig, d.h. für seinen Unterbaum

- Der Name für dieses System lautet Domain Name System (Domain = Bereich)

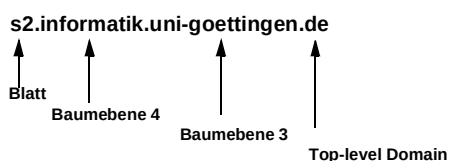
- Eigenschaften des Namensbaums:

- Die Baumtiefe ist „unbeschränkt“
- Die Zahl der Söhne pro Vaterknoten „unbeschränkt“
- Die Gesamtzahl der Blätter ist max. 2^{32} wg. 32-Bit-IP-Adresse
- Die zweitoberste Baumebene wird als Top-Level Domain bezeichnet und hat einen festgelegten Aufbau

- Jeder Name/Adresse entsteht aus einem Baumdurchlauf von einem Blatt bis zur Top-Level Domain

- Der Name ist die „Summe“ der durchlaufenen Knotennamen, die durch Punkt verbunden sind

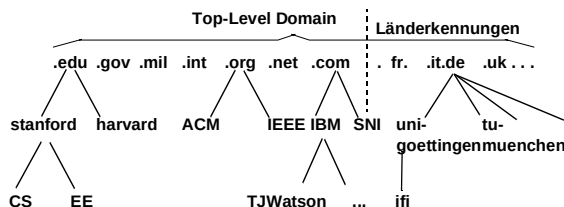
- Bsp.:



- Wichtige Knotennamen der Top-Level Domain sind:

- .int für weltweite Organisationen (UNO, ...)
- .org für internationale Organisationen (IEEE, ACM, ...)
- .mil für das US-Militär
- .gov. für die US-Regierung
- .edu für US-Universitäten
- .net für spezielle Internet-Adressen
- .com für Firmen (weltweit)
- Länderkennungen:
 - .de für Deutschland
 - .fr für Frankreich
 - .it für Italien u.s.w.
 - .at für Österreich
 - .ch für Schweiz
 - .uk für Großbritannien
 - u.s.w.

Der DNS-Namensbaum



- Jeder vollständige DNS-Name besteht aus dem eigentlichen Rechnernamen sowie einer Folge von Domain-Namen (Bsp.: ifi.uni-goettingen.de)

- Um eine neue Domain anzulegen, braucht man die Genehmigung des übergeordneten Vaterknotens im Namensbaum (= übergeordnete Domäne) und muss diese registrieren lassen

- Die Registrierung erfolgt kostenpflichtig entweder bei einem sog. Network Information Center (NIC) oder bei speziellen Firmen, die damit Geld verdienen

- Der Preis für die Registrierung beträgt grössenordnungsmässig einige Euro bis Dutzend Euro pro Jahr

- Für .de ist das NIC www.denic.de, für .com, .net, .org ist das NIC www.networksolutions.com

- Bemerkung: Inzwischen sind die 2^{32} -IP-Adressen stellenweise knapp geworden. Deshalb werden bei IP Version 4 bei Einwahl ins Internet mittels eines Providers IP-Adressen dynamisch vergeben.

- Dadurch entsteht die Situation, dass mehrere DNS-Namen dieselbe IP-Adresse haben können, allerdings kann sich von diesen zu einer Zeit immer nur einer unter dieser IP-Adresse ins Internet einwählen.

- Wollen zwei Rechner derselben IP-Adresse sich gleichzeitig einwählen, bekommen beide vom Provider verschiedene IP-Adressen zugewiesen.

Realisierung des DNS

- Jeder Knoten im Namensbaum, der nicht Blatt ist, ist ein sog. Name Server-Rechner und sorgt für seine Sohnknoten

- Die Sohnknoten sind entweder weitere Name Server oder im Fall von Blättern Benutzerrechner

- Die Informationen zur Abbildung von Namen auf IP-Adressen ist in Teilen in jedem Name Server-Rechner in Form von sog. Ressource Records gespeichert

- Die Ressource Records sind aus Sicherheitsgründen oft mehrfach vorhanden (als Originale und als Kopien)

Resource Records

- Resource Records sind lokale Datensätze, die es auf jedem Name Server gibt, und in denen die jeweils notwendigen Adreßabbildungen zur Namens“auflösung“ stehen.

- vereinfachtes Bsp. für 2 Resource Records:

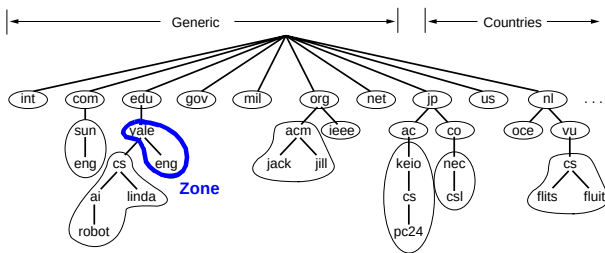
- lokaler Rechnername1 -> IP-Adresse1

- lokaler Rechnername2 -> IP-Adresse2

Zonen

- Soll ein Name Server eine Domäne verwalten, die kleiner ist als ein Unterbaum, also nur einige, aber nicht alle Sohnknoten umfaßt, dann teilt man die Sohnknoten und deren Unterbäume in sog. Zonen ein.

- Beispiel:



=> Iterative Namensauflösung ist für den Client aufwendiger als für den Server, rekursive Auflösung ist umgekehrt

- An der Yale-Universität gibt es einen Namens-Server xxx.yale.edu, der yyy.eng.yale.edu aber nicht zzz.cs.yale.edu verwaltet

Namens“auflösung“

- Es gibt es 2 Methoden zur Adreßabbildung:

1.) rekursive Auflösung

2.) iterative Auflösung

- rekursiv heißt, daß der Namensbaum im schlimmsten Fall hoch bis zur Wurzel und wieder abwärts bis dem Server durchlaufen wird, zu dessen Zone der gesuchte Rechner gehört.

- wird vor Ende des kompletten Baumdurchlaufs ein Server erreicht, der die Namensauflösung vornehmen kann, wird der Durchlauf an diesem Knoten beendet

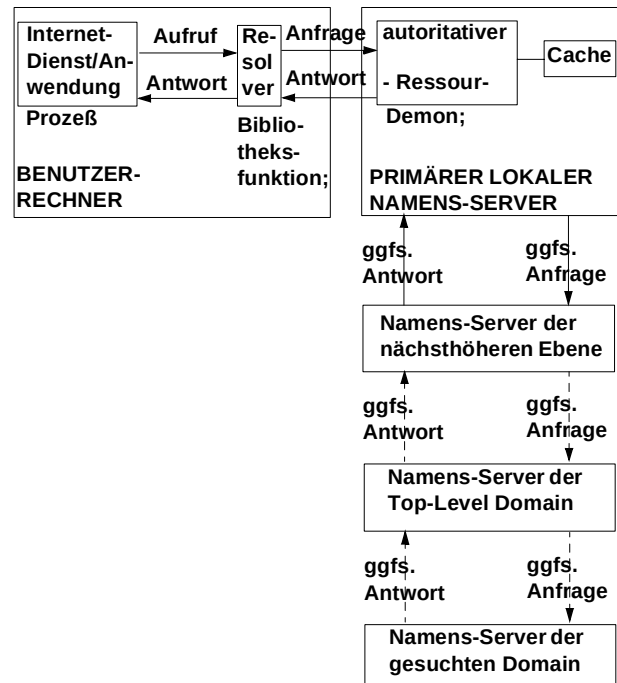
- Für die Durchführung des Baumdurchlaufs ist der lokale Name Server verantwortlich, nicht der anfragende Client.

- Bei der iterativen Namensauflösung hingegen muß der Client selbst einen Server finden, der die Adreßauflösung kennt.

- Er befragt dazu zuerst den lokalen Name-Server und dann, wenn nötig, die jeweils übergeordneten Name-Server.

- Iterative Namensauflösung wird dann eingesetzt, wenn der lokale Name-Server die rekursive Auflösung nicht kann.

Schema der rekursiven Namensauflösung



- Um Baumdurchläufe abzukürzen, hat sowohl der Resolver als auch der Name Server einen Cache, in dem die Ergebnisse früherer Anfragen gespeichert sind.

- Zur besseren Zuverlässigkeit existiert i.d.R. zu jedem primären auch ein sekundärer Namens-Server.
- Um Konsistenzprobleme zwischen primärem und sekundärem NS zu vermeiden, gibt es das Konzept des autoritativen Name Servers.

Autoritativer Namens-Server

= Derjenige Server, der für eine Zone maßgeblich verantwortlich ist. Sein Resource Record gilt. Alle anderen Server dieser Zone haben nur Kopien der autoritativen Ressourcensätze.

Aufgaben des autoritativen Namens-Servers

- 1. Abbildung Rechnername -> IP-Adresse
- 2. Inverse Abbildung IP-Adresse -> Rechnername
- 3. Korrekte Zustellung von email
- 4. Erteilen allgemeiner Auskünfte über die Rechner einer Zone (Prozessortyp, Betriebssystem)

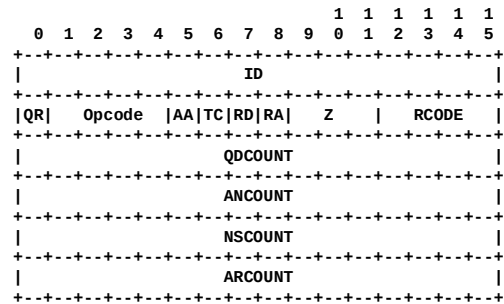
Anfrage an den Name Server (DNS-Query)

- Jede Anfrage wird über ein Protokoll zwischen Client und Name-Server abgewickelt.
- Das Protokolldatenpaket hat 5 Felder:
- 1. *Query Header*: enthält Steuerinformation zur Anfrage

- **OPCODE**: 0=Standardanfrage, 1=inverse Anfrage, 2=Server Status Request, 3-15 Reserved
- **AA (Authoritative Address)**: ist gesetzt, wenn es sich bei dem antwortenden Server um den autoritativen Server der RRs handelt. Für AA = 0 sind die Informationen aus dem Cache des Servers zu nehmen (=nur Kopien)
- **TC (TrunCation)**: die Antwort wurde in mehrere Pakete aufgeteilt
- **RD (Recursion Desired)**: der Client möchte eine rekursive Namensauflösung
- **RA (Recursion Available)**: gibt an, ob der Server rekursive Anfragen unterstützt (=1) oder nicht (=0)
- **Z**: reserviert - muß immer 0 sein
- **RCODE (Response Code)**: gibt an, ob eine Anfrage erfolgreich bearbeitet werden konnte, oder ob Fehler aufgetreten sind
- **QDCOUNT**: Anzahl der Einträge in der Question Section
- **ANCOUNT**: Anzahl der Einträge in der Answer Section
- **NSCOUNT**: Anzahl der Einträge in der Nameserver Section
- **ARCOUNT**: Anzahl der Einträge in der Additional Section

- 2. *Question*: Rechnername, dessen Adresse gesucht wird
- 3. *Answer*: Resource Records als Antwort auf eine Anfrage
- 4. *Authority*: Resource Records, die den Weg zu einem autoritativen Server angeben
- 5. *Additional*: zusätzliche Resource Records, die der Client in Zukunft vielleicht brauchen wird, z.B. die IP-Adresse des Mail Servers der Zone

Aufbau des Query Headers (=1. Feld):



- **ID**: eine 16-Bit Nummer, anhand derer der Client eine empfangene Antwort einer zuvor gestellten Anfrage zuordnen kann
- **QR**: 0 für Query, 1 für Response

Dateien, die jeder Namens-Server braucht

- **Boot-Datei**: enthält Konfigurationsparameter für den Server und Angabe, welche Zonen der Name Server verwalten soll
- **Zonen-Dateien**:
 - Für jede Zone, für die der Server primärer Namens-Server ist, gibt es zwei Dateien für autoritative Ressourcensätze; Die eine Datei ist für die Abbildung Name->Adresse, die andere für die inverse Abbildung zuständig
 - Für jede Zone, für die der Server sekundärer Namens-Server ist, gibt es zwei Dateien, die als Caches für nicht-autoritative Ressourcensätze dienen
- **Root Datei**: enthält die Namen und IP-Adressen der Top-Level Domain Namens-Server (zweitoberste Ebene des Namensbaums)

Aufbau der Resource Records im Namens-Server

- Für jeden Rechner in einer Zone verwaltet der zuständige Name Server mehrere Resource Records (RRs)
- Jeder Ressourcensatz ist ein Fünftupel (hat fünf Felder):
- 1. **owner**: Name der Zone, zu der der RR gehört
- 2. **type**: 16-Bit-Wert, der den Typ des RRs angibt, z.B. Adreßtyp für die Adreßabbildung, Aliastyp für synonyme Namen, etc

- **3. class**: 16-Bit-Wert, der die Protokollfamilie angibt, für die der RR gültig ist. Hier wird nur "IN" für Internet verwendet, auch wenn es prinzipiell jedem Netz offen stünde, den DNS zu verwenden.
- **4. TTL** (Time To Live): 32-Bit-Wert, der die Lebensdauer des RRs in Sekunden angibt. Damit kann man festlegen, wie lange der RR im Cache von Resolvern stehen darf.
- **5. RDATA** (Resource Data): Hier werden in Abhängigkeit von *type* und *class* die eigentlichen Informationen gespeichert, beispielsweise:
 - die IP-Adresse, falls *type* = A (=Adreßtyp)
 - ein synonymer Name, falls *type* = CNAME (=Aliastyp)

Aufbau des Typ-Feldes

- Folgende Typen von Ressourcensätzen sind definiert:
 - 1) **A** (Adreßtyp): enthält Name und IP-Adresse eines Rechners. Dient zur Abbildung Rechnername -> IP-Adresse
 - 2) **CNAME** (Canonical NAME): definiert ein Synonym (Alias) für einen Rechner. Damit ist es möglich, unter zwei Namen auf einen Rechner zuzugreifen. Pro Rechner ist nur ein Alias zulässig.
 - Bemerkung: **CNAME** Mail Aliases sind nicht mit DNS Aliases zu verwechseln. Bei DNS sind durch dynamische IP-Adressvergabe durch einen Provider beliebig viele Aliases für dieselbe IP-Adresse möglich.
- 7.3) **retry**: Intervall, nach dem der sekundäre NS den primären NS nach einem Verbindungsfehler wieder zu kontaktieren versucht (600s)
- 7.4) **expire**: Nach dieser Zeitspanne verwirft der sekundäre NS die Zone, wenn er bis dahin keine Verbindung zum primären NS bekommen hat (3600000, ca. 42 Tage)
- 7.5) **minimum**: Das Minimum der Lebensdauer (Time to Live, TTL) für die RRs der Zone
- 7.6) **sonstige Parameter**: Refresh Time, Cache Time, Seriennummer des Zonefiles, . . .

- 3) **HINFO** (Host Info): gibt den Prozessortyp und das Betriebssystem eines Rechners der Zone an
- 4) **MX** (Mail Exchanger): gibt den Server an, der in der Zone für emails zuständig ist. Dort landet alle Post, die an den Rechner geschickt wird.
- Durch die zusätzliche Angabe einer Priorität kann Ausfallsicherheit erreicht werden, sofern in einer Zone mehrere Mail Server existieren.
- 5) **NS** (Nameserver): gibt den autoritativen Name Server für die Domain an
- 6) **PTR** (Pointer): Zeiger auf einen Rechner. Dient u.a. zur inv. Abbildung IP-Adresse -> Rechnername
- 7) **SOA** (Start Of Authority): zeigt den Beginn von zusätzlicher Verwaltungsinformation für die Zone an
- Hier werden insbesondere folgende Werte gespeichert:
 - (In Klammern die Default-Werte)
 - 7.1) **serial**: Versionsnummer der Zone, die dem sekundären Name Server anzeigt, ob sich an den RRs des primären Name Servers etwas geändert hat
 - 7.2) **refresh**: Zeitintervall, nach dem der sekundäre NS die RRs des primären Name Servers auf Änderungen überprüft (3600s)

Beispiel einer Anfrage (DNS Query):

```
jonny@pluto:~ > dig www.uni-ulm.de
; <<>> DiG 8.1 <<>> www.uni-ulm.de
;; res options: init recurs defnam dnsrc
;; got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 6
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 5, AD-
DITIONAL: 5
;; QUERY SECTION:
;;   www.uni-ulm.de, type = A, class = IN
;; ANSWER SECTION:
www.uni-ulm.de.      1h31m24s IN CNAME  orion-fddi.rz.uni-
ulm.de.
orion-fddi.rz.uni-ulm.de. 1d22h3m8s IN A  134.60.240.13
;; AUTHORITY SECTION:
rz.uni-ulm.de.      14h10m46s IN NS  titania.mathematik.uni-
ulm.de.
rz.uni-ulm.de.      14h10m46s IN NS  theophys.physik.uni-
ulm.de.
rz.uni-ulm.de.      14h10m46s IN NS  ns.informatik.uni-
ulm.de.

;; ADDITIONAL SECTION:
```

Telnet und FTP

TELNET

- TELNET vereint drei Konzepte:
- 1. Virtuelles Terminal
- 2. Verhandelbare Optionen
- 3. Symmetrische Verbindungen
- Virtuelles Terminal heißt, daß TELNET die Eigenschaften eines idealisierten Terminals auf die konkreten Hardware-Möglichkeiten des Client-Rechners abbildet (=Abstraktion der zugrundeliegenden Hardware).
- Vorteil: der Server braucht bei der Kommunikation keine Kenntnis über die Client-Hardware, sondern kann sich auf einen immer vorhandenen Satz von Funktionen berufen.
- Verhandelbare Optionen: Der Server handelt mit dem Client Parameter der Kommunikation aus, z.B. zusätzliche Terminaleigenschaften.
- Symmetrische Verbindungen: Client und Server sind aus der Sicht des Protokolls gleichberechtigt. TELNET erlaubt daher auch Interprozessorkommunikation über Botschaftenaustausch.

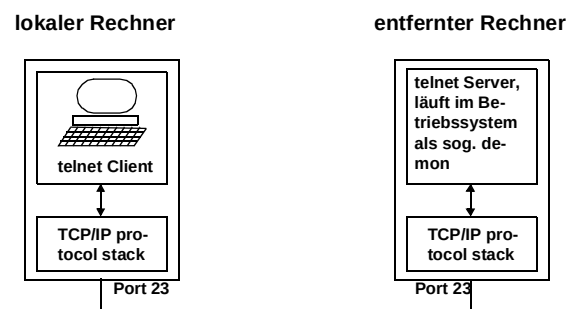
Einsatzgebiete von TELNET

- 1. *Remote login* = Arbeiten an anderem Rechner über alphanumerisches Terminal
- 2. *Linking* = Kopplung zweier TELNET-Terminals zum Chat

- 3. *Distributed Computing* = Kopplung zweier Anwendungsprozesse über das TELNET-Protokoll zum Botschaftenaustausch

Remote Login bei telnet

- Emuliert ein virtuelles, alphanumerisches Terminal auf dem lokalen Rechner, das Verbindung zum entfernten Rechner hat
- Man kann damit lokal so arbeiten, als ob man vor dem entfernten Rechner sitzen würde, allerdings ohne Graphik- und Mausunterstützung
- Benützt TCP als Transport- und IP als Wegewahlschicht
- Basiert auf dem Client/Server-Konzept



- Verwendet die sog. Port-Adressierung für den Zugriff auf den entfernten Rechner (Port 23)

- Ports sind feste Eintrittspunkte für die Kommunikation mit Prozessen, die auf anderen Rechnern laufen

FTP

- ftp erlaubt das Senden, Empfangen, Löschen und Umbenennen von Dateien, sowie das Einrichten und Löschen von Verzeichnissen und den Wechsel des aktuellen Verzeichnisses auf dem Entfernten Rechner
- Die Dateiübertragung kann in binärem- oder ASCII- Modus erfolgen
- Im binären Modus ("image file type") wird der Bitstrom aus dem Speicher des Senders ausgelesen und unverändert übertragen
- Im ASCII-Modus werden alphanumerische Zeichen gemäß folgenden Regeln übertragen:
 - Ein Zeilenende wird als <CR><LF> dargestellt; wenn der Empfänger eine andere lokale Darstellung hat (z.B. <LF><CR>), wird diese gewählt
 - Wenn der Empfänger einen anderen Code als ASCII hat (z.B. ISO oder EBCDIC), wird umcodiert
- => Der ASCII-Modus implementiert eine einfache Funktionalität der Darstellungsschicht
- Verwendet das Client/Server-Prinzip und die Port-Adressierung zur Kommunikation
- Server läuft als demon im Benutzeradressraum

- Als Antwort für jeden Befehl wird ein Statuscode zurückgeliefert
- Der Zugang zum entfernten Rechner erfolgt über Benutzerkennung und Passwort

Anonymous ftp

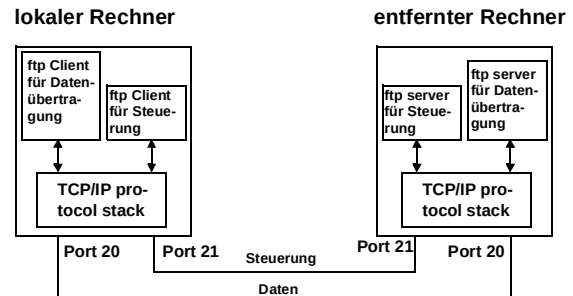
- Dient zum kostenlosen Verteilen von Software und/oder Dokumentation per ftp-download
- login-Kennzeichen: ftp oder anonymous
- Passwort: guest oder eigene email-Adresse
- Software ist gepackt, z.B. als tape archive (tar), Ziv-Lempel (zip), GNU zip (gz), archive (arc), u.s.w.

Email (Electronic Mail)

- Email dient zum Versenden von Textnachrichten, Graphiken, Bildern sowie Audio- und Video-Nachrichten.
- In den meisten Netzen von Firmen und Institutionen (Intra- und Extranets) gibt es email, z.T. mit inkompatiblen Adressen, Formaten, etc. Bsp.: X.400- und Internet-Adressen
- Die Versendung von emails zwischen inkompatiblen Netzen (z.B. Internet -> Compuserve oder T-Online) erfolgt über sog. Mail Gateways, die eine Formatkonversion auf Ebene 7 des ISO-Modells vornehmen (=Anwendungsebene)

=> Keine Integration des Dateisystems des entfernten Rechners in das lokale Dateisystem (ftp ist nicht „transparent“)

- Es werden 2 separate TCP-Verbindungen für Datenübertragung und Steuerung der Datenübertragung aufgebaut
- Der Grund liegt in erhöhter Handhabbarkeit: Abbruch einer gestörten Datenübertragung über den Steuerkanal



- Befehle für Verzeichnisse: cd, ls, rm, mkdir
- Befehle für Dateien: (m)put, (m)get, ...
- Befehl für Authentifizierung: user
- help-Funktionen

- Im Internet gibt es 2 Mail-Formate: RFC 822 und MIME

- Eine Email-Nachricht nach RFC 822 besteht aus

- einem Umschlag (envelope)
- einem Kopfteil (header)
- und der eigentlichen Nachricht (body)

- Typische Einträge im Kopf:

- To: Email-Adresse der Empfänger (können auch mehrere sein)
- Cc: Email-Adressen derjenigen, die eine Kopie der Nachricht erhalten (Carbon copy)
- Bcc: Wie Cc, nur daß dieses Feld bei den Empfängern in To: und Cc: leer bleibt (Blind CC)
- From: Adresse desjenigen, der die Nachricht erzeugt hat
- Sender: Adresse desjenigen, der die Nachricht tatsächlich verschickt hat
- Received: Je eine-Zeile für jeden Mail-Transfer-Server entlang der Route
- Date: Datum und Zeit des Absendens
- Subject: Kurze Zusammenfassung des Inhalts (eine Zeile)
- Reply-To: Email-Adresse, an die eventuelle Antworten geschickt werden sollen

- Email basiert auf miteinander kommunizierenden Programmen, den sog. Mail-Agenten

- Es gibt 2 Typen von Mail-Agenten:

- 1. Benutzeragenten (User Agents)

- 2. Nachrichtentransferagenten (Mail Transfer Agents)

Benutzeragent

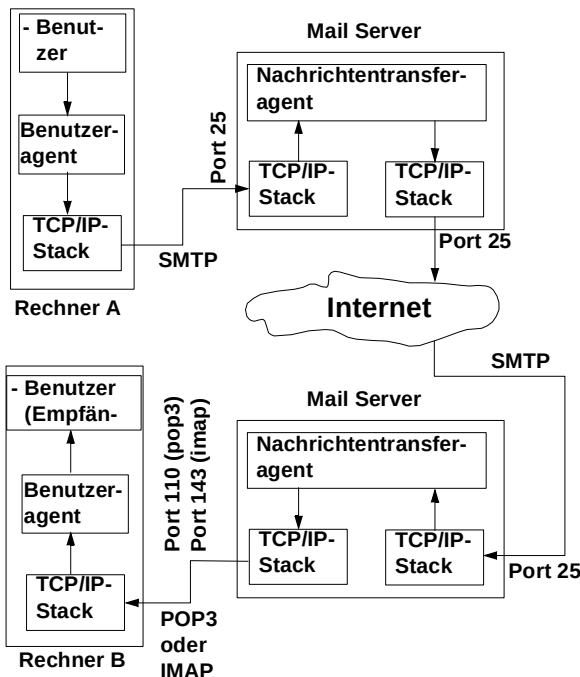
- Programm, mit dem Benutzer emails erstellen, lesen und speichern können. Bsp: Netscape, Eudora, Outlook
- Der Benutzeragent ist ein Client-Programm

Nachrichtentransferagent

- Programm im Mail-Server des Senders, das im Verbund mit dem Nachrichtentransferagenten des Mail-Servers des Empfängers dafür sorgt, daß emails durch das Internet transportiert werden
- Der Nachrichtentransferagent ist ein Server-Programm
- Der Benutzeragent ist auf dem Benutzerrechner angesiedelt, der Nachrichtentransferagent auf dem jeweiligen lokalen Mail Server
- Wenn in den Resource Records einer Internet-Adreßdomäne das MX-Feld definiert ist, gibt es in dieser Domäne bzw. Zone einen Mail Server, der die emails aller Rechner der Domäne verwaltet und auf dem ein Nachrichtentransferagent läuft.
- Der Mail Server besteht aus drei Teilen:

- 1.) Einem Briefkasten (Mailbox) für jede email-Adresse der Domäne (speichert ankommende und abgehende emails)

Zustellung einer email



Pop3-Protokoll (Post Office Protocol No. 3)

- Dient zum Lesen von emails auf lokalem Mail Server

- 2.) Ein Protokoll zur Kommunikation mit dem Benutzeragenten (

- 3.) Ein Protokoll zur Kommunikation mit dem Nachrichtentransferagenten des Mail Servers des Empfängers

- Der Mail Server kann mit dem 1. Router in der Zustellkette der Wegewahl (Routing) identisch sein, muß aber nicht

- Die Nachrichtentransferagenten, die für den sendenden bzw. den empfangenden Client verantwortlich sind, verwenden SMTP zur Kommunikation untereinander

- Der Benutzeragent kommuniziert mit seinem lokalen Mail Server über das POP3 oder IMAP-Protokoll

- Für das Senden von Mail vom Client zum lokalen Mail Server kann alternativ auch SMTP verwendet werden

- Alle 3 Protokolle verwenden TCP/IP für die sichere Übertragung und richtige Wegewahl

- Ebenso baut der Nachrichtentransferagent des Mail-Servers des Senders bzw. dessen SMTP-Software für den Transfer eine TCP-Verbindung zu dem Nachrichtentransferagenten des Mail-Servers des Empfängers auf

- SMTP verwendet TCP-Port 25 bei Sender und Empfänger

- Einfaches Protokoll, das nur ASCII-Zeichen überträgt

- erlaubt An- und Abmelden beim Mail Server, Herunterladen und Löschen von emails aus der Mailbox des Benutzers

IMAP (Interactive Mail Access Protocol)

- hat bedeutend mehr Möglichkeiten als POP3:

- Abfrage der Mail Box von mehreren Rechnern desselben Benutzers aus (Bsp.: Laptop, Festgerät, etc.)

- sortiert und filtert eingehende email nach Attributen (Bsp.: alle emails von Sender xy, alle emails mit Schlüsselwort abc)

SMTP (Simple Mail Transfer Protocol)

- baut eine direkte bidirektionale TCP-Verbindung vom Mail Server des Senders zum Mail Server des Empfängers auf

- sorgt für den Transfer von RFC 822- oder MIME-emails, nicht für deren Erzeugung, Empfang oder Speicherung

- wird zusätzlich für das Senden von email zwischen Client und Server verwendet

- hat die folgenden Kommandos zur Steuerung des email-Versands:

- HELO: zum Verbindungsaufbau

- MAIL: zur Angabe des Absenders der email

- RCPT: zur Angabe des Empfängers der email

- VRFY: zur Bestätigung der Empfängeradresse
- DATA: danach folgt die eigentliche email-Nachricht (body)
- <CR LF>.<CR LF>: Endezeichen des Nachrichtentextes
- QUIT: zum Verbindungsabbau
- Jedes Kommando muß vom Empfänger quittiert werden. Dazu dient das 2-Byte-Zeichen „OK“.
- Zum Aufbau von Verteilerlisten gibt es das EXPN-Kommando. (Verteilerliste = mehrere Adressaten derselben email)
- Als Daten werden 7-Bit-ASCII-Zeichen übertragen.
- Für emails, die Binärdaten enthalten, ist entweder eine Kodierung (z.B mit uuencode/uudecode) erforderlich oder die Verwendung von MIME-emails.
- Die SMTP-Software wird implementierungstechnisch als sog. System-Demon (=verborgener Systemprozeß) implementiert, da ununterbrochen TCP-Port 25 auf eingehende emails abgefragt werden muß.

Beispiel-Interaktion mit SMTP

- R: 220 Beta.GOV Simple Mail Transfer Service
- S: HELO Alpha.EDU
- R: 250 Beta.GOV
- S: MAIL FROM:<Smith@Slphs.EDU>
- R: 250 OK
- S: RCPT TO:<Green@Beta.GOV>
- R: 550 No such user here

MIME

(Multipurpose Internet Mail Extensions)

- Erweiterung von emails nach RFC 822 für verschiedene Arten von Nachrichten
- Überträgt Text, Graphiken, Bilder sowie Audio- und Video-Nachrichten (Multimedia-Übertragung möglich)
- Verwendet spezielle Codierungen für die Übertragung von Nicht-ASCII-Nachrichten: das sog. Base64 Encoding oder das Quoted Printable Encoding

Base64Encoding

- je 3 aufeinanderfolgende Bytes werden umgepackt zu 4 Zeichen a 6 Bit. Jedes der 6-Bit-Zeichen wird als ASCII übertragen.

Quoted Printable Encoding

- Alle 8-Bit-Bytes, die >127 sind, werden in eine Folge von 3 ASCII-Zeichen zerlegt. Das erste ASCII-Zeichen ist ein Gleichheitszeichen, gefolgt vom Hexadezimal-Code des zu übertragenden Bytes, der ziffernweise in ASCII codiert ist; Beispiel: =F6
- MIME verwendet das Konzept von mehrteiligen Nachrichten. Damit können Nachrichten unterschiedlichen Typs, z.B. einfache RFC 822-emails und Multimedia-emails in derselben MIME-email versendet werden.

- S: RCPT TO:<Brown@Beta.GOV>
- R: 250 OK
- S: DATA
- R: 354 Start mail input; end with <CR>LF>.<CR><LF>
- S: ...sends body of mail message...
- S: ...continues for as many lines as message contains
- S: <CR><LF>.<CR><LF>
- R: 250 OK
- S: QUIT
- R: 221 Beta.GOV Service closing transmission channel

FTP Mail Server

- Erlauben, Dateien per email automatisch zustellen zu lassen. Beispiel: Zustellung von Dokumentation per email, sobald eine email entsprechen Formats an den FTP Mail Server geschickt wird.

List Server

- verwalten Listen von email-Adressen. Alle Teilnehmer einer Liste werden automatisch über ein Kürzel angesprochen
- emails an den List Server erlauben das automatische An- und Abmelden sowie die Abfrage von Statusinformation

- MIME definiert 5 neue Felder zum alten RFC 822-Header:

Header	Meaning
MIME-Version:	Identifies the MIME version
Content-Description:	Human-readable string telling what is in the message
Content-Id:	Unique identifier
Content-Transfer-Encoding:	How the body is wrapped for transmission
Content-Type:	Nature of the message

- Content-Description: vom Empfänger lesbarer Text über den Inhalt der Mail
- Content-Id: eindeutige Nummer zur Kennzeichnung einer email
- Content-Transfer-Encoding: gibt an, wie die email aufbereitet werden muß, um sie lesen zu können
- Insgesamt stehen 5 Kodiermöglichkeiten zur Verfügung: ASCII, Base64 Encoding, Quoted Printable Encoding, ...
- Content-Type: gibt den Typ der Nachricht an. Es sind inzwischen >7 Typen mit mehreren Untertypen definiert:

Typen von MIME-emails (kleiner Auszug)

Type	Subtype	Description
Text	Plain	Unformatted text
	Richtext	Text including simple formatting commands
Image	Gif	Still picture in GIF format
	Jpeg	Still picture in JPEG format
Audio	Basic	Audible sound
Video	Mpeg	Movie in MPEG format
Application	Octet-stream	An uninterpreted byte sequence
	Postscript	A printable document in PostScript
Message	Rfc822	A MIME RFC 822 message
	Partial	Message has been split for transmission
	External-body	Message itself must be fetched over the net
Multipart	Mixed	Independent parts in the specified order
	Alternative	Same message in different formats
	Parallel	Parts must be viewed simultaneously
	Digest	Each part is a complete RFC 822 message

- **Text/Plain:** einfacher ASCII-Text
- **Text/Richtext:** Text mit einfachen Formatierungen. Es wird die Standard Generalized Markup Language (SGML) als Auszeichnungssprache verwendet.
- **Image/GIF:** Bild im GIF-Format
- **Image/JPEG:** Bild im JPEG-Format

USENET (NEWS)

- Weltweites Diskussionsforum über sehr viele Themen
- Vergleichbar mit einem riesigen schwarzen Brett, das nach Themen geordnet ist
- Themen werden in einer Baumhierarchie zu Themenbereichen zusammengefaßt. Jeder Themenbereich wird von einer sog. NEWS-Gruppe repräsentiert.
- Es gibt ca. 10000 verschiedene NEWS-Gruppen.
- Für einige NEWS-Gruppen gibt es Benutzer, die als Moderator und Zensor für die NEWS-Gruppe arbeiten, damit die „Netiquette“ eingehalten wird.
- Beispiele für NEWS-Gruppen der zweitobersten Baumhierarchie:

Name	Topics covered
Comp	Computers, computer science, and the computer industry
Sci	The physical sciences and engineering
Humanities	Literature and the humanities
News	Discussion of USENET itself
Rec	Recreational activities, including sports and music
Misc	Everything that does not fit in somewhere else
Soc	Socializing and social issues
Talk	Diatribes, polemics, debates and arguments galore
Alt	Alternative tree covering virtually everything

- Audio: Tondaten
- Video: Bilddaten ohne Ton (MPEG Video)
- Application/Octet-Stream: Bytestrom, der keinem der aufgeführten Typen entspricht
- Application/Postscript: email enthält Postskript-Datei
- Message/RFC 822: email enthält im Nachrichtenteil komplette andere email im RFC 822-Stil, incl. deren Header
- Message/Partial: Nachricht wurde in mehrere kleine emails zerlegt, da sie zu lang war
- Message/External-Body: Nachricht muß separat vom Internet geholt werden, z.B. von einem ftp-Server. Wird für sehr lange Nachrichten (z.B. Videofilme) verwendet.
- Multipart/Mixed: Nachricht aus mehreren Teilen, die verschiedenen Typs sein können (gemäß der gezeigten Liste). In jedem Teil muß der Typ angegeben werden.
- Multipart/Alternative: Gleiche Nachricht in verschiedenen Kodierformaten oder Sprachen
- Multipart/Parallel: Alle Teile der email müssen gleichzeitig ausgegeben werden (Multimedia-Präsentation)
- Multipart/Digest: Jeder Teil der email ist vom Typ RFC 822. Wird verwendet, wenn viele emails zu einer email zusammengesetzt werden sollen.

- Jeder kann eine Art Leserbrief an USENET schreiben, der dann als Artikel in einer bestimmten NEWS-Gruppe erscheint.
- Jede NEWS-Gruppe ist nicht nur mit einem bestimmten Themenbereich verknüpft, sondern auch mit einem individuellen Leserkreis.
- Ein an NEWS interessierte Leser erhält seine Artikel, indem er eine odere mehrere NEWS-Gruppen abonniert.
- Abonnieren, Lesen und Schreiben von Artikeln für eine NEWS-Gruppe erfolgt mit Hilfe eines NEWS-Readers (z.B. Netscape).
- Der NEWS-Reader wird auch als Benutzeragent bezeichnet, da er im Auftrag des Benutzers neue Artikel von USENET holt.
- Artikel spiegeln grundsätzlich die persönliche Meinung des Verfassers zu einem Thema wider bzw. stellen eine Mitteilung des Verfassers dar.
- Meistens entwickelt sich ein reger Meinungsaustausch innerhalb der NEWS-Gruppen, indem Artikel als Antworten zwischen den Benutzern hin und hergehen.
- Der Unterschied zu email ist, daß alles, was geschrieben wird, nicht an einen, sondern an viele Empfänger geht (alle Abonnenten einer NEWS-Gruppe).
- Bzgl. Sprache und Inhalt der Artikel sind Regeln einzuhalten, um ein Minimum an Niveau zu gewährleisten.

- Da NEWS im Gegensatz zu www rein textbasiert ist, werden in den Artikeln und Diskussionen häufig Abkürzungen verwendet.

- Beispiele für Abkürzungen:

Smiley	Meaning	Smiley	Meaning	Smiley	Meaning
: -)	I'm happy	= : -)	Abe Lincoln	: +)	Big nose
: - (	I'm sad/angry	=) : -)	Uncle Sam	: -))	Double chin
: -	I'm apathetic	* < : -)	Santa Claus	: - {)	Mustache
; -)	I'm winking	< : - (	Dunce	# : -)	Matted hair
: - (0)	I'm yelling	(- :)	Australian	8 -)	Wears glasses
: - (*)	I'm vomiting	: -) X	Man with bowtie	C : -)	Large brain

Implementierung von USENET

- Zuerst werden Artikel lokal an Unis, Firmen, Internet-Provider u.s.w. gesammelt und in Verzeichnissen abgelegt, die den Namen der NEWS-Gruppen entsprechen.

- Dann werden die Artikel mit Hilfe des Network News Transfer Protocol (NNTP) zwischen benachbarten lokalen Standorten übertragen.

- Jeder Standort hat die Rolle eines „Newsfeeders“, da er Nachrichten in das USENET einspeist.

- Der Standort, der gerade Nachrichten einspeist, handelt als Server, der Standort, der Nachrichten einliest, als Client.

- Die Rolle von Client und Server kehrt sich später um (symmetrische Kommunikationsbeziehung).

- Da jeder Newsfeeder seine gesammelten Nachrichten an seine Nachbarn weitergibt, hat jeder Standort nach einer gewissen Zeit potentiell alle Nachrichten erhalten.

- In der Praxis werden jedoch nie alle Nachrichten von allen Clienten eingelesen, da die NEWS pro Tag > 500 MB sind.

- Schließlich holen die NEWS-Reader ihre Artikel von ihrem lokalen Standort ab.

- Beim Nachrichtentransfer mittels NNTP gibt es 2 Strategien: 1. Push, 2. Pull.

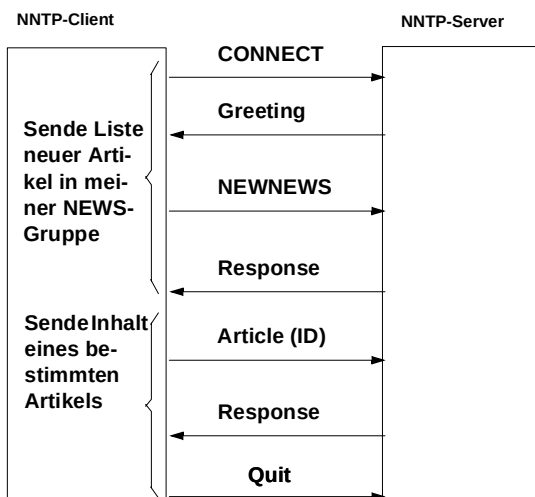
- Bei Pull fordert der Client den Nachbarn auf, der momentan die Rolle eines Newsfeed-Servers hat, ihm die neuesten NEWS zuzuschicken.

- Bei Push fordert der Newsfeed-Server den Nachbarn auf, der momentan die Rolle eines Client hat, die neuesten NEWS von ihm einzulesen.

- Die Festlegung, wer gerade Client und Server ist, geschieht über NNTP. Dazu tauschen Client und Server Kommandoworte aus.

- Bei der Pull-Strategie baut der Client eine TCP-Verbindung zum Port 119 seines Newsfeed-Servers auf (=CONNECT), dann folgt der Austausch der Kommandoworte.

Ablauf bei Pull-Nachrichtenaustausch



- Für die Push-Strategie gibt es die Kommandos POST und IHAVE.

- Mit POST stellt der Server dem Client die Artikel zur Verfügung, die er von anderen Servern erhalten hat.

- Mit IHAVE stellt der Server dem Client Artikel zur Verfügung, die lokal von Benutzern per Newsreader erzeugt wurden.

NNTP-Kommandoworte

Command	Meaning
LIST	Give me a list of all newsgroups and articles you have
NEWGROUPS date time	Give me a list of newsgroups created after date/time
GROUP grp	Give me a list of all articles in grp
NEWNEWS grps date time	Give me a list of new articles in specified groups
ARTICLE id	Give me a specific article
POST	I have an article for you that was posted here
IHAVE id	I have article id. Do you want it?
QUIT	Terminate the session

Was ist das www?

- Aus: World Wide Web FAQ:

- „The www project, [...] seeks to build a distributed hypermedia system.“
- „In practice, the web is a vast collection of interconnected documents, spanning the world [...]“
- „To access the web, you run a browser program. The browser reads documents, and can fetch documents from other sources.“
- „Information providers set up hypermedia servers which browsers can get documents from.“

- Das www ist:

- Ein offenes, verteiltes Hypermedia-System
- Basierend auf Client/Server-Anwendungen
- Eine riesige Sammlung von Informationen
- Organisiert in „web-Seiten“ (pages)
- Der bei weitem beliebteste Internet-Dienst
- Das am schnellsten wachsende Medium aller Zeiten

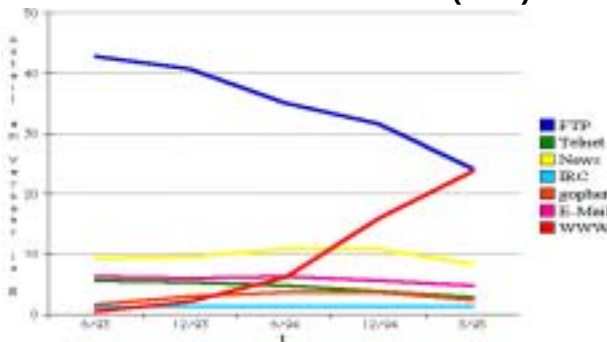
- Das www ist beliebt, weil:

- Einfach zu bedienen
- Multimedial, unterhaltsam, informativ und „in“

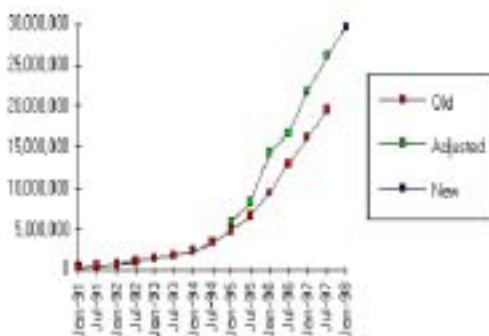
World Wide Web

(www)

Verkehr am NSF-Backbone (USA)



Zahl der Rechner am Internet



Geschichte des www

- T. Berners-Lee (1989):

„This proposal concerns the management of general information about accelerators and experiments at CERN.“

- März 1989: Tim Berners-Lee schreibt "Information Management: A Proposal"
- Oktober 1990: Geburtsstunde des „World Wide Web“
- Februar 1991: Erste Präsentation des WWW-Projektes
- August 1991: Erste Veröffentlichung im Internet
- November 1992: 26 HTTP-Server online
- Februar 1993: ca. 50 HTTP-Server online
Marc Andreessen veröffentlicht „Mosaic für X“
- Oktober 1993: über 200 HTTP-Server online
- März 1994: Gründung von Netscape
- 25. bis 27. Mai 1994: erste WWW-Conference in Genf am CERN
- Juli 1994: W3-Organisation gegründet
- Februar 1995: WWW Hauptthema des G7-Gipfels in Brüssel

- www-Konsortium (W3C): INRIA (Europa), MIT (USA), ...

- Direktor: T. Berners-Lee

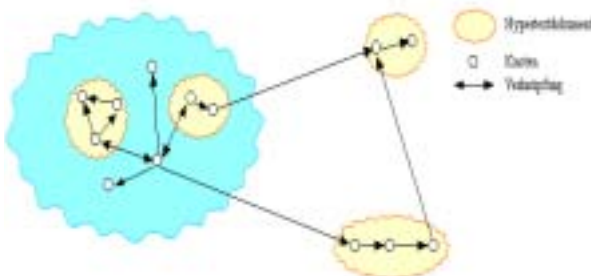
Die wichtigsten Ziele des www

- Ziel: Hypermedia-basierend und rechnerunabhängig
=> html-Seitenbeschreibungssprache
- Ziel: Einfacher Zugriff auf weltweit verteilte Informationen
=> URL-Adressierung + Client-Server-Betriebsweise
- Ziel: Plattform- und betriebssystemunabhängiger Transfer
=> http-Protokoll zum Transport des HTML-Protokolls
- Ziel: Gleiche Benutzungsoberfläche für alle Arten von Informationen
=> Browser zur Informationsdarstellung
- Ziel: Einbindung einer Vielzahl von Technologien und Dokumenttypen und leichte Erweiterbarkeit
=> Browser Plug-Ins + Java-Applets
- Ziel: Existierende Objekte wie z.B. ein Dateisystem sollen als Hypertext darstellbar sein und dadurch auch in www zugänglich werden
=> verschiedene URL-Protokolle

- Das „Anklicken“ eines Textankers bewirkt einen Übergang von einem Knoten (=Textpassage) zu einem Nachbarknoten (= verwandte Textpassage)
- Der Übergang erfolgt entlang einer gerichteten Kante
- „Lesen“ eines Hypertext-Dokuments besteht mindestens aus einer Hamilton-Tour (=einmaliger Besuch aller Knoten)

Zusammenfassung Hypertext

- Ein Hypertext wird i.a. als eine Menge aus:
 - Hypertextobjekten, oder auch Knoten, und
 - Hypertextkanten oder auch Verknüpfungenbetrachtet
- Verknüpfungen können sowohl zwischen verschiedenen Hypertextdokumenten bestehen, als auch selbstreferenzierend sein
- Representation als zyklischer gerichteter Graph = Netzwerk



Was ist ein Hypermedia?

- Hypermedia := Hypertext + Multimedia
- *Multimedia* := ≥ 1 zeitkontinuierliches Medium + ≥ 1 zeitdiskretes Medium
- Bsp. für Multimedia: Video + Text oder Audio + Graphik

Hypertext

- Theodor H. Nelson:
 - "Hypertext is the combination of natural-language text with the computer's capacities for interactive, branching or dynamic display, when explicitly used as a medium".
 - "The generic term for any text, which cannot be printed [...] on a conventional page"
- => Bietet Informationen nicht linear dar sondern delinearisiert
- Hypertext := Graph aus markierten Knoten und Kanten
- Kanten sind gerichtet
- Knotenmarkierungen := Textpassage
- Kantenmarkierungen (sog. Textanker) := unterstrichener oder hervorgehobener Text.
- Graph => Bezeichnung: „web“ => World Wide Web

Geschichte von Hypertext

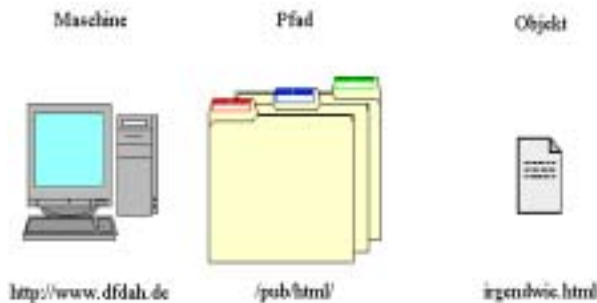
- 1945 Vannevar Bush schreibt „As we may think“
- 1965 Nelson prägt den Begriff „Hypertext“ und entwirft „Xanadu“
- 1968 Erste Demonstration von NLS
- 1978 Aspen Movie Map
- 1985 Erste Systeme werden veröffentlicht
- 1987 Hypercard erscheint
- 1990 Das WWW entsteht

www als Hypermedium

- Hypermedia ist ein Hypertextsystem mit Text, Graphik, Audio und Video, d.h. multimedial.
- Das www ist ein einziger, zusammenhängender Hypermedia-Graph.
- Der Graph des www wird über die Hypertext Markup Language (html) realisiert.
- Die Kanten des Hypermedia-Graphen werden auch als Hyperlinks bezeichnet.

URL-Adressierung

- Web-Seiten werden weltweit eindeutig über einen Universal Resource Locator (URL) adressiert.
- Jede Web-Seite, die zu einer URL gehört, ist ein Hypermedia-Dokument.
- Die URL entsteht durch Kombination existierender und neuer Namensräume:



- Eine URL besteht aus dem Tripel verwendetes Protokoll, DNS-Name des Rechners und lokaler Dateiname.

=> Die URL löst die Fragen:

- Wie heißt die page?
- Wo befindet sich die page?
- Wie kann auf die page zugegriffen werden?

- `http://www.altavista.de/cgi-bin/query?pg=aq&what=web&kl=de&r=&q=%22Harald+Rich-ter%22&d0=&d1=&what=web&search.x=35&search.y=9`

Beispiele für Protokolle

Name	Used for	Example
http	Hypertext (HTML)	<code>http://www.cs.vu.nl/~ast/</code>
ftp	FTP	<code>ftp://ftp.cs.vu.nl/pub/minix/README</code>
file	Local file	<code>/usr/suzanne/prog.c</code>
news	News group	<code>news:comp.os.minix</code>
news	News article	<code>news:AA0134223112@cs.utah.edu</code>
gopher	Gopher	<code>gopher://gopher.tc.umn.edu/11/Libraries</code>
mailto	Sending email	<code>mailto:kim@acm.org</code>
telnet	Remote login	<code>telnet://www.w3.org:80</code>

Das einfache URL-Format

Eine URL hat das Format :

`<Protokoll>://<host>[:<port>]/<path>`

`<Protokoll>=` `http|gopher|ftp|file|news| usw...`

`<host>=` IP-Number oder DNS-Eintrag

`<port>=` Netzwerk-Port des Servers,
HTTP hat Port 80 per default

`<path>=` Directorypfad zum gewünschten Objekt

- Beispiel:

`http://www.informatik.uni-augsburg.de/~richterh/offenetu-er.html`

- Zusätzlich können, durch ? eingeleitet, Parameter an die URL angehängt werden. Mehrere Parameter werden durch & unterschieden (= komplexes URL-Format)

- Parameter sind z.B. notwendig, um Einträge des Benutzers in einer Tabelle an den Server zurück zu übermitteln.

- Beispiel: Suchstring bei Altavista

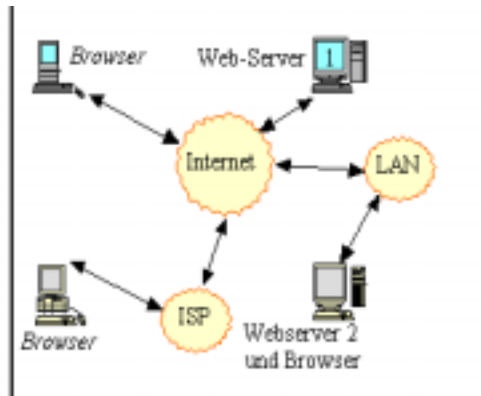
Client Server-Architektur

http

Java Applets, Browser, Java Servelets

Client-Server-Betriebsweise

- Das gesamte www basiert auf dem Client/Server-Modell.
- Auf der Benutzerseite sind die Clients (hier: Browser), auf der Server-Seite sind sog. CGI-Skripte oder Java-Servlets.



- Der Client (Browser) läuft auf dem lokalen Rechner.
- Der Client hat die Aufgaben:
 - Anfragen an den www-Server zu stellen
 - Die empfangenen Daten aufzubereiten und darzustellen
 - Navigationshilfen durchs web zur Verfügung zu stellen
 - E-Mails zu senden und empfangen (Zusatzfunktion)

- HTTP ist das Protokoll, mit dem www-Client und Server miteinander kommunizieren.

HTTP - Entwurfsziele

- Einfaches request/response- Protokoll, ASCII-Format. Minimale Transaktionsbelastung des Servers: zustandsloses Protokoll
- Für jede einzelne web-Seite wird eine TCP-Verbindung aufgebaut und wieder abgebaut; Serverprozeß belegt danach keine Ressourcen mehr.
- Caching von Seiteninhalten beim Client, um den Server zu entlasten
- Übertragung über ein zuverlässiges Transportprotokoll => TCP
- Die normale HTTP-Verbindung besteht aus 4 Phasen:
 - Connection (Verbindungsaufbau)
 - Request (Anfrage nach einer web-Seite)
 - Response (Lieferung der web-Seite)
 - Disconnection (Verbindungsabbau)
- Für die Übertragung einer web-Seite müssen alle 4 Phasen durchlaufen werden.

HTTP-Transaktion (Phase 1-4)

1.) Verbindungsaufbau (Connect)

- www-Seiten zu erstellen (Zusatzfunktion)
- Der www-Server hat die Aufgaben:
 - Anfragen von Clients beantworten, indem er die angefragten web-Seiten bereitstellt
 - Operationen auf den verwalteten pages durchzuführen, z.B. Eintragen von Werten in Tabellen.
- Die Software, die diese Aufgaben erledigt, heißt Common Gateway Interface (CGI).
- CGI-Programme werden in der Regel in sog. Skriptsprachen geschrieben, Beispiele: PERL, Python
- CGI ist das Programm auf der Serverseite, mit dem der Client kommuniziert.
- Ein CGI-Skript leistet u.a. folgende Aufgaben:
 - Auswerten der vom Client gesendeten Parameter
 - Erstellen von dynamischen Web-Seiten
 - Suchen von indizierten Web-Seiten
 - Prüfen der Zugriffsrechte auf die Web-Seiten

HTTP-Protokoll

- Tim Berners-Lee:

"The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative hypermedia information systems."

- WWW-Client baut eine TCP-Verbindung zum WWW-Server über TCP-Port 80 auf

2.) Anforderungen (Request)

- Client sendet Request über aufgebaute Verbindung (i.d.R. über die Kommandos GET, PUT und POST)

3.) Antwort (Response)

- Reaktion des Servers auf Request, z.B. angeforderte web-Seite
- Datenübertragung stets mit 7 Bit pro Zeichen (ASCII), die in Bytes verpackt sind
- zurückgelieferter Code über Status des Requests (alles OK?)

4.) Verbindungsabbau

- nach Übertragung Beendigung der Verbindung durch Server

- Die einzelnen Phasen laufen im Detail wie folgt ab:

1. Connection

- Der Client stellt zum Verbindungsaufbau eine TCP/IP-Verbindung zum Server über Port 80 her.

- Die benötigten Informationen kommen aus der URL.

2. Request

- Der Client sendet eine Anfrage an den Server in Form einer ASCII-Zeichenkette:

- Backus-Naur-Notation der Anfrage:

<Request> ::= <Simple-Request> | <Full-Request>

<Simple-Request> := <Method> <URL> CRLF

<Method> := GET

GET: Hole eine web-Seite vom Server.

Eventuelle Parameter des Client werden durch ? getrennt an die URL angehängt.

<Full-Request> :=<Request-Line> * (Header CRLF) [Data]

<Request-Line> := <Method> <URL> [<HTTP-Vers.>] CRLF

Header: enthält u.a. den MIME-Datentyp der web-Seite

[Data] ist für POST- und PUT-Methoden erforderlich

<HTTP-Version> gibt die verwendete HTTP-Version an

<Method> := GET|HEAD|PUT|POST|DELETE|LINK|UNLINK

HEAD := Hole den Header einer web-Seite vom Server

PUT := Speichere eine web-Seite beim Server

POST:= Wie PUT. Neue web-Seite wird jedoch an bestehende web-Seite angehängt.

POST dient zusätzlich zum Speichern von Benutzereingaben in ein Formular (sog. html forms)

LINK bzw. UNLINK: Auf- und Abbau von Hyperlinks zu web-Seiten

Zusammenfassung der Zugriffsmethoden

Method	Description
GET	Request to read a Web page
HEAD	Request to read a Web page's header
PUT	Request to store a Web page
POST	Append to a named resource (e.g., a Web page)
DELETE	Remove the Web page
LINK	Connects two existing resources
UNLINK	Breaks an existing connection between two resources

3. Response

- Der Server sendet eine Antwort (Response) als ASCII-String zurück:

<Full-Response> :=<Status-Line> * (Header CRLF) [Data]

Header: enthält u.a. MIME-Datentyp und Länge der Antwort.

[Data]: die angefragte web-Seite, sofern die Anfrage erfolgreich war.

<Status-Line> :=<HTTP-Version> <Status> <Reason>CRLF

<Status> Ein dreistelliger Status-Code

<Reason> Ein ASCII-Text zur Erläuterung des Status-Code

Beispiel für Status-Codes

Status	Reason	Erklärung
200	OK	Der Request wurde erfüllt
201	Created	web-Seite wurde erstellt
202	Accepted	Der Request wird bearbeitet
301	Moved	Web-Seite hat neue URL
400	Bad Request	Zeigt einen Syntax-Fehler o.ä. an
401	Unauthorized	Unauthorisierter Zugriff
402	Payment required	Kostenpflichtiger Zugriff
404	Not Found	Server kann web-Seite nicht finden
500	Internal Error	Interner Fehler des Servers

Beispiele für Response-Header

Header	Bedeutung
Server:<server-type>	Der Name des Server-Programms z.B. Apache
Content-Length:<length>	Die Länge (in Byte) der Antwort
Content-Type:<MIME-Type>	Der MIME-Type der web-Seite
Title:<title>	Der Titel der web-Seite
Date:<date>	Das Erstellungs- oder Änderungsdatum der web-Seite
Expires:<date>	Gibt an, wann web-Seite ungültig wird.

-RequestHeader istähnlichwieResponse Header

4. Disconnection

- Der Server beendet die TCP/IP- Verbindung, sobald die Web-Seite übertragen ist.

Web Browser

- Browser müssen eine Vielzahl von Graphik-, Audio- und Video-Formaten darstellen bzw. wiedergeben können.

- Dazu dienen Plug-Ins, die als „Software-Einschubmodule“, den Browser erweitern.

- Plug-Ins werden beim ersten Bedarf vom Netz in den Browser geladen bzw. bei Aktualisierung des Browsers erneut geladen

- Alle Browser interpretieren die per http transportierten web-Seiteninformationen, die wiederum in der Seitenbeschreibungssprache html verfasst sind.

- Da in den html-basierten web-Seiten sog. applets enthalten sein können, muss jeder Browser zusätzlich in der Lage sein, applets auszuführen.

- Browser können in jeder Programmiersprache geschrieben sein, am häufigsten werden jedoch die Standardsprachen C/C++ bzw. Java verwendet.

- Java-basierte Browser bilden insofern eine Sonderrolle, als dass hier zusätzlich der Browser-Code interpretiert wird (von der sog. Java Virtual Machine, JVM)

- Im Extremfall wird bei einem Java-basierten Browser sogar erst der Java-Code für den Browser vom Netz geladen
- Danach werden je nach Bedarf Java Plug-Ins zur Darstellung von Dateiformaten (pdf, postskript, JPEG, ...) und für Audio- und Video-web-Seiten geladen.
- Beispiele für Browser: Netscape, Mozilla, Opera, MS Internet Explorer
- Beachte: die lokale Festplatte wird als Cache für web-Seiten benutzt. Angezeigte web-Seiten sind deshalb nur nach RELOAD-Kommando und entsprechender Konfiguration im Browser aktuell, da web-Seiten zunächst aus dem Browser Cache angezeigt werden.

Applets

- Applet = applicationlet = kleines Anwendungsprogramm, das bei der Anzeige einer entsprechenden web-Seite vom Server geladen und beim Client ausgeführt wird
- Applets werden benutzt, um web-Seiten interaktiver zu gestalten bzw. mit Animationen zu versehen
- Beispiele für applets: Seitenbesuchszähler, Taschenrechner, Gästebücher, bewegte „Männchen“.
- Applets können in verschiedenen Sprachen geschrieben werden, am häufigsten wird Java oder Javaskript verwendet.
- Java Applet = kleines Anwendungsprogramm in Java, das vom Server geladen und Client-seitig ausgeführt wird

:

Java Servlet

- (Kleines oder grosses) Anwendungsprogramm, das beim Aufruf einer entsprechenden web-Seite Server-seitig ausgeführt wird
- Wird benutzt, um die übliche CGI-Funktionalität zu erweitern
- Dies ist dadurch möglich, dass Servlets in Java geschrieben sind. Damit können alle existierenden Java-Bibliotheken verwendet werden, was die Funktionalität der CGI-Schnittstelle deutlich erhöht.

Java Server Pages

- Java Server Pages sind eine Erweiterung der Java Servlet-Technologie, die den CGI-Funktionsumfang nochmal erweitert.

Internet-Anwendungen

- Die meisten Internet-Anwendungen beruhen auf einer oder mehrerer der folgenden Software-Technologien:
 - Common Gateway Interface (CGI)
 - Remote Procedure Calls (RPCs)
 - Remote Method Invocations (RMIs)
 - Distributed Computing Environment (DCE)
 - Common Object Request Broker Architecture (CORBA)
 - Distributed Component Object Model (DCOM)
- RMI, DCE, CORBA und DCOM wiederum beruhen auf RPCs
- Die obigen Technologien werden als Middleware bezeichnet, da sie zwischen Anwendung (engl. application) und dem Betriebssystem sitzen
- Middleware hilft durch die Bereitstellung allgemein benötigter Software-Infrastruktur für verteilte Programmierung.
- Heutzutage beruhen viele Internet-Anwendungen auf kommerzielle oder frei verfügbare Middleware. Dies trifft auch auf einige Internet-Dienste zu.
- Beispiel für einen Internet-Dienst, der Middleware verwendet, ist das Network File System (NFS). Es beruht auf RPCs.
- NFS war ursprünglich eine SUN-proprietäre Internet-Anwendung. Durch die weite Verbreitung entstand jedoch eine RFC (RFC 1094), und NFS wurde von der Anwendung zum Dienst.

Das Network File System (NFS)

- Wurde ursprünglich von der Fa. SUN für SUN-Arbeitsplatzrechner (Workstations) entwickelt
- Schon lange Standard bei allen UNIX/Linux-Systemen
- Implementiert ein verteiltes Dateisystem, das nach einer Initialisierungsphase für den Benutzer „transparent“ ist
- => Benutzer sieht nicht mehr, ob Dateien lokal oder entfernt gespeichert sind
- Es gibt Software-Konverter, die NFS mit einem verteilten Windows-Dateisystem verbinden (z.B. Samba)
- Der kommerzielle Nachfolger von NFS ist das Andrew File System (AFS)
- AFS kann zur Klasse der Internet-Anwendungen gezählt werden, da es keine RFC dafür gibt.

Merkmale von nfs

- Transparenter Zugriff auf entfernte Dateien aufgrund nahtloser Integration in das Betriebssystem/Dateisystem
- Offenes System (Spezifikation und Referenzimplementierungen sind öffentlich und oftmals kostenfrei)
- Eignet sich für heterogene Netzwerke (=gemischte UNIX/Windows/MacOS-Dateisysteme)

- Bezüglich des Zugriffs auf nicht lokale Dateien ist er zugleich **NFS-Client**, sobald er seinerseits auf entfernte NFS-Dateien zugreift. => symmetrische Client/Server-Beziehung

- NFS-Server und NFS-Client kommunizieren über das NFS-Protokoll miteinander.

- Ein Rechner, der auf eine nichtlokale Datei zugreifen möchte, muß zuvor dasjenige exportierte Unterverzeichnis „mounten“, in dem sich die gewünschte Datei befindet.

(to mount: anbringen, befestigen)

- NFS abstrahiert das lokale Dateisystem jedes NFS-Rechners zu einem virtuellen File-System, das Betriebssystem-unabhängig ist.

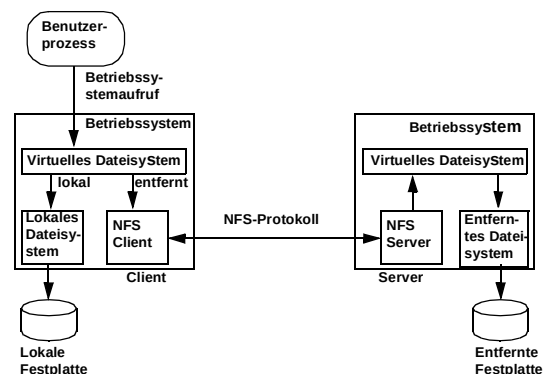
- Alle NFS-Dateioperationen gehen zuerst an das virtuelle Dateisystem.

- Portierung ist einfach
- Basiert auf einem zustandslosen Client/Server-Modell
- Kommunikation über UDP-Transportprotokoll, also verbindungslos
- Rechner importieren/exportieren ihre Dateiverzeichnisse oder Teile davon als Unterbäume in das Netz
- Keine Funktionen der ISO-Darstellungsschicht, nur Lesen und Schreiben von Byte-Strömen
- Betriebssystemunabhängige Datenrepräsentation kann von Zusatz-Software auf Schicht 7 vorgenommen werden: External Data Representation (XDR)

Funktionsweise von NFS

- Jeder Rechner, der sich an NFS beteiligen will, kann einige oder alle seiner lokalen Dateien nach NFS „exportieren“ (für NFS freigeben). Sie sind dann für andere Rechner sichtbar.
- In der Regel werden nicht einzelne Dateien nach NFS exportiert, sondern ganze Unterverzeichnisse.
- Jeder Rechner fungiert als **NFS-Server** für seine lokalen NFS-Dateien, d.h. er sorgt dafür, daß andere Rechner auf die Dateien zugreifen können, die er nach NFS exportiert hat.

- Lokale Dateizugriffe werden von dort an das reale File-System weitergereicht, entfernte gehen an den NFS-Client



NFS-Protokoll

- Basiert auf dem Datagramm-Dienst UDP des Internet
- => Es gibt keine feste Verbindung zwischen Client und Server auf der Transportebene des Internet.
- Jeder Server ist aus Fehlertoleranzgründen „zustandslos“
- => Er speichert weder die Historie der Zugriffswünsche noch den Status bzgl. ihrer Erfüllung.
- => Bei entfernten Datei-Zugriffen gibt es keine „open“- oder „close“-Operation, sondern nur read, write und lookup (= dir-Befehl, bzw. ls -la)

- Das Mounten von Unterverzeichnissen wird über ein eigenes Protokoll durchgeführt (Mount-Protokoll)
- Mount- und NFS-Protokoll werden über SUN Remote Procedure Calls (RPCs) abgewickelt.

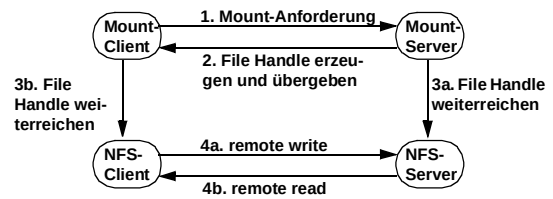
Mount-Protokoll

- Das NFS-Protokoll dient nur zum Lesen und Schreiben, das Bereitstellen entfernter Dateien geschieht durch das Mount-Protokoll
- Mount verbindet ein entferntes Datensystem mit einem lokalen Verzeichnis, d.h. der gesamte entfernte Dateibaum wird in das lokale Verzeichnis „eingehängt“.
- Danach kann über NFS auf die entfernten Dateien wie auf lokale Dateien zugegriffen werden.
- Nach Ende eines Mount-Vorgangs stellt Mount einige Informationen für NFS zur Verfügung, z.B. Rechnernamen und Pfad.
- Mount basiert wie NFS auf dem Client/Server-Modell
- Mount- und NFS-Server-Prozesse sind als System-Demons realisiert und werden beim Hochfahren des Servers automatisch gestartet.
- Beim MOUNT-Vorgang wird auf der Server-Seite ein eindeutiger Dateisystem-Steuerblock („File Handle“) erzeugt und an den Client gegeben.

Lock-Manager

- Problem: Was passiert wenn gleichzeitig >1 Klienten auf dieselbe Datei schreibend zugreifen wollen?
- Lösung: Der sog. Lock-Manager ermöglicht exklusiven Schreibzugriff eines Clients auf eine Datei durch Aussperren anderer Klienten
- Neues Problem: wechselseitige Blockierung dadurch, dass der Zugriff-habende darauf wartet, dass der ausgesperrte auf die exklusive Datei schreibt (Deadlock)
- Lösung: nicht vorhanden; das Problem wird als Programmier- oder Bedienfehler des Benutzers betrachtet

- Der NFS-Client verwendet diesen bei allen späteren Zugriffen auf den entfernten Verzeichnis(teil)-Baum

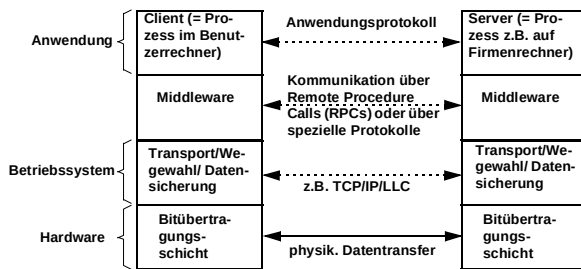


NFS und Sicherheit (security)

- Anschluß an ein Rechnernetz ist ein potentielles Sicherheitsrisiko, z.B. lassen sich Ethernet-Pakete abfangen
- NFS unterstützt Verschlüsselung
- Zusätzlich:
 - Mount- und NFS-Zugriff sind nur von sog. privilegierten Ports erlaubt (Port-Nr. <1024), die nur von bekannten Anwendungen und dem Betriebssystem benutzt werden dürfen
 - Bei Mount- und NFS-Zugriff werden Rechnernamen und IP-Adressen mit den lokalen Listen der „friendly hosts“ und der „unfriendly hosts“ verglichen (diese stehen bei UNIX in /etc/hosts.allow bzw. /etc/host.deny)

Middleware

- Definition: Middleware ist die Software-Schicht zwischen Betriebssystem und Benutzeranwendung



- Beispiele für Middleware:

- CGI (Common Gateway Interface)
- DCE (Distributed Computing Environment)
- DCOM (Distributed Component Object Model)
- CORBA (Common Object Broker Architecture)
- Novell Netware

Middleware - Überblick

- CGI: www-Standard. Software, die es ermöglicht, externe Programme auf einem Server zu starten, und zwar von HTML-Dokumenten aus, die auf einem Client angezeigt werden
- DCE: Standard der Open Group/Open Software Foundation (OSF = Industriekonsortium)

CGI Middleware

- CGI ist eine Software, die es ermöglicht, externe Programme von HTML-Dokumenten aus zu starten
- Dabei wird das externe Programm von einem web-Server ausgeführt, während das HTML-Dokument auf einem Client-Rechner angezeigt wird
- => CGI erlaubt es, aus einem WWW-Browser heraus über das Internet Programme auf einem Server auszuführen
- Die aufgerufenen Programme sind entweder als Skript, Java-Servlet oder in einer konventionellen Programmiersprache (C, C++, Visual Basic, ...) geschrieben
- Das Ziel von CGI ist es, die Inhalte von web-Seiten gemäß den Eingaben des Browser-Benutzers zu verändern

=> Dynamische web-Seiten-Erzeugung mittels CGI

Vorteile dynamischer web-Seiten

- Dynamische web-Seiten erlauben:
- Zugriffszähler, Gästebücher
- Animationen, Spiele
- Taschenrechner und ähnliche Anwendungen
- einfache Suchmaschinen, Online-Registrierung
- Einträge in Formulare z.B. für Online-Bestellungen
- Einträge in und Abfragen von Datenbanken (Queries)

- DCE dient dazu, verteilte Anwendungen leichter erstellen zu können. Hauptaufgabe: Verbergen der Heterogenität der Betriebssysteme und Rechnerarchitekturen

- DCE hat eine Client/Server- Architektur und ist RPC-basiert

- CORBA: Standard der Object Management Group (OMG = Industriekonsortium)

- CORBA: dient ebenso wie DCE als Middleware, um verteilte Anwendungen leichter zu erstellen

- CORBA ist objektorientiert, sprachunabhängig; basiert auf sog. Methodenfernaufrufen; z.T. auch DCE-basiert;

- DCOM (Distributed Component Object Model): Microsoft-eigene Entwicklung;

- DCOM dient ebenso wie DCE als Middleware, um verteilte Anwendungen leichter zu erstellen

- DCOM ist „Objektorientiert“ und RPC-basiert

- Java RMI (= Remote Method Invocation = Methodenfernaufruf): Entwicklung von Sun

- Java RMIs sind einfache Methodenfernaufrufe als Teil der Sprache Java bzw. des Java Development Kit (JDK), um verteilte Anwendungen leichter zu erstellen

- Java RMIs sind RPC-basiert

Erstellen einer dynamischen web-Seite

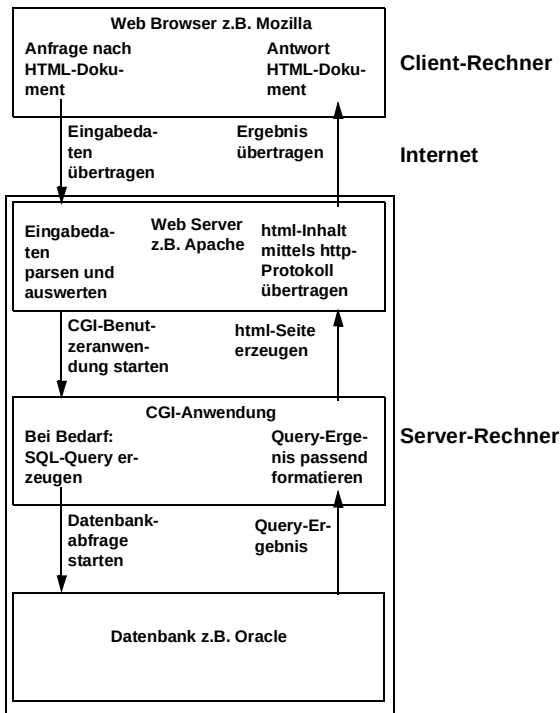
- Soll das CGI-Programm eine Antwort (z.B. das Ergebnis einer Datenbankabfrage) an den Browser zurückliefern, wird der Inhalt einer web-Seite neu erzeugt
- Die web-Seite ist nicht physisch auf der Platte vorhanden, sondern wird vom CGI-Programm per print auf stdout geschrieben
- http leitet alles von stout des Servers zu stdin des Client und damit zum Browser weiter
- http überträgt die web-Seite zusammen mit einem Header ähnlich der einer MIME-email

http-Header für dynamisch erzeugte web-Seiten

Header	Beschreibung
Version	http-Versionsnr.
Name	Name des Server-Programms
Content-length	Zahl der Bytes auf stdout
Content-type	MIME-Typ der Daten
Pragma	schaltet das Caching der web-Seite ein
Status	Status der http-Anforderung
Set-Cookie	Schlüsselwort für Cookie-Erstellung

- Nach Erstellen und Übertragen der dynamisch erzeugten web-Seite wird diese vom Client-Browser dargestellt

Dynamische web-Seiten mit CGI



- Die dynamische web-Seiten-Erzeugung umfaßt folgende Schritte:

- 1.) Übertragen der Benutzereingaben vom Client zum Server durch http
- 2.) Prüfen und Auswerten der Eingaben durch CGI
- 3.) Prüfen der Zugriffsrechte des Benutzers auf dem web-Server durch CGI
- 4.) Speichern der Benutzereingaben beim Client
- 5.) Erstellen der web-Seite durch CGI
- 6.) Transfer der web-Seite zum Client durch http

- Das Prüfen und Auswerten der Benutzereingaben ist in konventionellen Programmiersprachen aufwendig, da Parsen der Texteingaben „von Hand“ erfolgt

=> Verwendung von speziellen Skriptsprachen (PERL, etc.), die komfortabel String-Verarbeitung unterstützen

Formulare in Web-Seiten (Forms)

- Formulare werden als web-Seiten mit Leerfeldern dargestellt, in die der Benutzer Eingaben eintippt

- Formulare sind nützlich:

- um gleichartig strukturierte Auskünfte von Benutzern einzuholen
- um Benutzern das Eintragen und Suchen in Datenbanken zu ermöglichen

- um mit dem Benutzer individuell umzugehen, z.B. bei einer Bestellung in einem virtuellen Kaufhaus (eCommerce)

- Die Auswertung einer Formular-web-Seite erfolgt durch Aufruf eines CGI-Programms beim Server

Aufruf eines CGI-Programms für Formulare

- Der Aufruf eines CGI-Programms erfolgt durch Eintippen von Benutzerangaben in ein Formular in der HTML-Beschreibung des Formulars mittels des Kommandos:

```
<FORM ACTION = URL METHOD = HTTP-Methode>
```

```
</FORM>
```

- Dabei ist:

- URL: die Angabe einer Datei auf dem Server, die das CGI-Programm enthält, das beim Eintragen von Daten in das Formular aufgerufen werden soll
- HTTP-Methode: die Art der Parameterübergabe an das CGI-Programm. Möglich sind POST und GET

Beispiel:

```
<FORM ACTION="http://www.informatik.uni-augsburg.de/cgi-bin/programm.pl" METHOD="POST">
```

```
</FORM>
```

Art der Parameterübergabe zwischen Client und Server

- 1.) METHOD="POST": Der Client schreibt die Benutzereingaben als Parameter auf stdout (Standardausgabe).

- Da stdout über das HTTP-Protokoll mit der Netzwerkkarte verbunden ist, wandert die Ausgabe zu stdin des Servers

- 2.) METHOD="GET": Der Client hängt die Benutzereingaben an die URL an, die auf das auszuführende Programm zeigt

- Bei der URL-Übergabe wird zur Kennzeichnung des Beginns der Parameter das ? verwendet

- Bei der stdout-Übergabe wird die Zahl der auf stdout geschriebenen Bytes in die Umgebungsvariable CONTENT_LENGTH des Servers eingetragen

- Die Eintragung in CONTENT_LENGTH nimmt die Instanz beim Server vor, die HTTP implementiert

- Bei der URL-Übergabe trägt http-Instanz die Werte der Parameter in die Umgebungsvariable QUERY_STRING des Servers ein

=> Bei METHOD="POST" liest das CGI-Programm zuerst CONTENT_LENGTH und dann stdin

=> Bei METHOD="GET" liest das CGI-Programm QUERY_STRING

- Für GET und POST gibt es weitere Umgebungsvariable beim Server, deren Wert vom Client-Browser bestimmt wird (über die http-Instanz)

- Liste der Umgebungsvariablen:

Umgebungsvariablen	Beschreibung
HTTP_USER_AGENT	Welcher Browser vom Client verwendet wird
DOCUMENT_ROOT	Server-Verzeichnis der Web-Dokumente
REQUEST_METHOD	Art der http-Anforderung
CONTENT_TYPE	MIME-Typ der http-Anforderung
HTTP_ACCEPT	MIME-Typen, die der Client akzeptiert
HTTP_REFERER	Browser URL vor Aufruf des CGI-Programms
HTTP_COOKIE	Benutzereingaben, die beim Client gespeichert sind

- Im CGI ist festgelegt, in welchem Format die Parameter auf stdout geschrieben bzw. an die URL angehängt werden

CGI-Parameterformat

- Die Felder der Formulare mit den Benutzereingaben werden als Paare in der Form Schlüsselwort = Wert dargestellt
- Diese Paare wurden zuvor durch html-Anweisungen (sog. html-Tags) und die Eingaben des Benutzers festgelegt:

Beispiel für html-Tag zur Formulareingabe:

Straße: <INPUT TYPE = "text" NAME = "Straßenname">

Versteckte Felder

- In einer web-Seite werden versteckte Felder vom Browser nicht angezeigt
 - Der Inhalt versteckter Felder wird dennoch beim CGI-Aufruf als Parameter mit übergeben
 - Benutzereinträge einer web-Seite werden in versteckte Felder der nachfolgenden web-Seite geschrieben
- => Daten können so von einer web-Seite zur nächsten weitergereicht werden (mit GET oder POST vom Client zum Server und mit Schreiben in versteckte Felder zurück zum Client)

Cookies

- Zweite Möglichkeit zur persistenten Speicherung von Benutzereingaben sind Cookies (Cookie=süßer Keks, Plätzchen)
- Ein web-Server kann beim Client mit Hilfe des jeweiligen Browsers eine Textdatei anlegen, in der einige Eingaben des Browser-Benutzers gespeichert werden
- Benutzereingaben werden in dieser Datei als Paare der Form Schlüsselwort = Wert gespeichert, z.B. Straßenname="Maximilianstr."
- Jedes Paar dieser Art wird als Cookie bezeichnet

- Die Eingabe des Benutzers im Feld "Straße:", z.B. "Maximilianstr.", wird in der Variablen "Straßenname" gespeichert

=> das Paar Schlüsselwort = Wert wird zu Straßenname="Maximilianstr."

- Mehrere solcher Paare werden durch & getrennt

- Leerzeichen werden durch + ersetzt

- Umlaute, Sonderzeichen usw. werden durch den hexadezimalen ASCII-Code ausgedrückt, wobei ein % als Kennung vorangestellt wird

Beispiel:

Vorname=Michael+J%FCrgen&Nachname=Gr%FCndler

Speicherung der Benutzereingaben

- Zwischen Client und Server besteht nur zur Anforderung einer web-Seite und zu deren Übertragung jeweils eine TCP-Verbindung
 - Zwischen der Übertragung nachfolgender web-Seiten merkt sich der Server keine Information vom Client
- => Bei Formularen, die mehrere web-Seiten umfassen, müssen Benutzereinträge zwischengespeichert werden
- Die Zwischenspeicherung erfolgt beim Client in sog. versteckten Feldern oder „Cookies“

=> ein Cookie ist ein Eintrag in einer auf Wunsch eines Servers vom Browser lokal angelegten Datei im Format Schlüsselwort = Wert

- Beispiel: Der Netscape Browser legt zur lokalen Speicherung von Cookies die Datei cookie.txt an
 - Alle nachfolgenden Zugriffe auf web-Seiten, die vom Netscape Browser durchgeführt werden, können potentiell auf diese Datei zugreifen, sofern bestimmte Bedingungen zum Schutz des Benutzers erfüllt sind.
 - Dazu zählen eine entsprechende Browser-Konfiguration, sowie die Überprüfung, ob die Domain des Servers und eine Pfad-Angabe im web-Dokument erlaubt sind
 - Abgeprüft wird die Domain und die Pfadangabe anhand zweier Paare Domain = <DNS-Name> und path = Pfadangabe bei der Cookie-Definition
 - In Cookies können benutzerspezifische Daten gespeichert werden
- => Cookies können gegen das Datenschutzgesetz verstoßen

Beschränkungen bei Cookies am Beispiel des Netscape Browsers

- Jeder web-Server darf maximal 20 Cookies (=Einträge) auf einem Client anlegen
- Die maximale Größe pro Cookie ist 4 KB

- Maximal dürfen 300 Cookies beim Client existieren (von verschiedenen Servern)

=> Größe von cookie.txt ist auf 300*4KB = 1,2 MB beschränkt

- Beachte: das Erstellen, Schreiben und Lesen von cookie.txt wird vom Netscape-Browser vorgenommen.
- Andere Browser verwenden gegebenenfalls andere Dateien

Erstellen eines Cookies

- Ein Cookie wird vom Server beim Client dadurch erstellt, daß das Set-Cookie-Schlüsselwort im Header der http-Response enthalten ist

- Syntax des Set-Cookie-Schlüsselworts:

Set-Cookie: NAME=VALUE; expires=DATE;

path=PATH; domain=DOMAIN_NAME; secure

- Nach dem Rücksenden des oder der Cookies zum Server wird der Inhalt durch http automatisch in die Umgebungsvariable HTTP_COOKIE des Servers kopiert

- Das CGI-Programm kann dann auf HTTP_COOKIE lesend zugreifen

- Das Set-Cookie-Schlüsselwort kann vom CGI-Programm des Servers auf 2 Arten gesetzt werden:

- 1.) Spezieller Prozeduraufruf bei der http-Instanz des Servers (nur theoretisch)
- 2.) Spezielles HTML-Kommando im Quelltext der web-Seite (Regelfall)

- Der Browser interpretiert das http-Protokoll und die html-Seitenbeschreibung. Empfängt er ein spez. html-Kommando, wird er so konfiguriert, als ob im http-Protokoll das Set-Cookie-Schlüsselwort gesetzt wäre!

Spezielles HTML-Kommando zur Cookie-Erstellung

- Das HTML-Kommando ist Teil des sog. META-tags
- Das META-tag spezifiziert entweder Information über die web-Seite oder erzeugt ein Schlüsselwort im HTTP response header
- Es wird genau dann ein Response-Header-Schlüsselwort erzeugt, wenn beim META-tag der Parameter HTTP-EQUIV angegeben wird:

<META HTTP-EQUIV="FieldName" CONTENT="value">

- Bedeutung der Parameter (Datenfelder in cookie.txt):

Datenfeld	Beschreibung
Name	Name des Schlüsselworts
Value	Inhalt des Schlüsselworts
Expires	Verfallsdatum des Cookies
Domain	Domain, an welche dieses Cookie automatisch gesendet werden soll
Secure	wenn TRUE, darf Cookie nur mit HTTPS gesendet werden
path	Pfadangabe auf dem Server-Rechner

- Beim Anfordern einer web-Seite (URL) von einem Server sendet der Client automatisch alle bei ihm gespeicherten Cookies zurück, die folgende Bedingungen erfüllen:

- 1.) Der Server muß in der im Domain-Schlüsselwort angegebenen DNS-Domain liegen. Seine Domain ist aus der URL ersichtlich

Beispiel : Wenn Domain = uni-augsburg.de gesetzt ist, dann wäre ein Zugriff für informatik.uni-augsburg.de erlaubt

- 2.) Pfadangabe in der angeforderten URL muß als Zeichenkette bei der path = Pfadangabe enthalten sein

Beispiel : Wenn path = /offene gesetzt ist, dann wäre eine URL der Art http://www.informatik.uni-augsburg.de/~offen-etuer.html zulässig

- Beide Bedingungen müssen zum Datenschutz gelten

- Es können verschiedene Response-Header-Schlüsselwort erzeugt werden. Eines davon ist „Set-Cookie“

Beispiel:

```
<META HTTP-EQUIV="Set-Cookie" CONTENT="Straße=Maximilianstr" ;expires=Friday, 31-Dec-99 23:59:59 GMT; path="/">
```

- Das Löschen eines Cookies durch den Server beim Client erfolgt dadurch, daß das gleiche Cookie noch einmal geschrieben wird, aber mit einer Expires-Zeitangabe, die in der Vergangenheit liegt

- Cookies gelten als Sicherheitslücke, sind aber weit verbreitet und für mehrseitige Formulare oft unerlässlich

Remote Procedure Calls (RPCs = Prozedurfernaufrufe)

- Internet-Anwendungen werden grundsätzlich auf einem Rechnernetz verteilt ausgeführt, im einfachsten Fall nach dem Client/Server-Modell

=> zwischen den Rechnern ist Kommunikation erforderlich

- Zur Durchführung der Kommunikation sind verschiedene Verfahren üblich:

- 1.) Anwendungsspezifische Protokolle, die auf der Transportschicht aufsitzen (über Socket-Aufrufe)
- 2.) Client-/Server-Modell mit Prozedurfernaufrufen
- 3.) Verteiltes-Objekt-Modell mit Methodenfernaufrufen

- Für spezielle Kommunikationsnetze gibt es noch:

- 1.) Botschaftenaustausch nach dem Rendezvous- oder Mailbox-Modell
- 2.) gemeinsame Variable über verteilten gemeinsamen Speicher

- Hier soll nur das Client-/Server-Modell mit Prozedurfernaufrufen diskutiert werden

=> Es gibt dabei mindestens zwei beteiligte Partner: einen Client- und einen Server-Rechner

=> Erweiterung von normalen Prozeduraufrufen damit Unterprogramme auf anderen Rechnern als dem aufrufenden ausgeführt werden können

=> „Prozedurfernaufrufe“ (Remote Procedure Calls, RPCs) zusätzlich zu lokalen Prozeduraufrufen

- Der Rechner, der eine entfernte Prozedur aufruft, hat die Rolle eines Klienten(Auftraggeber)

- Der Rechner, der einen RPC ausführt ist, ein Server bzgl. dieses RPC (Auftragnehmer)

- Die Rollen von Client und Server können fest vergeben sein oder dynamisch wechseln

- Die Heterogenität der Rechner soll dem Programmierer verborgen bleiben

=> Zusatzsoftware erforderlich für:

- die automatische Formatkonversion
- zum Starten von Unterprogrammen auf entfernten Rechnern

Übersicht über RPCs

- Prozedurfernaufrufe gibt es seit Beginn der 80er Jahre
- Sind inzwischen weit verbreitet (für fast jedes OS)
- Sind eine Möglichkeit der verteilten Programmierung neben:

- Zusätzlich kann der Server auf mehrere Rechner verteilt implementiert sein (aus Geschwindigkeits- oder Zuverlässigkeitsgründen)

- Beteiligte Rechner sind i.a. heterogen (anderer Typ, anderes Betriebssystem, andere interne Zahlendarstellung, andere Hardware-/Software-Ausstattung, u.s.w.)

Beispiele für Inkompatibilitäten bei der Rechnerkommunikation

1.) Byte-Anordnung im Wort:

- „Big endian“: most significant Byte liegt auf der niedrigsten Speicheradresse (alle SUN-Rechner)
- „Little endian“: least significant Byte ist auf der niedrigsten Speicheradresse (alle Intel-Prozessoren)

2.) Wortausrichtung im Adreßraum

- 32-Bit-Worte beginnend bei jeder Byte-Adresse
- 32-Bit-Worte beginnend bei allen durch 4 teilbaren Byte-Adressen

3.) Interne Repräsentation von Zeichenketten (Strings)

- Länge des Strings als erstes Zeichen im String
- String wird mit spez. Zeichen abgeschlossen
- Abschluß einer Zeile mit <CR><LF>
- Abschluß einer Zeile mit <LF><CR>

- Ziel: die räumliche Verteilung der Benutzer-Anwendung soll dem Programmierer möglichst leicht gemacht werden

=> möglichst automatische Konversion zwischen den verschiedenen Formaten

- Anwendungsspezifischen Protokollen
- Verteilten Objekten (distributed objects)
- Botschaftenaustausch (message passing)
- Verteiltem gemeinsamen Speicher (distributed shared memory)

- RPCs sind die Basis für:

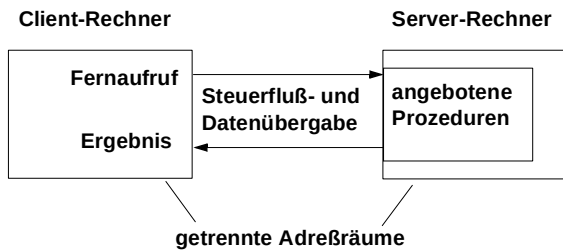
- DCE (Distributed Computing Environment)
- CORBA (Common Object Request Broker Architecture)
- JAVA RMI (Remote Method Invocation)
- MS DCOM (Distributed Component Object Model)

- Definition nach Nelson [1981]:

"RPCs sind die synchrone Steuerfluß- und Datenübergabe ... zwischen Programmen in Form von Prozeduraufrufen zwischen verschiedenen Adreßräumen über einen schmalen Kanal"

- Sind wie lokale Prozeduraufrufe blockierend (= "synchron")

=> Client muß i.d.R. warten, bis Prozedur fertig ausgeführt ist. (Gibt auch Ausnahmen)



Unterschiede zu lokalen Prozeduraufrufen

- Client- und Server-Programm laufen in verschiedenen Adreßräumen (getrennte Rechner!)
- => Parameterübergabe ist kompliziert, speziell bei Zeigern (Pointer) als Parameter
- Netzwerkkommunikation ist involviert
- => Transfer von Parametern und Ergebnis dauert lang
- Netzwerkprobleme können zu RPC-Fehlern führen
- => erweiterte "Fehlersemantik"
- Zusätzlich gibt es asynchrone (nicht blockierende) RPC-Varianten
- => Komplexere Programmierung, aber Nebenläufigkeit zwischen Client und Server möglich (nur bei asynchronen RPCs)

- => Parameter müssen in einer vereinbarten Rechner- und Betriebssystemunabhängigen Syntax ausgetauscht werden. Dazu gibt es zwei Möglichkeiten:
- 1.) Symmetrisches Verfahren: Client und Server verwenden ein gemeinsames, drittes Format
- => Konvertierungsroutinen sowohl bei Client wie bei Server erforderlich
- 2.) Asymmetrisches Verfahren: Man einigt sich auf das Format des Client oder des Server
- => Konvertierungsroutinen nur bei Client oder bei Server erforderlich
- Bei SUN RPCs wird ein gemeinsames drittes Format verwendet, die sog. External Data Representation, XDR
- In beiden Fällen müssen die Parameter erst konvertiert und dann einer nach dem anderen über das Netz geschickt werden (Serialisierung bzw. „marshalling“, *to marshall = anreihen*)
- Bei der Serialisierung muß vereinbart werden, in welcher Reihenfolge die Elemente der Datenstrukturen übertragen werden

(Bemerkung: Nebenläufigkeit ist der Oberbegriff sowohl für Parallelverarbeitung, d.h. echte Gleichzeitigkeit als auch für zeitliche Verzahnung, d.h. time-sharing)

Parameterübergabe bei RPCs

- 1. Problem:

Aufrufparameter und Ergebnis können nicht auf dem Stack abgelegt werden (getrennte Adreßräume)

- Lösung:

- Call-by-Value-Parameter wie INT oder CHAR werden als Werte über das Netz übertragen
- Datenstrukturen aus INT oder CHAR, wie Bäume, Listen oder Keller werden Wert für Wert übertragen
- Zeiger auf Datenstrukturen (Call-by-Reference-Parameter) werden dereferenziert und die Inhalte der Datenstrukturen werden Wert für Wert übertragen
- Beim Server wird eine Kopie der Datenstruktur wieder aufgebaut. Anschließend erfolgt eine neue Referenzierung auf die Kopie im Server-Adreßraum
- => Die Semantik von Zeigern ändert sich von Call-By-Reference zu Call-By-Copy-and-Restore

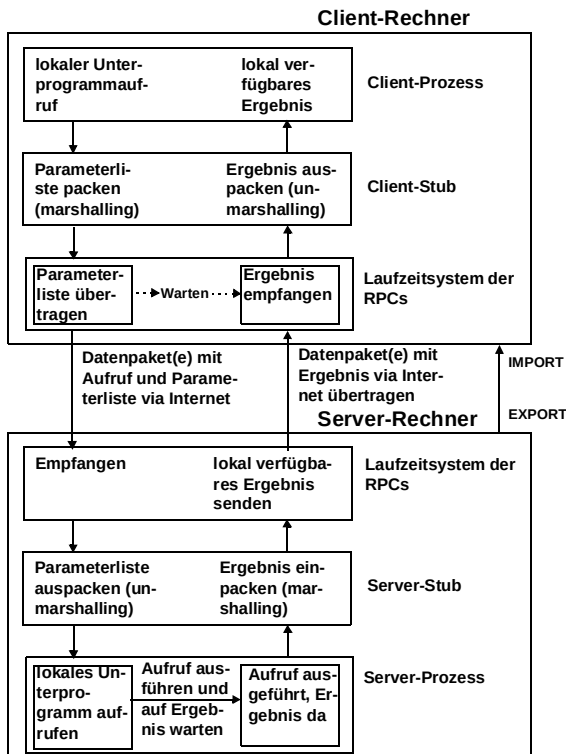
- 2. Problem:

Aufgrund der Heterogenität der vernetzten Rechner sind Byteordnung, Wortausrichtung, Zahlen- und Zeichenkodierung unterschiedlich

Aufruf einer entfernten Prozedur

- Für Konversion u. Serialisierung der Parameter und des Prozeduraufrufs beim Server ist Zusatz-Software bei Client und Server erforderlich
- => „Client Stub“ und „Server Stub“ (*stub = Stummel*)
- Prozedurfernaufruf mündet zunächst in einen Aufruf des lokalen Client Stubs
- Client Stub initiiert dann den Fernaufruf
- Client Stub kommuniziert dazu mit dem Server Stub auf der Gegenseite durch Transfer der konvertierten und serialisierten Parameter
- Server Stub de-serialisiert die Parameter (unmarshalling) und ruft die gewünschte Prozedur auf dem Server als lokale Prozedur auf

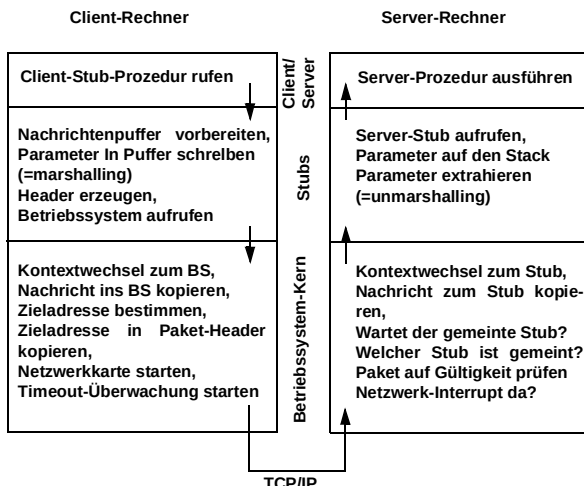
Ablauf bei Fernaufruf



Ablauf beim Fernaufruf

1. Der Client ruft die lokale Stub-Prozedur auf (*stub: Stummel*). Der Stub verpackt die Parameter als Nachricht (marshalling, *to marshall: anreihen*)
2. Netzwerkfunktionen des Betriebssystems zum Nachrichtentransfer werden vom Client-Stub aufgerufen (dazu Prozess-Kontextwechsel und Socket-Kommunikation)
3. Das Betriebssystem verschickt die Nachricht über eine verbindungslose oder verbindungsorientierte Kommunikation, d.h. über UDP oder TCP als Transportprotokoll
4. Der Server erhält einen Interrupt, der passende Server-Stub wird in der Liste der wartenden Prozesse herausgesucht und aktiviert. Danach entpackt der Server-Stub die Parameter (unmarshalling).
5. Der Server ruft die lokale Prozedur mit den empfangenen Parametern auf.
6. Nach Beendigung der Prozedur erhält der Server-Stub erneut die Steuerung über den Programmlauf.
7. Der Server-Stub verpackt die Ergebnisse als Nachricht.
8. Verschicken der Nachricht zum Client.
9. Empfangen der Nachricht vom Client.
10. Entpacken der Nachricht durch die Stub-Prozedur des Clients, empfangene Werte werden dem Anwendungsprogramm zur Verfügung gestellt.

Zusammenfassung des RPC-Ablaufs



Nicht-blockierende Fernaufrufe

- Fernaufrufe werden direkt im Sprachumfang um einen asynchronen Aufruf erweitert

- Dafür gibt es 3 Konzepte:

- Callbacks
- Futures
- Promises

Callback (Rückruf)

- Der Client setzt einen spez. nicht-blockierenden Fernaufruf ab und setzt sein Programm fort
- Der Server ruft den Client zurück, sobald das Ergebnis fertig berechnet ist.
- Der Rückruf wird durch ein Software-Interrupt (Event oder Signal) beim Client implementiert
- Das Ergebnis kann dann vom Client aus einem Resultat-Puffer gelesen werden

Futures

- Es wird sofort ein "provisorisches" Ergebnis (=Future) zurückgeliefert, so daß keine Blockierung stattfindet
- der Future ist eine spez. Variable, die als Platzhalter für das tatsächliche Ergebnis steht

- Der Datentyp des Futures entspricht dem Datentyp des Ergebnisses
- Der Client kann den Future bei Bedarf abfragen, ob die Berechnung fertig ist
- Sobald das Resultat berechnet wurde, wird es im Future gespeichert. Der Client kann es dann verwenden
- Mehrere Futures können zu einer Menge zusammengefaßt werden (Futureset)
- Das Futureset kann abgefragt werden, ob eines der Elemente ein Resultat hat

Promises (Versprechen)

- Promises sind Erweiterungen von Futures:
- Mehrere asynchrone Aufrufe werden gesammelt und als Block übermittelt, um die Effizienz der Übertragung zu erhöhen
- Die zeitliche Ordnung zwischen asynchronen Aufrufen und den Resultaten bleibt erhalten

Grundsätzliche Probleme von RPCs

1.) Mehr Fehlerquellen als bei lokalen Prozeduren:

- Aufgrund der meistens verwendeten „at-most-once-Semantik“ bleiben Aufrufe ab und zu ohne Ergebnis (at-most heisst, dass jeder Fernaufruf beim Server im Fehlerfall höchstens einmal ausgeführt wird)

=> Überwachung der RPC-Ergebnisse durch Timeout notwendig

- Bei Ausfall des Client bleiben beim Server verwaiste Aufrufe übrig

2.) Einschränkungen bei der Parameterübergabe:

- Zeiger sind problematisch, da Client und Server keinen gemeinsamen Arbeitsspeicher haben

=> Zeigerreferenzen werden dereferenziert, linearisiert, über das Netz geschickt, auf der Gegenseite wiederhergestellt und durch eine neue Zeigeradresse referenziert

3.) Einschränkungen bei der Programmierung verteilter Anwendungen:

- Kommunikation über gemeinsame Variable geht nicht, da globale Variable nicht gleichzeitig bei Client und Server zur Verfügung stehen

Welche Fernaufrufe gibt es in einem Netz?

- Die von Servern angebotenen Fernaufrufe werden i.d.R. im Netz einer zentralen Instanz (Binder) bekannt gemacht
- Zur Beschreibung eines Fernaufrufs ist mindestens die Angabe von Namen und Aufrufparameter notwendig
- Über spez. „Schnittstellenbeschreibungssprachen“ wird die sog. Signatur jeder Prozedur spezifiziert
- Die Signatur einer Prozedur besteht aus:

1.) Namen der Prozedur

2.) Reihenfolge und Typ der Parameter

3.) Vorschriften zum Verhalten im Fehlerfall (exceptions)

- Mit Hilfe einer Schnittstellenbeschreibungssprache (Interface Definition language, IDL) können die Client und Server Stubs automatisch erzeugt werden!

- Spezielle RPC-Compiler für die IDL erzeugen den Code der Stubs (C Source Code oder Object Code)

Beispiele von Schnittstellenbeschreibungssprachen

- ISO (International Standardization Organization): ASN.1 (Abstract Syntax Notation No. 1)
- SUN: XDR (External Data Representation)
- OSF (Open Systems Foundation): DCE IDL (Distributed Computing Environment Interface Definition Language)

DCE IDL

- DCE IDL ist eine deklarative Schnittstellenbeschreibungssprache (i.e. nicht prozedural)

- Erinnert in der Syntax an die Datendefinitionen in C (typedef, struct, ...)

- Erlaubt, alle Prozeduren, die ein Server zur Verfügung stellt, hinsichtlich ihrer Signatur zu beschreiben

- Definiert nur die Signaturen, nicht die Implementierung der Prozeduren

- Liefert zusätzliche Informationen für die automatische Stub-Code-Erzeugung

- Die IDL-Beschreibung der Menge aller Prozeduren eines Servers heißt IDL Interface

- Jedes IDL Interface enthält:

- einen Schnittstellennamen
- eine Schnittstellenadresse
- die Versionsnr. der Schnittstelle
- die Definition mindestens einer Prozedur

Aufbau eines DCE IDL Interfaces

```
[
uuid(<128-Bit-Schnittstellenadresse>),
version(<Versionsnr. der Schnittstelle>)
]
interface <Schnittstellenname>
{
<Prozedurdefinition 1>
<Prozedurdefinition 2>
...
<Prozedurdefinition n>
}
```

- Die uuid ist eine Kennung zur netzweit eindeutigen Identifikation der Serverschnittstelle (uuid = universal unique identifier)
- uuid wird von einem speziellen Hilfsprogramm erzeugt, dem uuid-Generator (uuidgen)
- uuidgen verwendet zur uuid-Erzeugung, Datum, Uhrzeit und Rechnername des Servers
- Der Schnittstellenname braucht nur auf dem Server-Rechner eindeutig sein, die uuid hingegen netzweit
- Jede Prozedurdefinition unterscheidet nach Eingabe-/Ausgabe- und gemischten Parametern (in, out, inout)

```
String header;           // Dokumentkopf
char *data;              // Dokumentdaten=eigentliches Dokument
}
Document                // Dokument
//Prozedurdefinitionen

[idempotent] long DocumentQuery
(
    // Anfrage nach Dokumenten
    [in] String documentName[maxDoc], // Dokumentnamen
    [out] DocumentDescription *dd[maxDoc], // Produktbeschreibungen
    [out] long *status                // Statuswert der Operation
);
long insertDocument
(
    // Einfügen eines Dokuments in Speicher
    [in] Document *d, // Neues Dokument
    [out] long *status // Statuswert der Operation
);
long removeDocument
(
    // Löschen eines Dokuments
    [in] String name, // Dokumentname
    [out] long *status // Statuswert der Operation
);
long fetchDocument
(
    // Beschaffen eines Dokuments
    [in] DocumentDescription *dd, // Dokumentbeschreibung
    [out] Document *d            // Gesuchtes Dokument
);
}
```

Beispiel:

DCE RPC-Schnittstellenbeschreibung zur Verwaltung von Textdokumenten:

- Zuerst wird per IDL je ein Datentyp zur Dokumentbeschreibung bzw. zum Dokument selbst definiert
- Danach werden Prozeduren zum Beschaffen und Löschen von Dokumenten sowie zum Einfügen von Textpassagen per IDL definiert

Bemerkung:

- Das IDL-Attribut "idempotent" heißt, daß die Prozedur mehrfach ausgeführt werden kann, ohne daß das Textdokument verändert wird

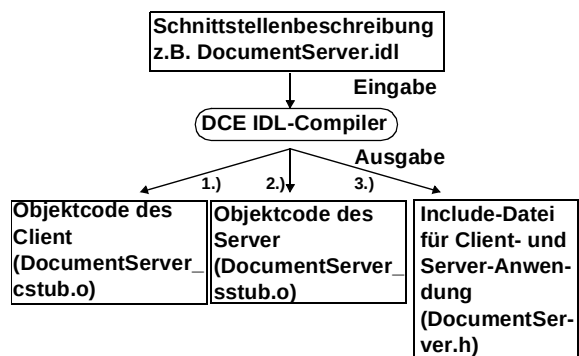
DCE RPC IDL-Beispiel

```
[ uuid(765c3b10-100a-135d-1568-040034e6783 1), version(1.0) ]
interface DocumentServer // Schnittstelle für Dokumenten-Server
{
import "globaldef.idl"; // Import allgemeiner IDL-Definitionen
const long maxDoc = 10; // Maximale Dokumentzahl in Anfragen
typedef [string] char *String; // Datentyp für Character-Strings
typedef struct
{
    String documentName; // Dokumentname
    String documentDescription; // Textuelle Beschreibung
    String keywords; // Schlüsselwörter
    long size; // Speicherumfang
}
DocumentDescription // Dokumentbeschreibung
typedef struct
{
    DocumentDescription desc; // Dokumentbeschreibung
}
```

Automatische Stub-Code-Erzeugung

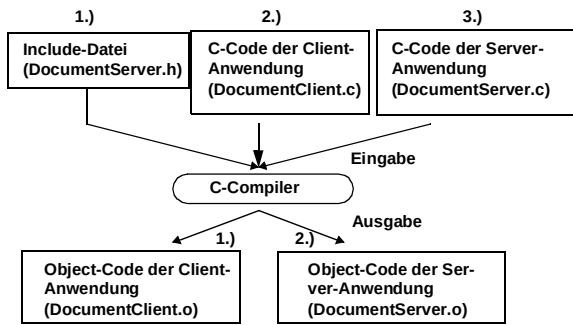
- Der Client- und Server Stub wird automatisch von einem spez. Compiler aus der Schnittstellenbeschreibung erzeugt, dem RPC-Compiler
- Bei DCE RPCs wird das Objektprogramm, bei SUN RPCs der C-Code automatisch erzeugt

Stub-Code-Erzeugung bei DCE RPCs

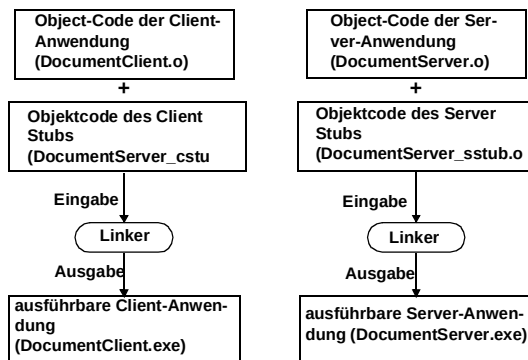


- Die erzeugten Dateien werden inkludiert bzw. vom Linker zu den Anwendungsprogrammen des Client und Server hinzugebunden

Automatische Include-Datei



- Zuletzt werden vom Linker die lauffähigen Programme erzeugt



```

beparameter */
program ADD_PROG /* Definition des Schnittstellennamens auf dem Server */
{
version ADD_VERS /* Definition der Version */
{
result ADD (request) = 1; /* Definition der Additionsfunktion */
}= 1; /* Version 1 der Funktion */
}= 20000000; /* Programmnummer zur netzweit eindeutigen Identifizierung der Additionsfunktion */
}

```

SUN RPCs

- Die von Sun entwickelten RPCs setzen sich im wesentlichen aus den folgenden Bestandteilen zusammen:

- 1.) Der betriebssystemunabhängigen Datendarstellung „XDR“ (= external data representation) für den Datentransfer zwischen den beteiligten Rechnern
- 2.) Einer sog. XDR-Filter-Bibliothek, welche die auf der lokalen Maschine benutzten Datendarstellung nach XDR konvertiert und vice-versa
- 3.) Das Programm „rpcgen“, das aus einer RRC-Schnittstellenbeschreibung den C-Quellcode für Client- und Server-Stub erzeugt erzeugt.
- 4.) Einer Bibliothek mit RPC-Dienstfunktionen.
- 5.) Zwei Verwaltungsinstanzen („Portmapper“ und „rpcbind“) für die zur Verfügung stehenden RPC-Aufrufe, die die auf einer Maschine registrierten RPC-Dienste und deren TCP/UDP-Port-Adressen verwaltet

Beispiel einer SUN XDR-Schnittstellenbeschreibung zur Addition zweier Zahlen

- Es sollen zwei Integerwerte *a*, *b* durch einen Server addiert werden

- Der Inhalt der XDR-Datei (=Protokollspezifikation für die Funktion `add.x`) sieht folgendermassen aus:

```

struct result { int x; }; /* Return Parameter */
struct request { int a; int b; }; /* 1 Ueberga-

```

Wie erfährt ein Client, welcher Server die gewünschte Fernprozedur ausführt?

- Die Information, wo der Server zu einer Fernprozedur zu finden ist, erhält der Client durch „verteiltes Binden“
- Durch verteiltes Binden entsteht eine Zuordnung (=Bindung) zwischen Client-Prozeduraufruf und Server
- Sinn des verteilten Binden ist, daß der Client erfährt, welcher Server im Netz die benötigten Dienste erbringt
- Nach dem Binden kommunizieren Client und Server direkt miteinander (verbindungslos oder verbindungsorientiert)
- Durch verteiltes Binden erfährt der Client:
 - Das Protokoll, das auf dem Netz verwendet wird (TCP oder UDP)
 - Den symbolischen DNS-Namen bzw. die IP-Adresse des Servers
 - Die sog. Port-Adresse jeder Prozedur des Servers
- Bemerkung: Die TCP- bzw. UDP-Port-Adresse ist zur Abwicklung des TCP- bzw. UDP-Protokolls nötig
- Im einfachsten Fall kennt der Client bereits a priori die Netz- und Port-Adresse des Servers
- Im Allgemeinfall erfährt der Client die Port-Adresse durch das verteilte Binden

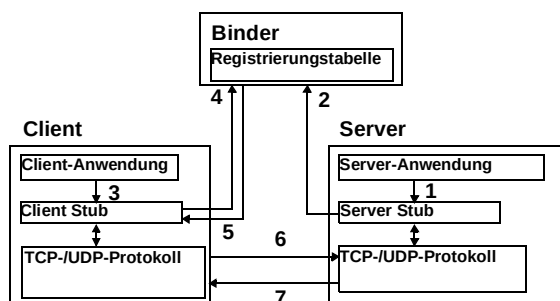
- I.d.R. wird der richtige Server dem Client anhand dessen Schnittstellenbeschreibung durch das Binden mitgeteilt
- Für die Implementierung des verteilten Bindens gibt es 2 Möglichkeiten:
- Fest programmierte Serveradresse (statisch)
- Verwendung eines "Binders" (dynamisch)

Fest programmierte Serveradresse

- Serveradresse ist im Code des Client verankert
- Client und Server werden zur Compile-Zeit aneinander gebunden
- Nachteil: Neukodierung und Neukompilation notwendig, sobald sich etwas ändert

Verwendung eines Binders

- Der Server macht seine Prozeduren bei einer dritten Instanz, dem Binder, bekannt
- Das Bekanntmachen beim Binder heißt auch „Registrieren“ bzw. „Exportieren“ der Schnittstellenbeschreibung
- Der Binder ist eine zentrale Software-Instanz, die in jedem lokalen Netz mindestens einmal vorhanden sein muß (bei RPCs ohne fest kodierte Server-Adressen)



- Die Registrierungstabelle des Binders enthält folgende Tripel:

(DNS-Server-Name, Schnittstellenbeschreibung, TCP-/UDP-Port)

Wann wird TCP, wann UDP als Protokoll eingesetzt?

- TCP ist verbindungsorientiert und zuverlässig, aber kostet Zeit für den Verbindungsaufbau und für Paketwiederholungen im Fehlerfall
- UDP verbraucht keine Zeit für Verbindungsauf-/abbau, aber garantiert keine sichere Übertragung

=> TCP ist bei Netzen mit höheren Fehlerraten (MAN und WAN) anzuraten

- Der Client wiederum fragt den Binder nach einem Server mit entsprechender Schnittstellenbeschreibung (=Importieren einer Serveradresse)
- Die Vergabe der richtigen Server-Adresse erfolgt dynamisch zu Beginn des Programmablaufs des Klienten oder sogar erst bei dessen erstem Fernaufruf
- Vorteil: Änderungen der Server-Adresse, Umschalten auf Backup-Server oder Systemumkonfigurationen bleiben dem Klienten verborgen

Ablauf beim Binden

- 1.) Server ruft im Server Stub eine Registrierungsprozedur auf
 - 2.) Server Stub nimmt beim Binder Registrierung der eigenen Schnittstelle vor
 - 3.) Client ruft im Client Stub eine Anfrageprozedur auf
 - 4.) Client Stub fragt beim Binder nach einem Server mit passender Schnittstellenbeschreibung
 - 5.) Client Stub erhält gewünschte Netzadresse zusammen mit einer Port-Nr. des Servers, dem sog. Handle (*Handle: Griff*)
- Danach kommunizieren Client und Server direkt:
- 6.) Client Stub schickt TCP-/UDP-Paket zum Server Stub, das Prozedurnamen und Aufrufparameter enthält
 - 7.) Server Stub antwortet mit TCP-/UDP-Paket, das das Ergebnis des Aufrufs enthält

=> UDP kann bei zuverlässigen LANs eingesetzt werden

- Des weiteren ist UDP bei dynamischem Binden, das unmittelbar vor dem Fernaufruf erfolgt, vorzuziehen, da Vorlaufzeit zum Verbindungsaufbau wegfällt

- Bei statischem Binden oder dynamischem Binden zu Beginn der Client-Programmausführung spielt die Zeit zum Verbindungsauf-/abbau keine Rolle => TCP

Fehler bei Fernaufrufen

- Es gibt verschiedene Möglichkeiten für das Ausreten eines Fehlers:

- 1.) Absturz eines beteiligten Rechners (Client oder Server)
 - ist besonders kritisch, wenn der Ausfall während der Ausführung einer Operation auftritt
 - zwei grundlegende Anforderungen dabei: a) kein endloses Warten des Clients auf einen abgestürzten Server b) keine verwaisten Aufrufbearbeitungen nach abgestürztem Client
- 2.) Unerreichbarkeit des Zielknotens (= Absturz des Rechnernetzes oder der Netzverbindung)
 - Handhabung durch sicheres Transportprotokoll und die RPC-Laufzeitumgebung (Wiederholung bei Übertragungsfehler)
 - Reaktion auf RPC-Ebene: Binden an neuen Server
- 3.) Eingestellter Export einer Prozedur (= Prozedur gibt es nicht mehr)
 - z.B. aufgrund eines Konfigurationswechsels im Server-System
 - Reaktion auf RPC-Ebene: Binden an neuen Server

RPC-Fehlersemantik

- Es gibt verschiedene Möglichkeiten, wie auf eine Fehlersituation reagiert werden kann
- Je nach Erfordernissen der Anwendung ist eine der unteren Strategien zu wählen:
- **Maybe:** höchstens einmalige Ausführung einer Fernprozedur ohne Benachrichtigung im Fehlerfall
 - ist sehr einfache Strategie mit unbefriedigenden Resultaten
- **At-Least-Once:** mindestens einmalige Ausführung
 - Tatsächliche Anzahl der Ausführungen ist unbekannt (0/1)
 - ist nur bei idempotenten Operationen sinnvoll (gibt es in der Praxis selten)
- **Exactly-Once:** Ausführung genau einmal
 - Transaktionskonzept mit Wiederanlauf (=Restart) von Software-Komponenten erforderlich
 - Sehr aufwendig, da Datenbankeigenschaft „ACID“ erforderlich
- **At-Most-Once:** Ausführung ganz oder gar nicht (=atomare Strategie)
 - garantiert, daß keine Auswirkungen im Fehlerfall entstehen
 - wird meistens in der Praxis eingesetzt, da guter Kompromiss zwischen Aufwand und Wirkung

RPC-Fehlersemantik (Zusammenfassung)

Fehlersemantiken	Fehlerarten			
	Fehlerfreier Ablauf	Nachrichterverluste	Zusätzlicher Server-Ausfall	Zusätzlicher Client-Ausfall
May Be	Ausführung: 1x Ergebnis: 1x	Ausführung: 0 oder 1x Ergebnis: 0 oder 1x	Ausführung: 0 oder 1x Ergebnis: 0 oder 1x	Ausführung: 0 oder 1x Ergebnis: 0 oder 1x
At-Least-Once	Ausführung: 1x Ergebnis: 1x	Ausführung: $\geq 1x$ Ergebnis: $\geq 1x$	Ausführung: $\geq 0x$ Ergebnis: $\geq 0x$	Ausführung: $\geq 0x$ Ergebnis: 0x
At-Most-Once	Ausführung: 1x Ergebnis: 1x	Ausführung: 1x Ergebnis: 1x	Ausführung: 0 oder 1x Ergebnis: 0 oder 1x	Ausführung: 0 oder 1x Ergebnis: 0x
Exactly Once	Ausführung: 1x Ergebnis: 1x	Ausführung: 1x Ergebnis: 1x	Ausführung: 1x Ergebnis: 1x	Ausführung: 1x Ergebnis: 1x

- Bei den meisten RPC-Systemen liegt die At-most-once-Semantik zugrunde, da Exactly-Once zu aufwendig in der Implementierung ist und die anderen Semantiken zu indeterministisch sind

Typische Leistungsdaten von DCE RPCs

(gemessen zwischen DEC Alpha 3000 AXP 700 Workstations)

Art des RPCs	Mittlere Dauer [ms]
RPC mit Rückantwort (ohne Parameter)	2,0
Asynchroner RPC mit Thread-Erzeugung und Rückantwort (ohne Parameter)	2,2
RPC mit 1 in- und 1 out-Parameter von 20 Byte	4,3
RPC mit in-Parameter von 1 KByte	6,9
RPC mit in-Parameter von 10 KByte	20
RPC mit in-Parameter von 100 KByte	60
Exportieren eines Servers an den Directory Service (=Binder)	300
Importieren eines Servers vom Directory Service	200

=> RPCs sind nicht besonders schnell

Effizienzsteigerungen bei RPCs

1.) Problem der Effizienz:

- Bei RPCs ist die Antwortzeit für einen Fernaufruf wichtiger als die Datenrate bei der Parameterübergabe, da nur wenige Parameter (=wenige Daten) übergeben werden
 - Bei TCP/UDP wird hingegen die Datenrate auf Kosten der Latenz optimiert, da die langen Header der TCP/UDP-Pakete sich nur bei großen Paketlängen rentieren
- => es wurden spez. Transportprotokolle für RPCs entwickelt (werden allerdings in der Praxis nur selten eingesetzt)

2.) Problem der Effizienz:

Je nach RPC-Implementierung werden die Parameter mehr oder weniger oft kopiert, bevor sie übertragen werden

=> Ziel: Minimierung der Kopiervorgänge

3.) Problem der Effizienz:

Wird ein Fernaufruf rekursiv verwendet, entsteht ein Blockadesituation, da ein Server sich nicht selbst aufrufen kann

- 1. Verwandtes Problem:

Trifft ein zweiter Fernaufruf bei einem Server ein, während dieser noch einen ersten Aufruf bearbeitet, muß der zweite warten

- 2. Verwandtes Problem:

Durch den blockierenden (=„synchronen“) Charakter der Fernaufrufe kann keine Parallelverarbeitung zwischen Client und Server entstehen

- 3 alternative Lösungsmöglichkeiten:

- Verwendung von sog. Vater- und Kindprozessen beim Client und/oder Server
- Verwendung von „leichtgewichtigen Prozessen“ (Threads) beim Client und/oder Server
- Erweiterung auf asynchronen (nicht-blockierenden) Fernaufruf

Bemerkung: Child-Prozesse bzw. Threads sind sehr allgemeine Methoden zur Effizienzsteigerung und können auf vielen Gebieten gewinnbringend eingesetzt werden.

4.) Problem der Effizienz:

RPCs sind zeitaufwendig, da ihre Implementierung zahlreiche Befehle erfordert

Verminderung der Kopienzahl zur Milderung des Effizienzproblems bei RPCs

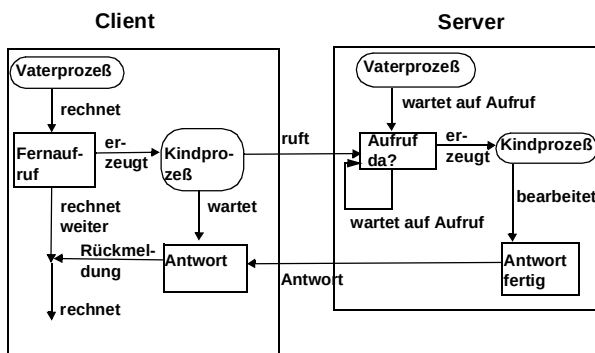
- Neben des Overheads durch Kontextwechsel, Datentransfer, Prüfsummenberechnung etc. stellen die benötigten Kopiervorgänge einen bremsenden Faktor dar.
- Die Anzahl Kopiervorgänge ist abhängig von der Unterstützung durch Hardware und Betriebssystem:

1.) Best case:

- Nur 1 Kopie: Netzwerkkarte des Klienten kopiert Daten per DMA (Direct Memory Access) direkt aus dem Adressraum des Clients auf das Netzwerkkabel
- Auf Server-Seite werden die Daten von der Netzwerkkarte direkt in den Adressraum des Betriebssystems kopiert und von dort von der MMU (Memory Management Unit) in den Adressraum des Servers eingeblendet (=> kein 2. Kopieren)
- Ohne MMU ist ein 2. Kopieren erforderlich.

2.) Worst case:

- 5 Kopien nötig: Client-Stub -> Client-BS -> Client-Netzwerkkarte -> Server-Netzwerkkarte -> Server-BS -> Server-Stub



- Nachteile:

- Durch Erzeugung der Kindprozesse und dem Time Sharing zwischen Vater- und Kindprozessen entstehen Zeitverluste
- Kommunikation zwischen Vater- und Kindprozess aufwendig, da beide in getrennten Adressräumen ablaufen

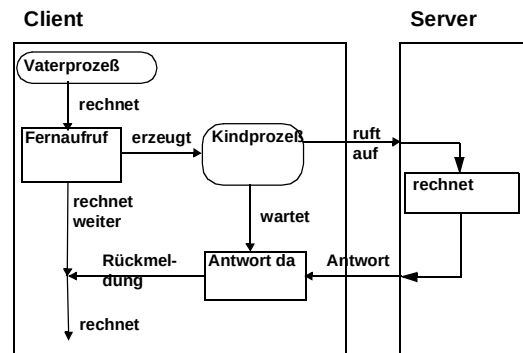
Leichtgewichtige Prozesse und RPCs

- Leichtgewichtige Prozesse (Threads) können im selben Adressraum ablaufen

=> die Umschaltzeit von Thread zu Thread ist sehr kurz, da kein Kontextwechsel nötig

Steigerung der Effizienz bei RPCs durch Vater- und Kindprozesse

- Der Vaterprozess des Client (Hauptprozess) erzeugt für jeden Fernaufruf einen neuen Kindprozess, der auf die Rückantwort wartet



=> Entkopplung des Fernaufrufs und seine Ausführung vom aufrufenden Hauptprogramm

=> Parallelverarbeitung durch Client und Server

- Zur weiteren Effizienzsteigerung kann der Vaterprozess des Servers für jeden eingehenden Aufruf einen neuen Kindprozess erzeugen, der dann den Aufruf bearbeitet

=> Server ist auch dann nicht blockiert, wenn er bereits einen Fernaufruf bearbeitet

=> Kommunikation zwischen Threads ist einfacher und schneller als zwischen Prozessen

- Kommunikation erfolgt über gemeinsame Variable zweier oder mehrerer Threads

- Nebenläufigkeit ist durch zeitliche Verzahnung der Threads hinsichtlich ihrer Bearbeitung gewährleistet.

- Threads werden innerhalb der Zeitscheibe, die dem Server-Prozess zur Verfügung steht, reihum abgearbeitet

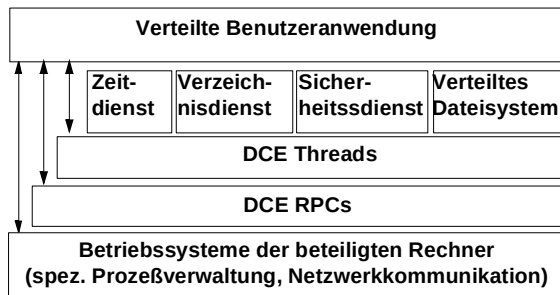
=> „Multithreading“

- durch Multithreading werden alle Nachteile von Vater-/ Kindprozessen umgangen

- Multithreading ist sehr gute Methode zur Effizienzsteigerung

DCE Middleware

- Das Distributed Computing Environment (DCE) wurde vom OSF-Industriekonsortium spezifiziert und standardisiert
- Dient als Middleware, um verteilte Anwendungen leichter zu erstellen
- Hauptaufgabe: Verbergen der Heterogenität der Betriebssysteme und Rechnerarchitekturen
- Basiert auf RPCs => prozedurales Programmiermodell
- Es gibt auch einfache objektorientierte Erweiterungen
- Arbeitet nach dem Client/Server-Konzept
- DCE-Referenzmodell



- DCE sitzt auf dem Betriebssystem des jeweiligen Rechners auf
- Darauf werden eigene sog. leichtgewichtige Prozesse und Prozedurfernaufrufe definiert
- Diese werden vom Zeitdienst, Verzeichnisdienst, Sicherheitsdienst und verteiltem Dateisystem benutzt
- Die Benutzeranwendung kann sowohl die Dienste der obersten DCE-Schicht als auch alle darunterliegenden verwenden

=> kein klares Schichtenmodell bei DCE

- Verteilte Anwendungen werden räumlich gegliedert
- Die Gliederungseinheiten heißen Zellen
- Zellen sind Verwaltungseinheiten für Benutzer und Rechner ähnlich wie Windows-Domänen

Zeitdienst (Time Service)

- Der Zeitdienst stellt allen Rechnern in DCE eine einheitliche Uhrzeit zur Verfügung
- Verteilt die Uhrzeit dreistufig: Zeit im DCE-Verbund, Zeit in jeder Zelle und Zeit im einzelnen Rechner

Verzeichnisdienst (Directory Service)

- Erlaubt DCE-weit eindeutige Datenamen zu vergeben (über Zell- und Rechnergrenzen hinweg)

- Verwaltet die Namen hierarchisch in zwei Verzeichnissen: innerhalb der Zellen und im DCE-Verbund
- Erlaubt zusammen mit dem Internet-DNS die Zuordnung von Rechnernamen auf IP-Adressen
- Nutzt Cache-Dateien, um frühere Anfrageergebnisse schnell wiederverwendbar zu machen
- Erlaubt redundante Verzeichnisse auf Backup-Systemen

Sicherheitsdienst (Security Service)

- Prüft die Identität der Kommunikationspartner über Passwort
- Schützt vor Abhören und Dateimanipulation durch Verschlüsselung
- Ist konform zu Standards wie DES. Jeder Client hat eigenen Schlüssel zum Chiffrieren
- Prüft die Zugriffsrechte anhand von Zugriffslisten (Access Control Lists) für jede Resource
- Es gibt 5 Sicherheitsklassen: von "nur einmal authentisieren" bis "alle Kommunikation verschlüsseln"

Verteiltes Dateisystem (DFS)

- Basiert auf dem Andrew File System (AFS; = Nachfolger von NFS)

- Die Verteilung der Dateien auf die DCE-Rechner bleibt den Benutzern verborgen
- Ebenso die Heterogenität der verschiedenen Dateisysteme
- Nutzt den Sicherheitsdienst als Zugriffsschutz
- Gleichzeitige Mehrfachzugriffe auf eine Datei werden unterstützt (Session-Semantik)
- Zur Geschwindigkeitssteigerung werden Dateien in Dateicaches gestellt und/oder repliziert (=vervielfacht)

- Unterstützt Sicherheits-Backups von Benutzerdateien

DCE-Erweiterung für Objektorientiertes Programmieren

- Unterstützt verteilte Objekte für C++
- Erzeugen, Löschen und Aufrufen entfernter Objekte
- Vererbung von Objektmethoden möglich

CORBA Middleware

- Common Object Request Broker Architecture wurde vom OMG-Industriekonsortium standardisiert
- Dient ebenso wie DCE als Middleware, um verteilte Anwendungen leichter zu erstellen
- Basiert im Gegensatz zu DCE auf Methodenfernaufrufen

=> objektorientiertes Programmiermodell

- Arbeitet nach dem Client/Server-Konzept
- Unterstützt viele Sprachen (C/C++, Java, Smalltalk, ADA, FORTRAN, Cobol, ...)
- Implementierung teilweise auch über DCE
- Einige Vorteile von CORBA gegenüber DCE:
 - Unterstützt wg. Objektorientierung Vererbung und Polymorphismus => Wiederverwendbarkeit von Software
 - Hat sowohl synchrone als auch asynchrone Methodenfernaufrufe
- Nachteile:
 - Objekte werden durch Prozesse implementiert => grobgranular im Vergleich zu threads
 - Objekte bleiben einem Server fest zugeordnet, auch wenn sie von einem weit entfernten Client gebraucht werden => keine Objektmigration

- Alle Kommunikation zwischen Klienten und Servern wird über den ORB abgewickelt.
- Objekte, die Methoden zum Aufruf anbieten, müssen sich, wie zuvor die Prozeduren beim Binder von RPCs, beim ORB registrieren lassen
- Die ORB wiederum erzeugt mit Hilfe der IDL-Beschreibung der Methoden und eines Stub-Compilers einen Server Skeleton
- Will ein Client eine entfernte Methode aufrufen, beschreibt er diese dem ORB ebenfalls in IDL. Der ORB erzeugt daraus mit Hilfe des Stub-Compilers einen Client-Stub
- Stub bzw. Server Skeleton bzw. die Methoden, die sie aufrufen, werden statisch zum Client- bzw. Server-Code gebunden.
- Alternativ kann auch erst zur Laufzeit dynamisch gebunden werden, was Polymorphismus erlaubt
- In beiden Fällen geht jeder nachfolgende Methodenaufruf, über den ORB. Dieser bietet dazu sog. Objekt-Adapter an

Dynamisches Binden

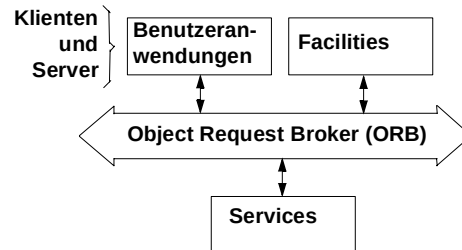
- Zum dynamischen Binden vergleicht der ORB zur Laufzeit die IDL-Methodenbeschreibung, die der Klient liefert, mit den IDL-Methodenbeschreibungen, die die Server anbieten
- Dann wählt er eine passende Server-Methode aus.

Aufbau von CORBA

- CORBA-Systeme bestehen aus vier Komponenten:

- 1.) Den verteilten Benutzeranwendungen, die nach Client und Server gegliedert sind
- 2.) Dem sog. Object Request Broker (ORB)
- 3.) Den CORBA facilities
- 4.) Den CORBA services

- CORBA-Referenzmodell



- Der ORB ist wie ein "Software-Bus", in den andere Komponenten eingehängt werden können

Object Request Broker (ORB)

- Dient als Bindeglied (Software-Bus) zwischen Klienten, Servern, CORBA facilities und CORBA services.

- Die IDL-Methodenbeschreibungen werden vom ORB in einem speziellen Verzeichnis gehalten (repository)

Methoden-Fernaufwurf

- Je nachdem, ob die entfernte Methode statisch oder dynamisch an den Klienten gebunden wurde, sind 2 verschiedene Aufrufsemantiken möglich:

- 1.) nur synchroner (blockierender) Aufruf bei statischer Bindung
- 2.) synchroner oder asynchroner Aufruf bei dynamischer Bindung.

- Zusätzlich kann bei beiden Arten ein „Einwegaufruf“ gemacht werden, der keine Ergebnisparameter zurückliefert und asynchron ist

Client Stub

- Hat dieselbe Aufgabe wie bei RPCs (marshalling der parameter etc.)

Server Skeleton

- Entspricht einem Server Stub bei RPCs. Sorgt für demarshalling und Methodenaufruf

Objekt-Adapter

- Ist die Schnittstelle des Software-Busses ORB zu einem Server

- Jeder Methoden-Fernaufwurf von einem Client wird über den Objekt-Adapter des ORB an den passenden Server weitergeleitet
- Über den Objekt-Adapter werden auch die Server aktiviert, bevor sie ihren ersten Methoden-Fernaufwurf ausführen können

Server-Aktivierung

- Für die Aktivierung eines Servers gibt's 4 verschiedene Strategien:
- 1.) Der Server wird vom Objekt-Adapter aktiviert und bietet mehrere Inkarnationen von sich an. Jede Inkarnation kann gleichzeitig einen anderen Methoden-Fernaufwurf ausführen (= shared server).
- 2.) Der Server wird nicht vom Objekt-Adapter aktiviert, sondern bereits beim Booten (= persistent server)
- 3.) Der Server wird vom Objekt-Adapter aktiviert. Allerdings gibt es nur eine Inkarnation des Server-Codes. Nur ein Methoden-Fernaufwurf kann zu einer Zeit ausgeführt werden (= unshared server)
- 4.) Für jeden Methoden-Fernaufwurf wird eine Inkarnation eines Server-Codes vom Objekt-Adapter aktiviert und nach Ende des Methoden-Fernaufwurfs wieder deaktiviert (= server-per-method)

Inter-Orb-Protocol

- Dient zur Kommunikation zwischen verschiedenen ORBs. Bsp.: Internet-Inter-ORB-Protocol, IIOP.

(D)COM Middleware

- COM (Component Object Model) stammt von Microsoft
- COM dient zur Interaktion zwischen Software-Komponenten desselben Herstellers (Microsoft)
- Definiert ebenso wie ActiveX bzw. Java Beans Schnittstellen und Namenskonventionen für die Objekte
- COM verwendet DCOM als Kommunikationsbasis im Netzwerk
- COM unterstützt verteilte "Objekte", die nach dem Client-Server-Prinzip organisiert sind
- COM erlaubt Multithreading; ist RPC-basiert

Aufbau von COM

- COM verwendet Interfaces, Klassen und Instanzen
- Interfaces sind abstrakte Klassen, die nur Methodenköpfe enthalten. Sie werden über eine 128 Bit Interface-ID adressiert.
- Interfaces sind die COM-Gegenstücke zu Schnittstellenbeschreibungen bei RPCs
- Klassen sind Implementierungen von Interfaces
- Klassen werden über 128 Bit Klassen-ID angesprochen
- Instanzen sind Inkarnationen (=Vertreter) von Klassen

CORBA Services

- Enthalten allgemeine Dienste für die Anwendung. Bsp. sind:
- 1.) Namensdienst
- 2.) Ereignisdienst
- 3.) Sicherheitsdienst
- 4.) Zeitdienst
- 5.) Transaktionsdienst

CORBA Facilities

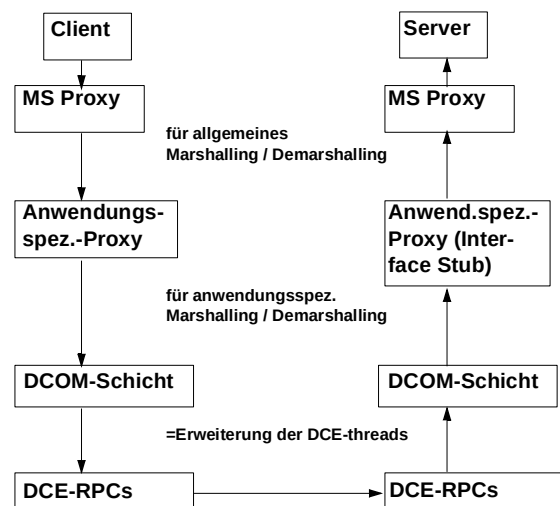
- Enthalten spezielle Dienste für die Anwendung. Bsp. sind:
- 1.) Benutzerschnittstellen
- 2.) Verwaltung der Anwendung

ODP (open distributed processing)

- Gegenstück zu DCE und CORBA von ISO/OSI. Gliedert die verteilte Anwendung hinsichtlich verschiedener Blickwinkel. Bsp. enterprise viewpoint, technical viewpoint

Bsp.: Winword ist ein Server mit einem bestimmten Interface. Alle MS-Word-Dokumente bilden eine Klasse. Mein eigenes Word-Dokument ist eine Instanz der Klasse

Aufbau von COM/DCOM



- Proxy, engl.: Stellvertreter. Proxies sind die COM-Gegenstücke zu RPC Stubs

Aufruf von COM

- COM spezifiziert eine Binärschnittstelle (Application Binary Interface, ABI) zum Client und Server

=> das Speicherlayout für den Methodenfernaufruf muß in einen bestimmten Format sein

Proxy von Microsoft

- Wird von MS bei COM mitgeliefert
- Ist ein Standard-Marshaller, der Zeiger auf Interfaces in Zeiger auf Klassen umsetzt und serialisiert

Proxy für den Client (= Interface Stub)

- Wird in der COM Interface Definition Language (COM IDL) definiert und dann automatisch erzeugt (über einen speziellen COM IDL Compiler)
- Macht auf der Client-Seite das restliche marshalling, das nach dem Standard-Marshaller noch übrig bleibt
- Enthält auf der Client-Seite die Definition der Methoden, die der Server anbietet

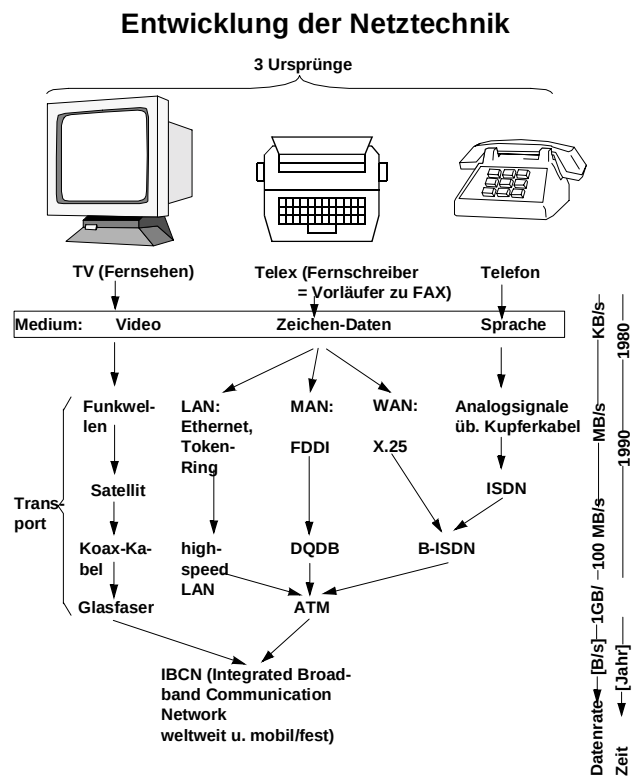
Proxy für den Server (= Interface Stub)

- Wird wie der Client Stub in COM IDL definiert und dann automatisch erzeugt
- Führt auf der Server-Seite das unmarshalling und den Methodenaufruf durch

DCOM-Schicht

- Ist eine Erweiterung der DCE RPCs
- Übernimmt von DCE RPCs das marshalling, die Sicherheitsmerkmale und die Prozedurfernaufrufe

Echtzeitübertragung im Internet



Internet-Echtzeitzugang für Multimedia

- Übertragung multimedialer Daten erfordert:
 - Bewahrung der zeitlichen Synchronisation zwischen Bild-, Ton-, Text- und Graphikdaten
 - Wiedergabe der Daten in Echtzeit
- => Datenübertragung muß mit Zeitbezügen untereinander und zur Realzeit erfolgen (=Echtzeitübertragung)
- => Die übliche Ethernet-LAN-Technik ist ungeeignet, da indeterministisch
- Geeignet sind Netztechniken mit garantierter Zustellzeit.
- alle Punkt-zu-Punkt-Netze sind prinzipiell geeignet
- alle Broadcast-Netze mit „deterministischem Zeitmultiplex“ sind geeignet

Beispiel Punkt-zu-Punkt-Netz: Telefonnetz

Beispiele deterministischer Zeitmultiplexnetze:

Token Ring, FDDI, PCM, S-ISDN, ATM

Internet-Echtzeitzugang über Telefonnetz

- Zugang zum Telefonnetz kann analog oder digital sein
- Analoger Zugang erfolgt über Modem

Internet-Echtzeitzugang über digitalen Telefonanschluss

- erfolgt mittels ISDN-Anschluss der Telekom oder anderer Betreiber
- nicht Gegenstand der Vorlesung Rechnernetze 2

Internet-Echtzeitübertragung über digitale Netze und Verfahren

- PCM-Verfahren (= Protokoll > Schicht 1a)
- Sonet/SDH - Netz (= Protokoll = Schicht 1)
- ATM-Verfahren (= Protokoll > Schicht 1a)

- Digitaler Zugang erfolgt über S-ISDN oder xDSL
S-ISDN = „Schmalband Integrated Services Digital Network“ = Schmalbandiger integrierter (Multimedia)-Service mittels digitalem Telefonnetz

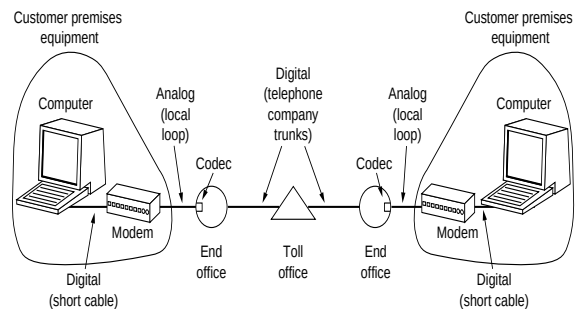
xDSL = ADSL oder SDSL

SDSL = Symmetric Digital Subscriber Line = Teilnehmeranschluß mit symmetrischen Baudraten für up-/download

ADSL = Asymmetric Digital Subscriber Line = Teilnehmeranschluß mit asymmetrischen Baudraten für up-/download

- Intern arbeitet Telefonnetz mittels digitaler PCM-Technik und überträgt in Echtzeit

Internet-Echtzeitzugang über analogen Telefonanschluß



- Codec erzeugt 8000 8-Bit-Worte pro Sek. => 64 Kb/s
- Zwischen Codecs werden Daten mittels PCM übertragen

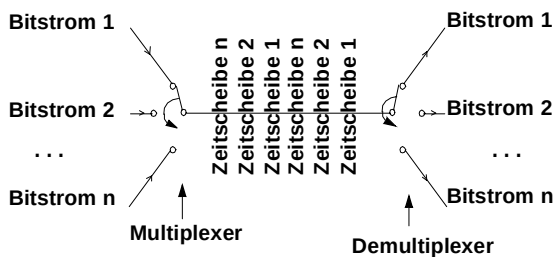
PCM-Verfahren zur Echtzeit-Datenübertragung

Teil 1

PCM-Übertragungstechnik (Teil 1)

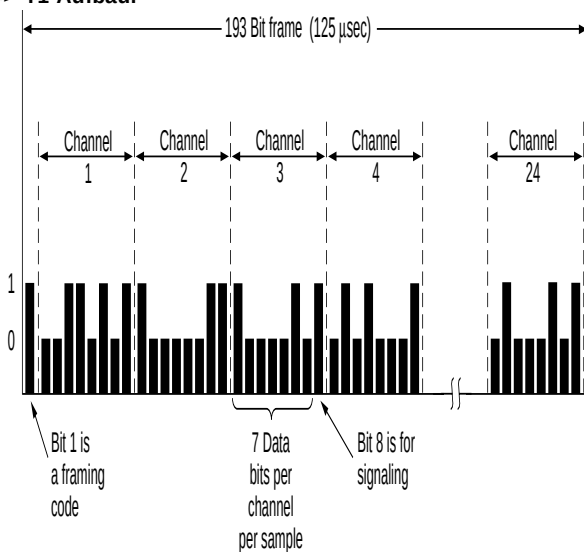
- PCM = „Pulsmodulation“ (eigentlich Codierung nicht Modulation)
- Dient zur Echtzeitübertragung digitaler Daten zwischen Telefonvermittlungsstellen
- Verwendet synchrones Zeitmultiplex:
- Binäre Worte verschiedener Datenquellen werden im Kreis abwechselnd nacheinander übertragen
- Jeder Quelle wird periodisch eine Zeitscheibe (time slot) zugeteilt.
- Multiplexer und Demultiplexer laufen synchron

Prinzip:



T1-PCM-System (USA und Japan)

- Periodendauer = 125 μ s bei 24 Kanälen zu je 64 Kb/s
- => Es gibt 24 Zeitscheiben zu je 8 Bit pro Periode
- Datenmenge pro Periode heißt „Rahmen“
- => Bruttodatenrate des Rahmens ist 1,544 Mb/s
- => T1-Aufbau:



PCM-Multiplexsysteme (1)

- Für jedes Zeitmultiplex-System sind Systemparameter festzulegen, z. B.:
- die kleinste Übertragungseinheit pro Zeitabschnitt (Bit, Byte, n-bit-Wort oder Block)
- die Häufigkeit der Zeitscheibenzuteilung an einen Übertragungskanal
- Synchronisierungshilfen zwischen Multiplexer und Demultiplexer
- Melde und Signalisierdaten (Zeichengabe)

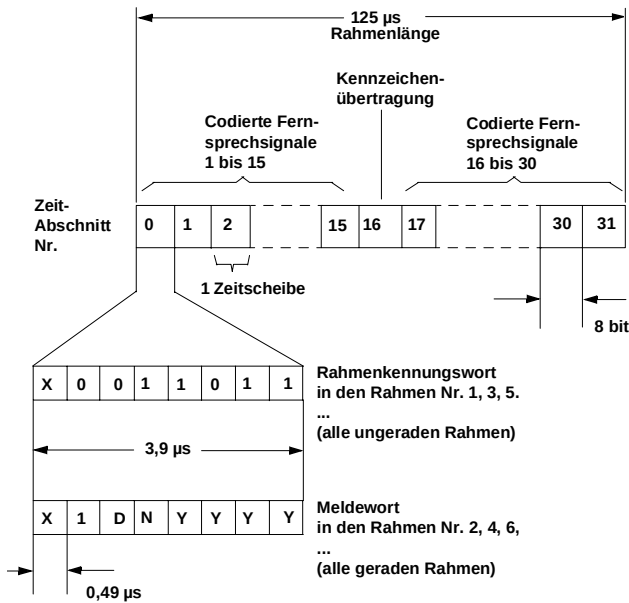
Struktur des genormten PCM in der Telefonie:

- pro Zeitscheibe: 8 Bits
- Übertragungszeit pro Kanal: ca. 3,9 μ s
- Verschachtelungsgrad (die Periode):
24 Kanäle (T1-PCM)
32 Kanäle (E1-PCM)

E1-PCM-System (Welt ohne USA und Japan)

- Periodendauer = 125 μ s, bei 32 Kanälen zu je 64 Kb/s
- => alle 125 μ s kommen 32 Zeitscheiben zu je 8 Bit
- => Bruttodatenrate 2,048 Mb/s
- 30 Zeitscheiben sind für Nutzdaten, 2 Zeitscheiben für Verwaltungsinformation (sog. Zeichengabe)

Pulsrahmen des Systems PCM 30 (=E1)



D Meldebit für dringenden Alarm
 N Meldebit für nicht dringenden Alarm
 X für internationale Verwendung reservierte Bits
 Y für nationale Verwendung reservierte Bits

E1-PCM-Multiplexsystem (2)

- Zeitabschnitte sind mit 0 bis 31 nummeriert
- Ein Abschnitt ist ca. 3,9 µs lang. Die gesamte Rahmendauer ist bei PCM30 mit 125 Mikrosekunden genormt
- Im Zeitabschnitt 0 werden abwechselnd Rahmenkennworte (u. a. zur Rahmenidentifizierung, Synchronisierung) und Meldeworte (u. a. zur Überwachung der Digitalsignalleitung) übertragen
- Der Kennzeichenabschnitt dient zur Übertragung vermittlungstechnischer Daten, wie Wählzeichen usw
- Die 30 übrigen Zeitabschnitte nehmen jeweils 8 Bits (einen Abtastwert) eines digitalen Fernsprechkanaals von 64kbit/s auf
- Anstelle von Fernsprechsignalen können auch beliebige andere digitalisierte Daten in Einheiten von 8 Bits über ein digitales PCM-System übertragen werden

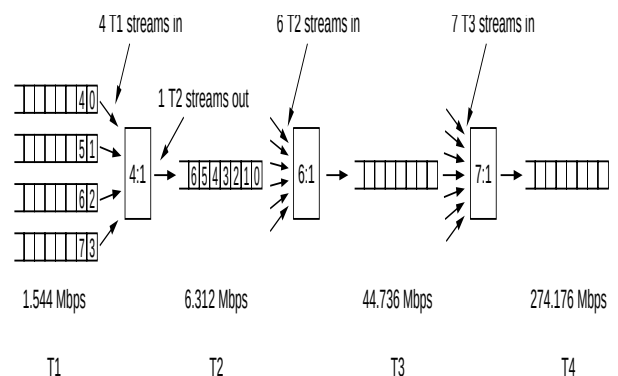
PCM-Verfahren zur Echtzeit-Datenübertragung

Teil 2

PCM-Hierarchien

- Sowohl E1- als auch T1-PCM-Systeme erlauben, mehrere Rahmen zu „Trägern“ höherer Baudraten zu multiplexen
- Beim Multiplexen werden mehrere vollständige Träger niedriger Datenrate incl. deren Verwaltungsinformation zu einem Träger höherer Datenrate zusammengefaßt

Beispiel T1-Hierarchie:



- Multiplexen erfolgt bitweise nicht byteweise!
- $4 * T1 = 1 * T2$; $6 * T2 = 1 * T3$; $7 * T3 = 1 * T4$
- Das Multiplexen von E1-Rahmen bei E1-Systemen erfolgt in Schritten von 4

E1-PCM-Hierarchie

PCM	30 =	2,048 Mbit/s (= E1)
x 4		
PCM	120 =	8,448 Mbit/s (= E2)
x 4		
PCM	480 =	34,368 Mbit/s (= E3)
x 4		
PCM	1920 =	139,296 Mbit/s (= E4)
x 4		
PCM	7680 =	564,992 Mbit/s (= E5)

- PCM-Hierarchien sind üblich, d. h. sehr weit verbreitet
- E1- bzw. T1-Systeme sind untereinander inkompatibel => Datenaustausch erschwert

Sonet/SDH-Netz zur Echtzeitdatenübertragung

SONET/SDH

- SONET = Synchronous Optical Network
- SDH = Synchronous Digital Hierarchy
- SONET ist praktisch identisch zu SDH
- Warum gibt es SONET/SDH?
 - E1- und T1-PCM-Systeme beruhen noch auf Kupferkabeln
 - Moderne PCM-Systeme sollten jedoch Glasfaser-basiert sein
 - SONET/SDH ist das von ITU standardisierte Glasfaser-basierte PCM-System
 - Marktanteil von SONET/SDH bei Glasfaser-PCM-Systemen ist sehr hoch

Ziele von SONET/SDH

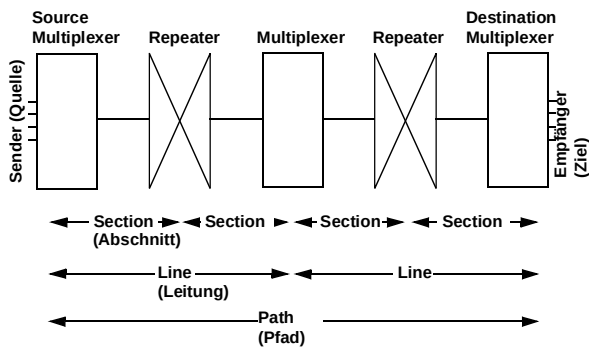
- 5.) Weltweit vereinheitlichter Rahmenaufbau für alle Netzbetreiber
=> Weltweite Kompatibilität aller Glasfaser-PCM-Systeme
- 6.) Kompatibel zu den alten Kupferkabel-basierten PCM-Systemen E1 und T1
- 7.) Unterstützung von Hierarchien gemultiplexer Rahmen
=> Transportiert T1-, T2-, T3-, ..., E1-, E2-, E3-, ..., Träger
 - Transportiert auch ATM-Zellen (SONET/SDH = Bitübertragungsschicht von ATM)

- 8.) Unterstützt automatisierte Verwaltung und Wartung der Netzkomponenten (=Operation, Administration and Management, OAM)

Eigenschaften von SONET/SDH

- Synchrones Zeitmultiplexsystem (PCM) mit max. 1ns Unterschied zwischen Multiplexer und Demultiplexer (= „Jitter“)
- Besteht aus Vermittlern, Multiplexern und Repeatern, die über Glasfaser verbunden sind
- Repeater: dienen zum Verstärken des Datensignals
- Multiplexer: fassen mehrere Teilbitsströme zu einem „höheren Träger“ zusammen, z.B. 4*E1 zu E2
- Vermittler:
 - dienen zur Wegewahl
 - sitzen bei Quell-/Zielmultiplexern und allen Multiplexern dazwischen
- SONET/SDH implementiert Punkt-zu-Punkt-Verbindungen (sog. SONET-Pfade)
- Mehrere Punkt-zu-Punkt-Verbindungen ergeben allgemeinen Graphen als Verbindungstopologie
- Meistens werden jedoch mehrere Punkt-zu-Punkt-Verbindungen zu einem Doppelring zusammengeschaltet.
=> doppelte Fehlertoleranz

SONET/SDH-Pfad



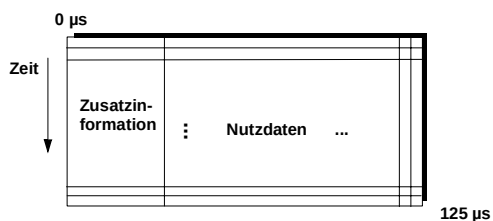
- Abschnitt (section) = Glasfaserstrecke zwischen 2 Geräten (Multiplexer oder Repeater)
- Leitung (line) = Faserstrecke zwischen 2 Multiplexern
- Pfad (path) = Faserstrecke zwischen Quell- und Zielmultiplexer

SONET/SDH-Bitübertragungsschicht

- SONET/SDH bildet die Bitübertragungsschicht gemäß ISO-7-Schichtenmodell
- Bitübertragungsschicht ist in 4 Subschichten unterteilt:
 - Phototeilschicht
 - Abschnittsteilschicht

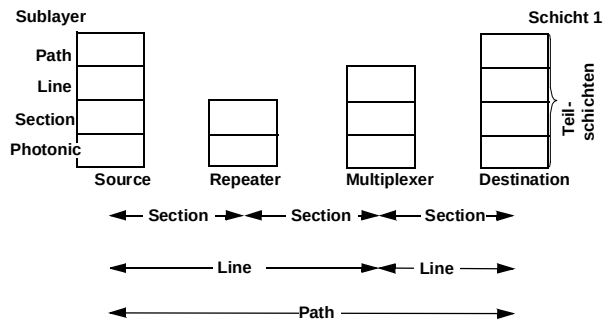
SONET/SDH-Rahmenaufbau

- Der sog. SONET-Basisrahmen ist ein Block von 810 Byte alle 125 μ s.
 - => Bitrate des Basisrahmens ist 51,84 Mb/s
 - => Rate von 8000 Rahmen/s paßt zu alten PCM-Systemen
- Byteblock wird gedanklich als 2-dim. Tabelle gegliedert



- Tabelle hat 90 Spalten und 9 Zeilen (=9*90 Byteblock)
- Die ersten 3 Spalten sind reserviert für Transportzusatzinformation
- Transportzusatzinformation dient für Betrieb, Verwaltung und Management der SONET/SDH-Geräte (OAM)
- Die ersten 3 Zeilen der Transportzusatzinformation enthalten Infos zur SONET/SDH-Abschnittsverwaltung

- Leitungsteilschicht
- Pfadteilschicht



- Phototeilschicht:

- Spezifikation der Glasfaser und der Wellenlänge
- Wandlung zwischen elektrischem und optischem Signal

- Abschnittsteilschicht:

- Erkennen von Rahmenanfang und -ende
- Übertragung von SONET-Rahmen

- Leitungsteilschicht:

- Multiplexen/Demultiplexen von Teilbitströmen aus verschiedenen Rahmen

- Pfadteilschicht:

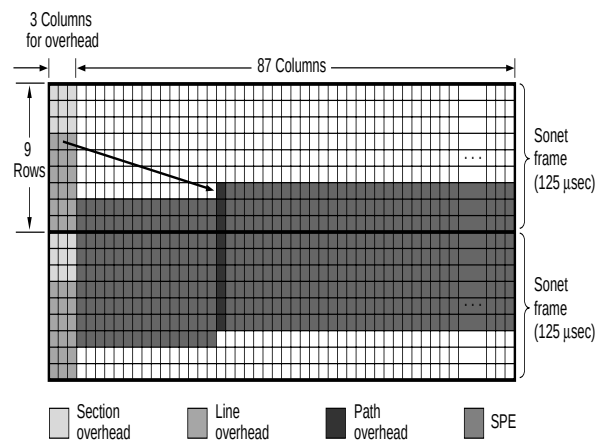
- Ende-zu-Ende-Transport von Daten zwischen Quell- und Zielmultiplexer

- Die restlichen 6 Zeilen der Transportzusatzinformation enthalten Infos zur SONET/SDH-Leitungsverwaltung

- Zusätzlich gibt es im 9*90-Byteblock Infos zur SONET/SDH-Pfadverwaltung

- Die Pfadverwaltungsinformation steht zu Beginn des sog. SPE (=Synchronous Payload Envelope)

- Der SPE enthält Nutzdaten und Pfadverwaltungsinfos



- Der SPE kann mitten im Byteblock beginnen

- Sein Beginn wird durch einen Zeiger in der Leitungsverwaltung gekennzeichnet

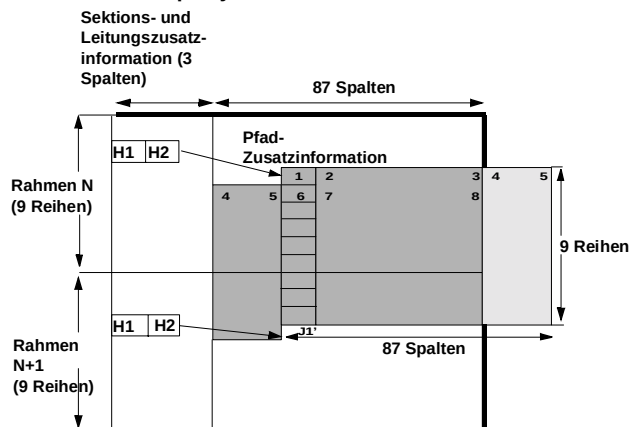
- Der SPE reicht bis zum Beginn des nächsten SPE im nachfolgenden Byteblock
- Der SPE dient zum flexiblen Transport von Nutzdaten verschiedener Länge

Abschnitts-, Leitungs- und Pfadinformation

- Diese Felder werden gebraucht für:
 - Rahmenerkennung
 - Datensicherung (Parität)
 - Fehlerüberwachung
 - Kennungen
 - Taktgabe
 - Synchronisation
 - Sprachkanäle (Byteblock enthält mehrere Kanäle mit 64 Kb/s)
- Die Infos der Abschnittsverwaltung werden am Ende jedes Abschnitts ausgewertet und dann für den nachfolgenden Abschnitt wieder neu erzeugt
- Die Infos der Leitungsverwaltung werden am Ende jeder Leitung ausgewertet und für den nachfolgenden Leitung neu erzeugt

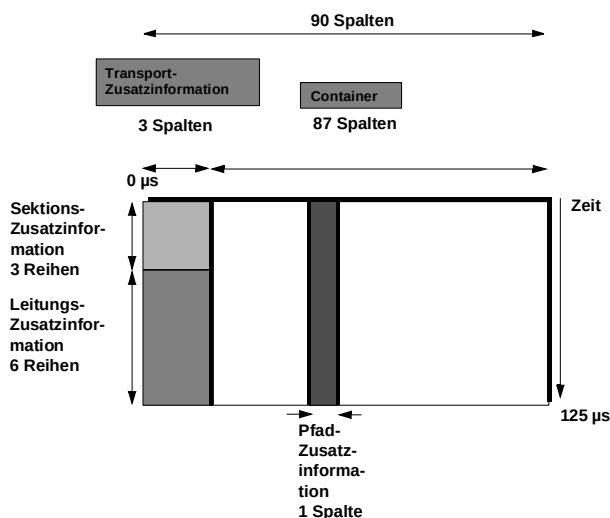
Lokalisierung des Nutzdatenbereichs (SPE)

- Zeiger in Leitungs-Zusatzinformation zeigt Beginn des Nutzdatenbereichs an
- H1 und H2 mit Zeigerwert gibt an: Anzahl der zwischen J1 und J1' liegenden Nutzdatenbytes
- => direkter Zugriff auf einzelne Kanäle (ohne komplexe Multiplex- und Demultiplexvorgänge)
- Nutzdatenbereich kann auch zwei STS-1-Rahmen in Anspruch nehmen
- H3 wird als Stopf-Byte benutzt

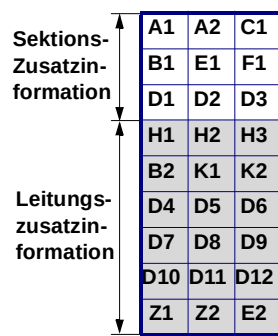


Sonet-Rahmenstruktur

- Synchronous Transfer Signal Level 1 (STS-1) ist Basisrahmen
- Ein STS-1-Rahmen pro 125 μ s
- 810 Bytes je Sts-1-Rahmen => Datenrate: 5184 Mbit/s
- Nutzdaten: 774 Bytes => Nutzdatenrate: 49.536 Mbit/s



Zusatzinformation im SONET-STs-1-Rahmen



Sektions-Zusatzinformation

- A1/A2: Rahmenerkennung: 11110110 / 0010 1000
- C1: Erkennung gemultiplexer STS-1-im STS-N-Rahmen (Binärwert entspricht Position)
- B1: Raritätsfeld (vorausgegangener STS-N-Rahmen)
- E1: 64 kbit/s Sprachkanal (Wartungskanal) für OAM
- F1: 64 kbit/s Sprachkanal (Benutzerkanal)
- D1-D3: Datenkommunikation mit 192 kbit/s (Management/Wartung von Sektionen) für OAM

Leitungs-Zusatzinformation

H1-H3: Zeiger, bzw. Zeigeraktion für SPE

B2: Paritätsfeld (Leitungs-Zusatzinformation und Nutzdaten des vorausgegangenen STS-N-Rahmens)

K1-K2: Signalisierung zwischen Multiplexern (z. B. Fehler) für Kommandoworte

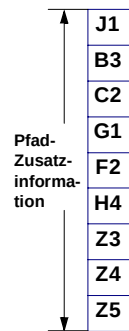
D4-D12: Datenkanal mit 576 kbit/s (Multiplexer-Management)

Z1-Z2: reserviert für Datenwerte zum OAM

E2: 64 kbit/s Sprachkanal (Multiplexer-Management) für OAM

Zusatzinformation im SONET-STS-1-Rahmen

Pfad-Zusatzinformation:



J1: 64 kbit/s-Kanal mit periodischer 16 oder 64 Byte Nachricht zur Funktionsüberprüfung für OAM

B3: Paritätsfeld (Nutzdaten des vorausgegangenen STS-N-Rahmens)

C2: Aufbau/Inhalt STS-Nutzdaten z. B. 13_H = ATM Zellen; 14_H = DQDB-Daten

G1: Rücktransport von Zustandsinformation über den entfernten Pfad-Endpunkt

F2: 64 kbit/s-Benutzerkanal für OAM

H4: Bedeutung abhängig von Nutzung (C2)

Z3-Z5: reserviert

SONET/SDH-Multiplexen

- Beim Multiplexen werden mehrere vollständige Byteblöcke

niederer Datenrate incl. deren Verwaltungsinformation zu einem Byteblock höherer Datenrate zusammengefaßt

- Der Byteblock höherer Datenrate wird als Träger bezeichnet.

- Multiplexen der Byteblöcke erfolgt auf Bytebasis

- Der neue Byteblock hat die n-fache Größe wie die Eingangsblöcke (bei n Eingangsströmen)

- Seine Wiederholrate ist ebenfalls 125 μ s

=> Datenrate ist n mal höher

- SONET verwendet ganzzahlige Vielfache der sog. STS-Grundrate für die Bitraten höherer Träger

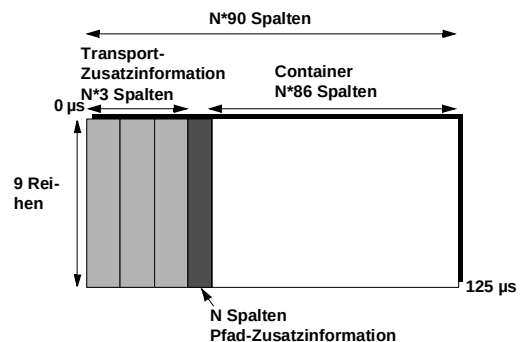
- STS = Synchronous Transfer Signal

- STS-Grundrate (STS 1) = 51.84 Mb/s

- Mehrere Multiplexer hintereinander definieren eine Multiplexhierarchie

- Es gibt (bis jetzt) die hierarchischen Datenraten STS 1, STS 2,..., STS 48

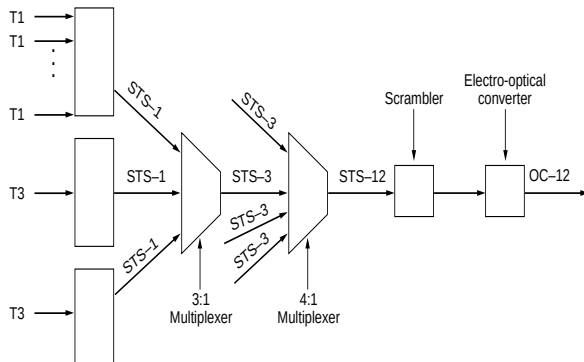
Struktur eines STS-N-Rahmens



- Die Daten werden byteweise in den STS-N-Rahmen gemultiplext

- Die Felder für Sektions-, Leitungs- und Pfad-Zusatzinformation sind jeweils N-mal vorhanden
=> direkter Zugriff auf die Nutzdaten möglich

Beispiel für Multiplexhierarchie bei SONET:



- Das gewandelte optische Signal heißt OC 1, OC2,..., OC 48

- OC = Optical Carrier = opt. Trägersignal

- OC-x ist bis auf eine Bitverwürfelung (Scrambling) das optische Pendant zum elektrischen STS-x-Signal

- Scrambling sorgt für Verschlüsselung

- Scrambling verhindert lange 0000...- oder 1111...-Folgen

- SDH (nicht SONET) verwendet die dreifache STS-Grundrate zum Multiplexen ($3 \times \text{STS-Grundrate} = 155.52 \text{ Mb/s}$)

- Diese wird als STM 1 bezeichnet und gibt direkt die Rate des optischen Signals an (STM 1 = OC 3)

- STM = Synchronous Transport Module

- Dabei werden korrespondierende Spalten mehrerer Byteblöcke derselben Quelle konkateniert

- Der neue Byteblock hat eine entsprechend größere Spaltenzahl

=> Der neue Byteblock hat die nicht gleiche Größe wie die Eingangsblöcke, vielmehr folgen mehr Spalten schneller hintereinander

- Zur Kennzeichnung wird der Buchstabe „c“ verwendet (für concatenated)

Beispiel OC3c:

- 3 OC1-Rahmen einer Quelle der Größe 9×90 werden spaltenweise konkateniert, so daß ein OC3c-Rahmen mit 270 Spalten und 9 Zeilen entsteht.

Hochgeschwindigkeitszugänge

für Multimedia

- Es gibt versch. Möglichkeiten, Multimediadaten zwischen Provider und Endkunden zu transportieren (schneller als per 56 Kb/s-Modem oder S-ISDN):

- Über tieffliegende Satelliten (Low Earth Orbiter, LEOs)
- Über funkbasierte LANs (z.B. neue Handy-Generation)
- Über Datenübertragung im Stromnetz (Powerline-Technik)
- Über den vorhandenen analogen Telefonanschluß per xDSL (x Digital Subscriber Line)

Multiplexraten bei SONET/SDH

SONET		SDH	Data rate (Mbps)		
Electrical	Optical	Optical	Gross	SPE	User
STS-1	OC-1		51.84	50.112	49.536
STS-3	OC-3	STM-1	155.52	150.336	148.608
STS-9	OC-9	STM-3	466.56	451.008	445.824
STS-12	OC-12	STM-4	622.08	601.344	594.432
STS-18	OC-18	STM-6	933.12	902.016	891.648
STS-24	OC-24	STM-8	1244.16	1202.688	1188.864
STS-36	OC-36	STM-12	1866.24	1804.032	1783.296
STS-48	OC-48	STM-16	2488.32	2405.376	2377.728

- 155 Mb/s und 622 Mb/s sind gebräuchliche Datenraten für ATM, wenn ATM über SONET/SDH transportiert wird

- 2,4 Gb/s (STM16) ist die Datenrate des jetzt in Deutschland aufgebauten G-WIN-Netzes (=Nachfolger von B-WIN)

Hohe Datenraten für eine Datenquelle

- Hohe Datenraten ($> \text{STS1}$ bzw. $> \text{STM1}$) können für eine Datenquelle dadurch erzielt werden, daß man mehrerer Basisrahmen dieser Quelle multiplext.

- In diesem Fall wird nicht byteweise gemultiplext sondern spaltenweise

• Über neu zu verlegende Glasfasern und Kupferkabel per FTTC (Fiber to the Curb)

• Über neu zu verlegende Glasfasern bis zum Endkunden per FTTH (Fiber to the Home)

• Über neu zu verlegende Glasfasern und Koaxkabel per HFC (Hybrid Fiber Coax)

- Powerline-Technik wird von den Stromherstellern forciert

- Satelliten, Funk-LANs, xDSL, FTTC und FTTH wird von den Telecoms favorisiert

- HFC wird von den Kabelfernsehanbietern propagiert

Potentielle Anwendungen von Hochgeschwindigkeitszugängen

- Schneller Zugang zum Internet-Provider z. B. für
 - Home shopping (eCommerce, mCommerce)
 - Distance learning (= Telelearning)
 - Teleconferencing; interactive TV (Bildtelefon)
 - Telecommuting; Teleworking (Heim Arbeitsplätze)
 - SOHO-Anwendungen (Small Office / Home Office)
 - Video-on-Demand
 - statt Video-Kassetten
 - auch Audio on Demand (statt CDs) möglich
 - Near-Video-on-Demand
 - Überwachungsdienste über web-Kameras
 - Infodienste mit Graphik über schnellen WWW-Zugang
 - Applikationssoftware im Netz statt auf lokaler Platte
 - Zugang zu Multimedia-Diensten und -Anwendungen im Internet
 - Surfen im Internet

HFC (Hybrid Fiber Coax)

- Heutige Koaxkabel für Kabelfernsehen mit ca. 450 MHz Bandbreite sollen durch Kabel höherer Bandbreite (z.B. 750 MHz) ersetzt werden => 300 MHz zusätzlich
- Zusätzliche Bandbreite von ca. 50 6-MHz-Fernsehkä- nalen soll zur Übertragung von Multimediadaten verwendet werden (unidirektional)

- Zur Kostenabrechnung werden die Daten verschlüsselt übertragen (ähnlich Pay-TV)
- Multimediadaten sind MPEG1/2-kodierte Videofilme (Video-on-Demand) mit zusätzlichen zeitdiskreten Informationen sowie web-Seiten vom Internet

FTTC (Fiber to the Curb)

- Curb = Randstein
- Glasfasern sollen bis in die einzelnen Straßen verlegt werden. Von dort geht es per verdrehten Kupferkabeln in die Häuser ("last mile")

=> 2-stufige Hierarchie eines Verteilnetzes

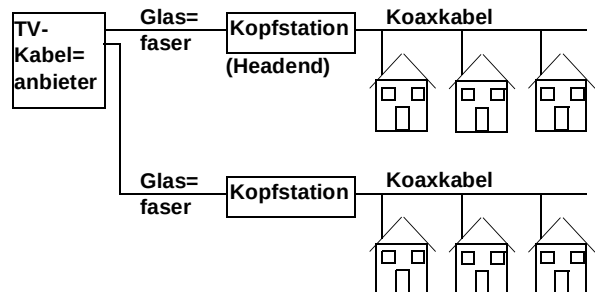
- Datentransfer ist im Gegensatz zu HFC Punkt-zu-Punkt
- Kupferkabel der "letzten Meile" sind kurz

=> genug Bandbreite für MPEG1/2 ohne teure Koaxkabel, incl. Telefonanschluß

- Um Rückkanal zu ermöglichen, sollen die unidirektionalen Verstärker der Kabelsignale durch bidirektionale ersetzt werden (teuer!)

- Koaxkabel werden vom Provider über Glasfasern mit analogen Fernsehprogrammen und digitalen Multimediadaten versorgt

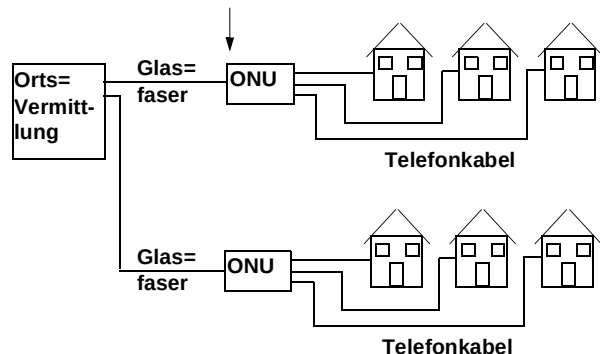
- Verteilung beim Erkunden über gemeinsam genutztes Medium (Kabel)



Beispiel:

- Die 50 neuen 6-MHz-Kanäle werden mit QAM (Qadrature Amplituden Modulation) moduliert
- => pro Kanal sind 40 Mb/s Echtzeitdatenrate möglich
- Alle 50 Kanäle werden per Broadcast an die Haushalte verteilt

=> Nutzung der bereits vorhandenen Telefonkabel



- ONU = Optical Network Unit = el./opt. Konverter
- Jede ONU versorgt ca. 16 verdrehte Kupferkabel mit Daten in beiden Richtungen (bidirektional)
- Nachteil zu HFC: viel mehr ONUs als Headends nötig

FTTH (Fiber to the Home)

- Jeder Haushalt erhält einen SONET/SDH-Anschluß mit z.B. 155 Mb/s (OC3)

=> ultimative Lösung der fernen Zukunft

- Nachteil: immense Kosten (>50% der Investitionen des Telefonsystems liegen in den Kupferkabeln zum Endkunden)

xDSL (Digital Subscriber Line)

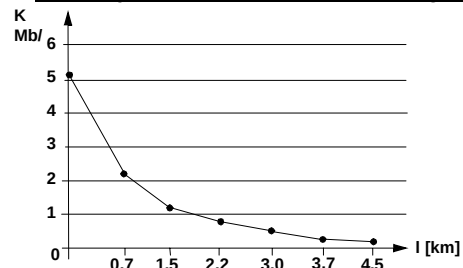
- xDSL ist die Technologie der "Supermodems" für Telefonanschlüsse (=verdrillte Zweidrahtleitungen)
- Betreibt analoge Telefonleitungen mit wesentlich höherer Bandbreite (> 3 KHz) als die 56-Kb/s-Modems
- Nutzt Tatsache, daß ab Ortsvermittlung alles digital ist und sehr hohe Bandbreiten hat (SONET/SDH-Hierarchie)
- Nutzt Fortschritte der digitalen Signalverarbeitung (Signalprozessoren) zur Beseitigung von Echos, Übersprechen und Störungen auf den Zweidrahtleitungen
- Grundlagen: Übertragungsgeschwindigkeit eines rauschbehafteten Kanals (Shannon-Theorem)

$$K = B \cdot \log_2 \left(1 + 10^{\frac{S}{N \cdot 10}} \right) \Rightarrow k \sim B$$

- K = Übertragungsgeschwindigkeit (Kanalkapazität) in Band = $\left[\frac{1}{s} \right]$
- B = Bandbreite des Kanals in Hertz = $\left[\frac{1}{s} \right]$
- S/N = Signal- zu Rauschleistung in dB

- Beispiele:
 - 3 KHz Telefonbandbreite erlaubt bei 35 dB Signal/Rauschverhältnis eine Datenrate von ca. 30 Kb/s
 - 8 Mb/s Datenrate erfordert ca. 800 KHz Kanalbandbreite
- Kanalbandbreite ist abhängig von Leitungslänge und Leitungsquerschnitt
- In Deutschland:
 - Leitungslänge stets kleiner 10 km,
 - Typische Länge ist 2-4 km,
 - Leitungsquerschnitt 0,4 oder 0,6 mm,
 - Fast nie gibt es Verstärker zwischen Kunde und Ortsvermittlungsstelle
- => Nur die Übertragungsbandbreite des Kabels muß berücksichtigt werden

Kanalkapazität als Funktion der Leitungslänge

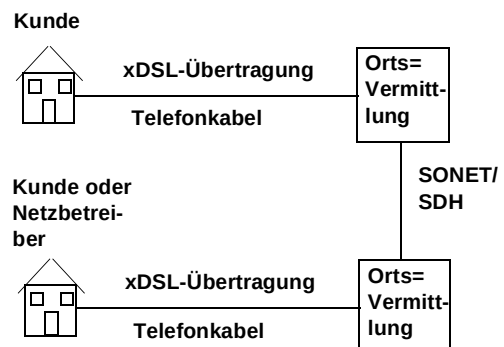


=> Telefonkabel erlaubt auf kurze Entfernungen hohe Datenraten

Warum ist die XDSL-Technik wirtschaftlich interessant?

- über 700 Millionen installierte Telefonanschlüsse weltweit
- 96 % Kupferkabel-basierend
- > 50 % der gesamten Investition im Telecom-Sektor
- Glasfaserkabel nach Hause werden noch viele Jahre dauern (hohe Installationskosten; weniger für das Kabel als für die Verlegung)
- => XDSL ist eine sehr kosteneffektive Lösung, bereits installierte Kupferkapazität wird ausgenutzt
- ADSL (Asymmetri Digital Subscriber Line) und die verwandten Techniken HDSL, SDSL und VDSL übertragen sehr hohe Bitraten (bis zu 50 Mbit/s) über un abgeschirmte Kupferdrähte (Telefondrähte).

xDSL Konfiguration



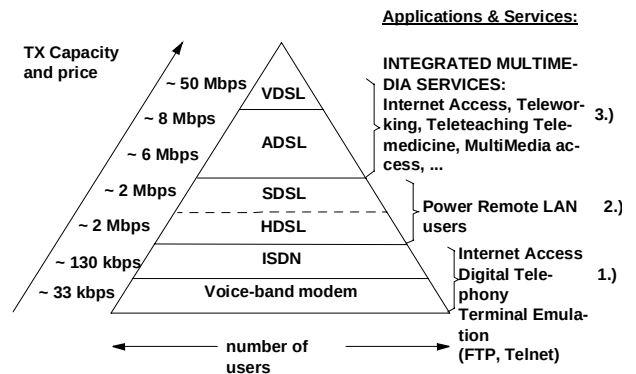
Beispiel: ADSL

- Kunde hat Zugriff auf Telefonnetz und digitales Datennetz eines Providers über dieselbe Telefonleitung
- Leitung wird über Frequenzweiche im Frequenzmultiplex betrieben (sog. splitter)
- Datenrate vom Kunden (upstream) bis 640 Kb/s
- Datenrate zum Kunden (downstream) bis 8 Mb/s

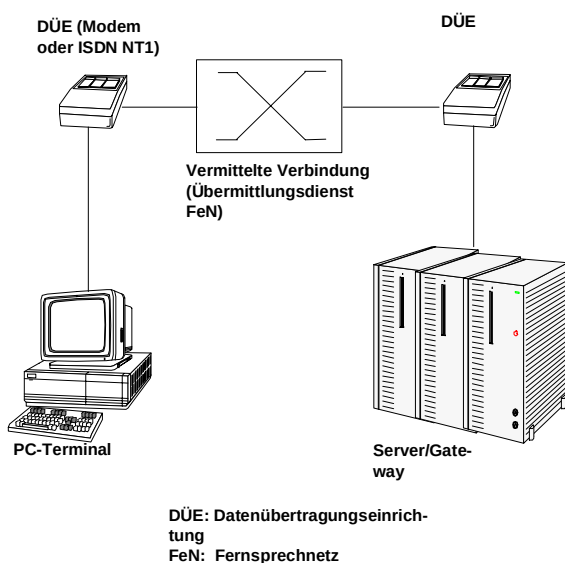
Die verschiedenen Typen von xDSL

- ADSL: Asymmetric Digital Subscriber Line
- HDSL: High Data Rate Digital Subscriber Line
- IDSL: das heutige ISDN
- SDSL (entspricht HDSL2): Single Line Digital Subscriber Line
- VDSL (Nachfolger von ADSL): Very High Rate Digital Subscriber Line
- UDSL (U-ADSL): Universal Digital Subscriber Line (ADSL-light, G. light)
- xDSL: Sammelbezeichnung für alle DSL-Arten

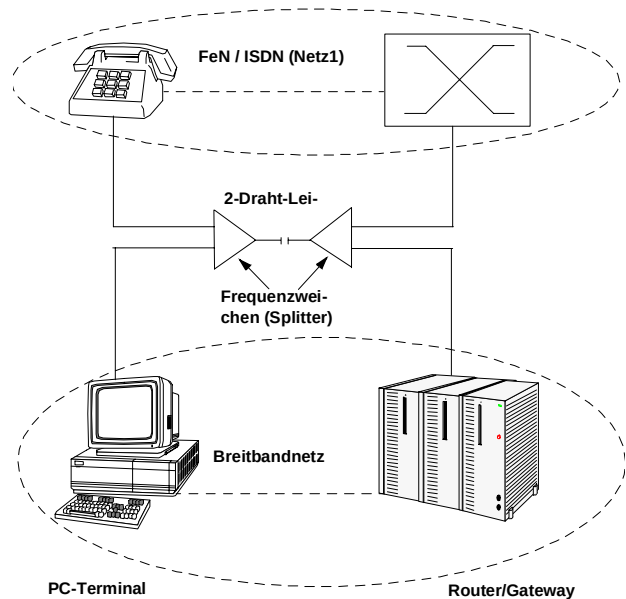
Übersicht über die xDSL-Techniken



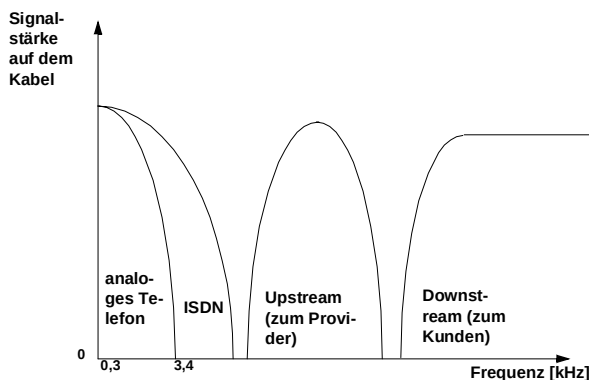
Vergleich: konventioneller Netzzugang und ADSL



Doppel-Netzzugang mittels ADSL

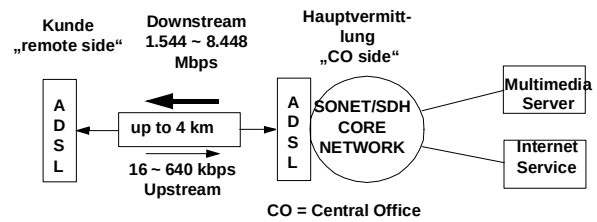


ADSL-Aufteilung der Übertragungsfrequenzen



POTS: Plain Old Telephone System

ADSL- Asymmetric Digital Subscriber Line



- Duplexübertragung mit „fixen“, asymmetrischen Datenraten über eine Kupfer-Zweidrahtleitung
- Die erreichten Übertragungsraten sind von der Entfernung und von der Leitungsqualität abhängig. Die Adaption erfolgt automatisch.
- Basiert auf CAP- oder DMT-Modulationstechniken oder QAM
- Telefondienst und ADSL-Datendienst simultan (stören sich nicht)
- Typische Anwendungen: schnelle Datenleitungen in Privathaushalten, Internet-Zugang, Fernzugang zu LANs, Multimedia-Zugang, Video-on-Demand, ... und ganz normales telefonieren oder faxen

CAP: Carrierless Amplitude/Phase Modulation

DMT: Discrete Multitone Modulation

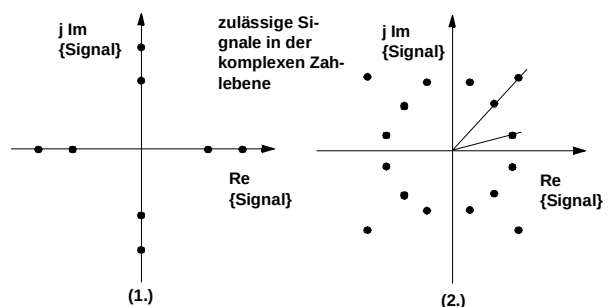
QAM: Quadrature Amplitude Modulation

ADSL: Warum asymmetrisch

- Die „upstream“-Signale laufen bei der Vermittlungsstelle in großer Zahl zusammen, was signifikante Störungen durch Übersprechen verursacht, wenn die Signale schon stark gedämpft sind
- Dagegen laufen die „down-stream“-Signale auseinander, zu getrennten Teilnehmer-Modems, wobei das Übersprechen sich wesentlich weniger auswirkt
- Deshalb kann man in der „downstream“-Richtung höhere Bitraten realisieren
- Zugleich können viele breitbandige Anwendungen mit asymmetrischen Bitraten gut leben, zum Beispiel das Browsen im Internet, Video-on-Demand usw.

Modulationstechniken für ADSL

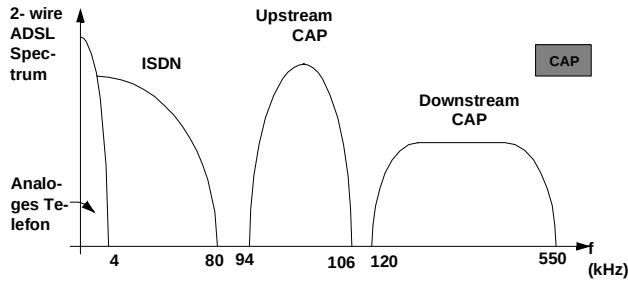
- Basis: QAM (Quadrature Amplitude Modulation)
- Dies ist eine Kombination von Amplituden- und Phasenmodulation
- Jeder „Datenpunkt“ im Diagramm entspricht einer Bitkombination



9.) 2 Amplituden, 4 Phasensprungwinkel, 8 Datenpunkte, also 3 Bits pro Baud

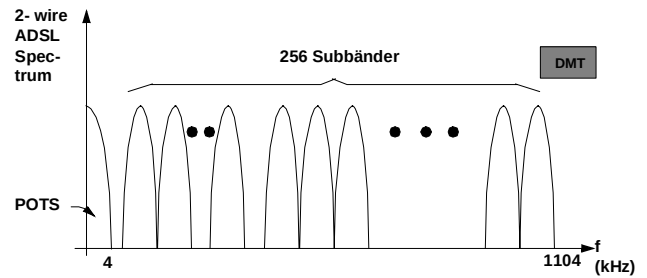
10.) 16 Datenpunkte, also 4 Bits pro Baud (verwendet im V.32-Modem für 9600 bit/s bei 2400 Baud)

- CAP-Carrierless Amplitude/Phase Modulation



- Eine Variante der Quadrature Amplitude Modulation (QAM)
 - Berechnung des kombinierten Signals durch einen digitalen Signalprozessor
 - Benutzung einer einzigen Trägerfrequenz, die rausgefiltert wird
 - Telefondienst und ISDN liegen unterhalb des CAP-Frequenzspektrums
- => aufwärtskompatibel

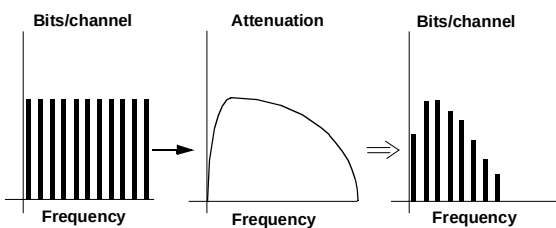
DMT-Discrete Multitone Modulation



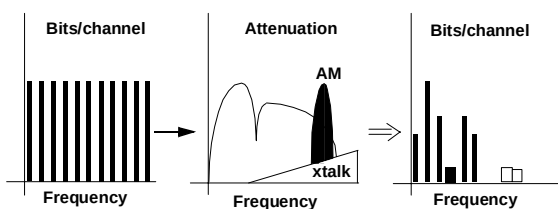
- Im Prinzip Frequenzmultiplexing (FDM) mit separater Bitraten-Adaption pro Trägerfrequenz
 - Frequenzspektrum: 26 kHz bis 1.1 MHz
 - Ist in 256 individuelle Sub-Trägerfrequenzen unterteilt, je 4 kHz breit
 - Jeder Kanal übermittelt bis zu 60 kbit/s
 - Der Telefondienst liegt unterhalb der DMT-Frequenzen für Datendienste und wird separat geführt (splitter an beiden Enden der Leitung)
 - ANSO T1.413 Standard (ADSL), auch ein europäischer ETSI-Standard
- FDM: Frequency Division Multiplex
- splitter: Frequenzweiche. Dient zur Aufteilung der Frequenzbereiche

Automatische Bitraten-Adaption bei DMT

- DMT adapts to every line condition
- TWISTED PAIR without side effects



- TWISTED PAIR with TAP, AM/RF, and XTALK



- Bei ADSL wird die Bitrate dynamisch an die Länge und die Qualität der Übertragungsstrecke angepasst

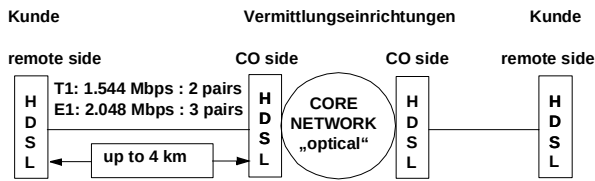
- Bei der DMT-Modulation messen die Modems ständig die Übertragungsgüte jedes einzelnen Kanals (jeder Trägerfrequenz) und adaptieren die Bitrate gemäß den aktuellen Eigenschaften

TAP: Kabelabzweigung mit Reflexion des Signals (Echo)

AM/RF: hochfrequente Einstreuung (von außen)

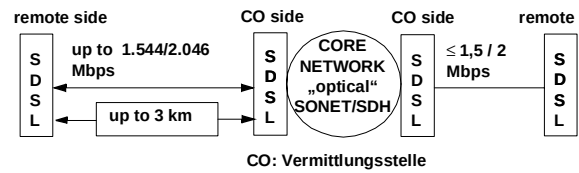
XTALK: Übersprechen (selbst verursacht)

HDSL - High Data Rate Digital Subscriber Line



- Hohe, symmetrische Bitraten über parallele Kupferdrähte
- Entstanden als kostengünstige Technik für die Telecoms zur Realisierung von T1 oder E1 (1,5 Mbit/s oder 2 Mbit/s) über zwei bis drei Zweidrahtleitungen
- Basiert auf 2B1Q (Quadrature Amplitude Modulation, 2 Bits pro Baud) oder CAP-Modulationstechniken (einer digitalen Variante von QAM)
- Kein simultaner Telefondienst auf dem Kabel
- Typische Anwendungen: T1 oder E1 in Gebäude, die keinen Glasfaseranschluss haben

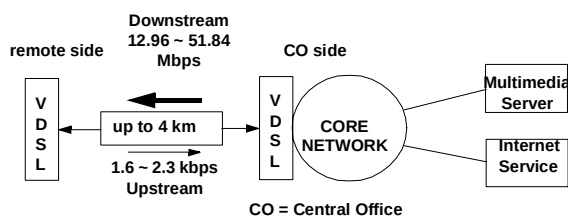
SDSL - Symmetric Digital Subscriber Line



- „SINGLE LINE“-Version von HDSL (nur eine Zweidrahtleitung)
- Symmetrische Bitraten
- Basiert auf 2B1Q (QAM), CAP oder DMT-Modulationstechniken
- Telefondienst und T1/E1 simultan
- Typische Anwendungen: wie HDSL

VDSL -

Very High Data Rate Digital Subscriber Line



- Duplexübertragung mit fixen, asymmetrischen oder symmetrischen Datenraten über eine Zweidrahtleitung
- Höhere Datenraten als ADSL, aber kürzere Kabellängen
- Telefondienst, ISDN und Datenübertragung simultan
- Typische Anwendungen: nächste Generation der über ADSL zur Verfügung gestellten Dienste
- Noch keine Standards, zurzeit in der Diskussion/Erprobung

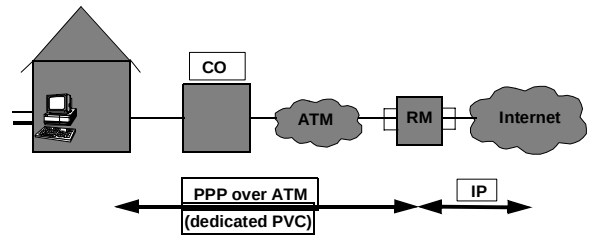
Die xDSL-Familie im Überblick

symmetrische Datenraten

Bezeichnung	ADSL	SDSL	HDSL	VDSL
Bedeutung	Asymmetrische DSL	Single Line DSL	High Data Rate DSL	Very High Data Rate DSL
Upstream	16 bis 768 Kb/s	1,544 bis 2,048 Mb/s	1,544 bis 2,048 Mb/s	1,5 bis 2,3 Mb/s
Downstream	1,5 bis 8,4 Mb/s	1,544 bis 2,048 Mb/s	1,544 bis 2,048 Mb/s	13 bis 52 Mb/s
max. Leitungslänge	2,7 bis 5,5 km	2 bis 3 km	3 bis 4 km	0,3 bis 1,5 km
Adernpaare	1	1	2 bei 1,544 Mb/s, 3 bei 2,048 Mb/s	1
POTS im Basisband	ja	nein	nein	ja
ISDN im Basisband	nein	nein	nein	ja
benutzte Bandbreite	bis 1 MHz	bis 240 kHz	bis 240 kHz	bis 30 MHz
Modulationsverfahren	CAP, DMT	CAP	CAP	CAP, DMT, DWMT

Bezeichnung	ADSL	SDSL	HDSL	VDSL
Einsatzbereiche	Fast Internet, interaktive Videodienste, Multimedia	T1/E1-Dienste, WAN- und Server-Zugänge	T1/E1-Dienste, WAN und Serverzugänge	Fast Internet, interaktive Videodienste, Multimedia
Verfügbarkeit	>Mitte 90er	>Anfang 90er	>Anfang 90er	>Ende 90er

Protokolle in den höheren Schichten (> Bitübertragungsschicht)



ONU = Optical Networking Unit

CO = Central Office (Knotenvermittlungsstelle)

RAS = Remote Access Server

PVC = Permanent Virtual Circuit

- Internet-bezogener Verkehr wird die Geschäftsgrundlage für breitbandige Zugangsnetze bilden (zumindest in der nahen Zukunft)
- Zugang über ADSL zum Internet wird durch ähnliche Mechanismen möglich sein wie beim Zugang über ein Telefonmodem, also insbesondere über Point-to-Point-Protocol (PPP) zu einem Remote Access Server (RAS)
- PPP erfüllt dabei verschiedene Aufgaben, z. B. dynamische Vergabe von IP-Adressen, Authorisierung usw.

- B-ISDN wurde ursprünglich als Konzept von der Vereinigung der Telefongesellschaften (CCITT) entworfen
- Später wurde es gemeinsam mit Computerherstellern im sog. ATM-Forum weiter entwickelt.
- Zur Implementierung des Konzepts wurde das ATM-Protokoll spezifiziert (ATM = Asynchronous Transfer Mode)

B-ISDN

-
-

(Broadband Integrated Services Digital Network = Breitbandiger integrierter Service über digitales Netzwerk)

Ziele von B-ISDN

- Weiterentwicklung von Schmalband-ISDN hinsichtlich höherer Bandbreiten und zusätzlicher Dienste
- Zusammenfassen aller Netze, die die Telekom betreiben, unter ein einheitliches Protokoll

Dienste, die ATM bereitstellen soll

- Sprachübertragung
- konventionelle Datenübertragung
- Übertragung multimedialer Daten
- Video-on-Demand
- Interaktives Fernsehen aus vielen Quellen
- Kopplung lokaler Netze (LANs)
- Alle Dienste sollen mit wesentlich höherer Bandbreite als bei S-ISDN durchgeführt werden (155 Mb/s, 622 Mb/s, 2,4 Gb/s)

ATM (Asynchronous Transfer Mode)

Was ist ATM?

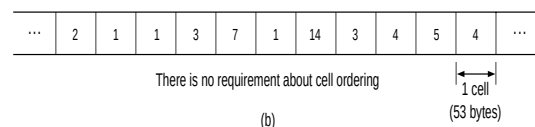
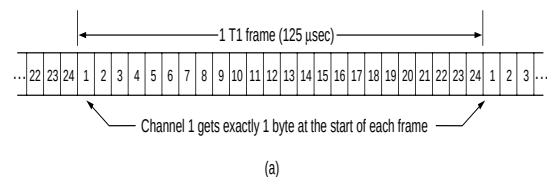
- ATM ist eine Menge von Protokollen, die Teile der Schicht 2, 3 und 4 (Sicherung, Vermittlung und Transport) des ISO-7-Schichten-Modells umfassen
- ATM spezifiziert keine eigene Bitübertragungstechnologie (keine Schicht 1!)
- ATM sitzt auf der Schicht 1 von SONET/SDH, FDDI, Fibre Channel oder anderen Technologien auf
- Die am meisten verwendete Bitübertragungsschicht für ATM ist SONET/SDH
- Für ATM werden spezielle Vermittlungseinrichtungen (ATM-Switches) verwendet, die in Hardware Aufgaben der Schicht 2 (=Vermittlung) übernehmen
- Während die Schicht 1 nicht ATM-spezifisch ist, sind ATM-Vermittler voll ATM-spezifisch
- => ATM ist die Kombination aus Protokollen (Software) und spezieller Vermittler-Hardware
- Oberhalb von ATM-Protokollen und Vermittler können die Internet-Protokolle IP mit TCP bzw.UDP aufsitzen (ist allerdings ineffizient)

Warum gibt es ATM?

- ATM ist die Implementierung von B-ISDN, d.h. der weltweiten Zusammenfassung von Telefon- und Datennetzen zu einem vereinigten digitalen Netz hoher Bandbreite
- ATM ist gedacht als die Verschmelzung der besten Eigenschaften von Telefonsystemen und Datennetzen, wie z.B:
 - Feste, garantierte Verbindungen zwischen Endgeräten
 - Echtzeitfähigkeit
 - Zuverlässigkeit und Qualität in der Übertragung
 - Hohe Datenraten, niedrige Latenzen
 - Hohe Flexibilität beim Einsatz durch skalierbare Datenraten und Entfernungen
 - Gute Nutzung der Übertragungswege durch asynchrone Dateneinspeisung (kein zentraler Multiplextakt)
 - Unterstützung von Multicast und Broadcast
 - Unterstützung multimedialer Übertragung

Warum heißt ATM ATM?

- ATM steht im Gegensatz zum STM (Synchronous Transfer Mode) von SONET/SDH, bei dem aufgrund der Multiplexhierarchie Zeitscheiben fest zugeteilt werden



a) Synchronous b) asynchronous Transfer Mode

- Daten von verschiedenen Quellen kommen i.d.R. zufallsgesteuert an (keine feste Reihenfolge bei der Datenerzeugung)
- ATM erlaubt den höheren Schichten, trotz Verwendung von STM-Technologien auf der Schicht 1, die Einspeisung von Daten zu beliebigen Zeitpunkten (im Rahmen der Bandbreitengrenzen des Mediums)

=> ATM ist asynchron, weil Daten nicht in festem Zeitraster übertragen werden, sondern bedarfsweise

- Dies steht im Gegensatz zur Telefonvermittlungstechnik, die synchron arbeitet
- Dort wird das Time Division Multiplex verwendet, das jedem Sender periodisch einen Zeitschlitz zuweist, egal ob er ihn braucht oder nicht
- Asynchroner Transfer Modus heißt, daß die Daten verschiedener Quellen zu beliebigen Zeitpunkten auf einen gemeinsamen physikalischen Kanal gemultiplext werden können
- Interessanterweise erfolgt trotz des asynchronen Medienzugangs die Datenübertragung meistens synchron z.B. über SONET/SDH als physikalisches Medium

Die wichtigsten Eigenschaften von ATM

- 1.) Bereitstellen eines asynchronen Zugangs zum synchronen Übertragungsmedium
- 2.) Integration aller Netzdienste (Sprache, Daten und Video,...) im Sinne einer multimedialen Übertragung
- 3.) Sehr hohe Bandbreiten bei gleichzeitig niedriger Latenz; Beides kann auf Wunsch garantiert werden
- 4.) Verbindungsorientiert in der Art, daß zwischen den ATM-Endgeräten Standleitungen nachgeahmt werden (sog. virtuelle Verbindungen)
- 5.) Übertragungsrahmen kleiner, fester Länge (immer 53 Byte), sog. Zellen
- 6.) Hoch effiziente Vermittlung der Datenströme in ATM-Vermittlern (ATM Switches)

Bemerkung: ATM-Vermittler sind die Gegenstücke zu Routern bei Datennetzen. Sie ähneln ungefähr digitalen Telefonvermittlungsanlagen

=> Verwendung kurzer Übertragungsrahmen (Zellen) mit wenig Overhead (53 Byte-Zellen mit 48 Byte Nutzdaten)

=> Keine Fehlerbehandlung oder Flußsteuerung im ATM-Netz, sondern nur zwischen den ATM-Endgeräten oder zwischen Endgerät und Netzanschluß

=> Kurze Durchlaufzeiten in den Vermittlern erlaubt kurze Ende-zu-Ende-Latenzen

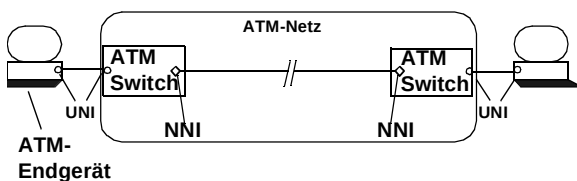
- 7.) Skalierbarkeit in der Geschwindigkeit (Übertragungsraten z.B. 155, 622 und 2,5 Gb/s)
- 8.) Skalierbarkeit in der Entfernung: einsetzbar für LAN, MAN und WAN
- 9.) Quality of Service-Konzept (Vertrag zwischen Dienstnehmer, d.h. ATM-Endgerät und Dienstbringer, d.h. ATM-Netz)

ATM-Netzaufbau

- Ein ATM-Netz besteht aus einer Menge von:

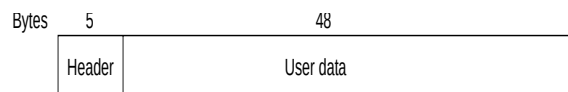
- Benutzerendgeräten
- Vermittlern (ATM Switches)
- Schnittstellen zwischen Endgeräten und Netz (user-to-network interface, UNI)

Schnittstellen zwischen Vermittler und Nachbar-Vermittler (network-to-network interface, NNI)



ATM-Zellen

- ATM-Zellen haben folgenden Aufbau:



- ATM-Zellen sind Datenpakete fester Länge und kleiner Größe

- Dies steht im Gegensatz zu den Paketen der Internet-Protokolle, die variable Länge haben und sehr groß werden können (Bsp: IP 64 KB)

- ATM-Zellen wandern im Netz von Vermittler zu Vermittler

- Im Vermittler werden sie on-the-fly, d. h. zeitlich schritthalte mit ihrem Eintreffen verarbeitet, weil sie wegen des festen Formats leicht zu dekodieren sind

- ATM-Zellen müssen im Netz nicht gespeichert werden (kein store-and-forward).

- Dies steht im Gegensatz zu den Internet-Protokollen und sorgt für kurze Latenzen

- Die sichere Zustellung von ATM-Zellen ist durch das ATM-Protokoll nicht garantiert!

- Garantiert ist die Einhaltung der Paketreihenfolge

ATM-Vermittler

- ATM-Vermittler etablieren Punkt-zu-Punkt-Verbindungen zwischen UNI und UNI bzw. zwischen NNIs
- Für eine ATM-Verbindung muß ein fester Weg von Vermittler zu Vermittler vor Beginn des Datentransfers vereinbart werden (=Verbindungsaufbau)
- => ATM ist bereits auf der Vermittlungsschicht (OSI-Schicht 3) Ende-zu-Ende-orientiert. Dies steht im Gegensatz zu den Internet-Protokollen, die erst auf der Schicht 4 (TCP) verbindungsorientiert sind
- ATM arbeitet wie S-ISDN verbindungsorientiert
- Die Verbindung wird von dem Vermittler aufgebaut, an den das Benutzerendgerät angeschlossen ist
- Zum Verbindungsaufbau signalisiert das UNI des Endgeräts den Wunsch an den Vermittler, an den es angeschlossen ist
- Dieser Vermittler baut eine Verbindung über mehrere Vermittler-Zwischenknoten bis zu dem Vermittler auf, an den das Zielgerät angeschlossen ist
- ATM-Vermittler sind flexibel, weil Verkehr sowohl mit konstanter Rate (z.B. für unkomprimiertes Video) als auch mit variabler Rate (für Daten) übertragen wird
- ATM-Vermittler unterstützen Broadcast-Verbindungen

Bemerkung 3:

Store-and-forward heißt, daß das Paket erst als ganzes eingelesen und gespeichert wird, bevor es weitergeschickt werden kann

Bemerkung 4:

Internet-Router für 100 Mb/s-Ethernet oder Gigabit-Ethernet arbeiten aus Geschwindigkeitsgründen ebenfalls in Hardware, nur ist der Dekodierungsaufwand viel größer als bei ATM-Zellen

- ATM-Vermittler sind in der Größe skalierbar, da sie auf mehrstufigen Verbindungsnetzwerken (z.B. Banyans) beruhen. Dies ist bei Internet-Routern nicht der Fall

Bemerkung 5:

Banyans sind spezielle Hardware-Anordnungen, wie sie in der Telefonvermittlungstechnik eingesetzt werden

ATM-Verbindungen

- Vor Beginn eines Transfers zwischen zwei Endgeräten, muß eine Verbindung zwischen beiden aufgebaut werden (verbindungsorientierte Kommunikation)
- Ist beim Verbindungsaufbau das Zielgerät und das Netz (speziell die sog. Zugangskontrolle (admission control)) mit der Verbindung einverstanden, wird ein "virtueller Kanal" zwischen den beiden Endgeräten etabliert

- Ein ATM-Vermittler benötigt Datenpuffer nur als Warteschlange und dekodiert die Zellen per Hardware on the fly
- Er verwendet Virtual-Cut-Through- oder Wormhole Routing

Bemerkung 1:

Virtual-Cut-Through Routing heißt, daß die Zelle weitergeschickt werden kann, sobald der Paket-Header eingelesen ist

- => Ein Paket kann sich über mehrere Vermittler erstrecken (z.B. Header ist schon im Switch i, Rest noch im Switch i-1)

Ein Paket wird nur dann eingespeichert, wenn ein Vermittler-Ausgang momentan belegt ist

Bemerkung 2:

Wormhole Routing ist wie Virtual-Cut-Through Routing nur, daß bei belegtem Vermittlerausgang das Paket nicht als Ganzes gespeichert wird, sondern nur der Teil, der sich gerade im Vermittler befindet

Die übrigen Teile werden in den vorangegangenen Vermittlern gespeichert

- Ein Internet-Router hingegen arbeitet im Store-and-forward-Modus und dekodiert die ineinander geschachtelten Header der einzelnen Schichten i.d.R. per Software

Virtuelle Kanäle

- Ein virtueller Kanal ist eine temporäre Ende-zu-Ende-Verbindung, die nur für die Zeit zwischen Verbindungsauf- und -abbau existiert
- Eine ähnliche Funktionalität wie sie virtuelle Kanäle haben, wird im Internet durch das TCP-Protokoll bereitgestellt
- Ein virtueller Kanal ist eine logische Direktverbindung zwischen Sender und Empfänger, die eine Standleitung (=fester Draht) nachahmt
- Jeder virtuelle Kanal wird unidirektional betrieben
- Eine bidirektionale Verbindung zwischen zwei Endgeräten entsteht aus zwei antiparallelen virtuellen Kanälen
- Mehrere virtuelle Kanäle können auf einen gemeinsamen physikalischen Kanal gemultiplext werden
- Auf einem physikalischen Kanal können die völlig unterschiedlichen Dienste von Sprach-, Bild- und Datenübertragung von ATM gleichzeitig und gleichermaßen gut übertragen werden
- => Quality of Service-Konzept (gibt es so nicht bei den Internet-Protokollen (IPv4))

Virtuelle Pfade

- Mehrere virtuelle Kanäle können zu einer übergeordneten Menge, dem virtuellen Pfad, zusammengefaßt werden

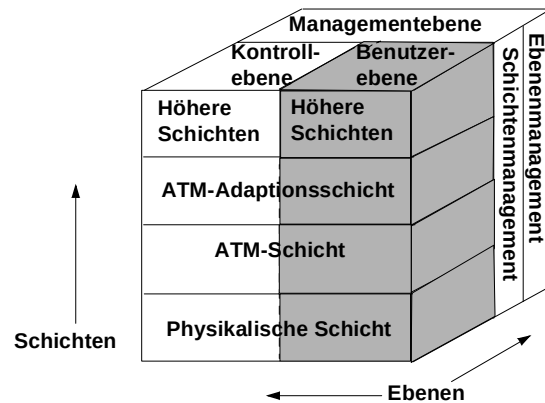
Beispiel:

Zwischen zwei Endgeräten sollen Sprache, Bewegtbild und Daten übertragen werden

=> die drei virtuellen Kanäle werden zu einem virtuellen Pfad zusammengefaßt

- Genau wie virtuelle Kanäle können auch mehrere virtuelle Pfade auf einen physikalischen Kanal gemultiplext werden

ATM/B-ISDN-Referenzmodell



- 3-dimensionales Referenzmodell

- Drei vertikale Ebenen (Säulen)

- Benutzerebene
- Kontrollebene
- Managementebene

- Drei hierarchische Schichten

- Physikalische Schicht (unten)
- ATM Schicht
- ATM-Adaptionsschicht (oben)

- Kontrollinformation wird getrennt von Benutzerdaten übertragen (Out-of-Band-Signalisierung)

B-ISDN: Funktionen der Ebenen

- Benutzerebene

- Übertragung von Benutzerdaten
- Überprüfung und Regulierung des Nutzdatentransfers
- Entspricht dem B-Kanal im (Schmalband-) ISDN

- Kontrollebene (Steuerebene)

- Übertragung von Signalisierungsinformation
- Aufbau, Abbau und Überwachung von Verbindungen
- Entspricht dem D-Kanal im (Schmalband-) ISDN

- Managementebene

- Übertragung von Managementinformation
- Unterteilung in zwei Kategorien
 - Ebenenmanagement: Management des Gesamtsystems, Koordination zwischen den verschiedenen Ebenen
 - Schichtenmanagement: Management einer spezifischen Schicht (z. B. Verwaltung des Pufferspeichers)

B-ISDN: Funktionen der Protokollschichten

• Physikalische Schicht

- Bitübertragung
- Generieren/Wiederherstellen der Übertragungsrahmen
- Entkoppeln der Zellraten
- Erkennen von Zellgrenzen

• ATM-Schicht

- Transport von ATM-Zellen (Dateneinheiten der ATM-Schicht)
- Adressieren von virtuellen ATM-Verbindungen (= Wegewahl, Routing)
- Multiplexen/Demultiplexen von ATM-Zellen auf physikalische Pfade

- Fluss- und Zugangskontrolle

• ATM-Adaptionsschicht (ATM Adaption Layer, AAL)

- Unterstützung unterschiedlicher Transportdienste (U. a. Multimedia-Echtzeit)
- "Konvergenz"funktion (z. B. Fehlerkontrolle, Synchronisation, ...)
- Segmentieren und Reassemblieren von Nachrichten in Zellen bzw. von Zellen in Nachrichten