Rechnernetze 2012

Prof. Dr. Konrad Froitzheim

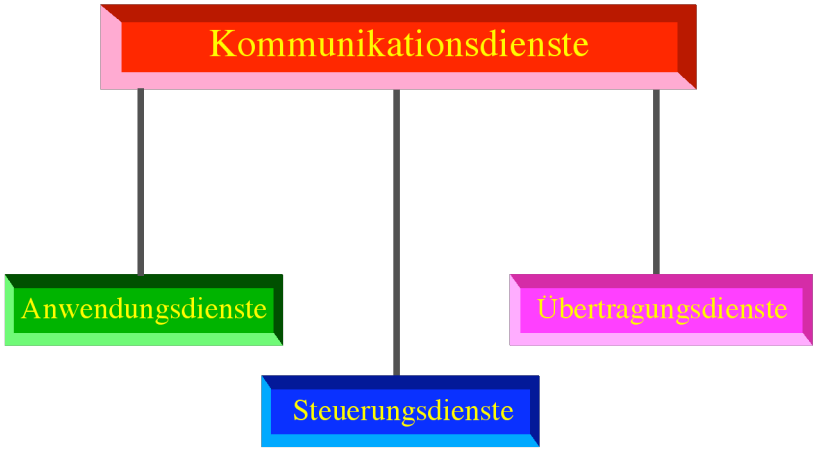
Thema

- Technische Kommunikation
  - Rechnerkommunikation
  - Datenaustausch
  - technische Kommunikation zwischen Menschen
  - Sprache, Video, … => Multimediakommunikation
- Stichworte
  - Telefon, Telefax, …
  - Fernsehen
  - Mobilfunk, Satelliten
  - Lokale Netzwerke
  - Filetransfer, E-Mail
  - Internet
  - Protokolle

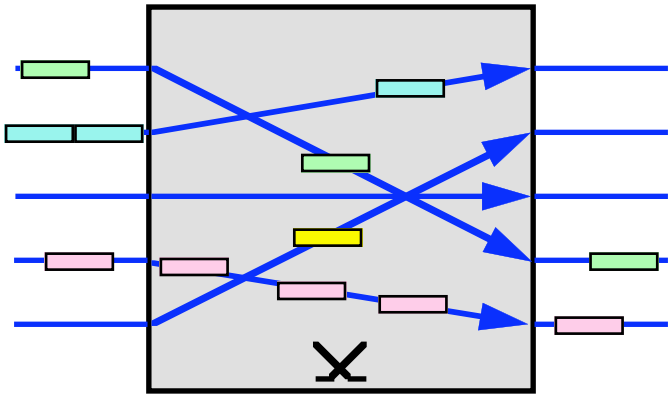
Inhaltsübersicht

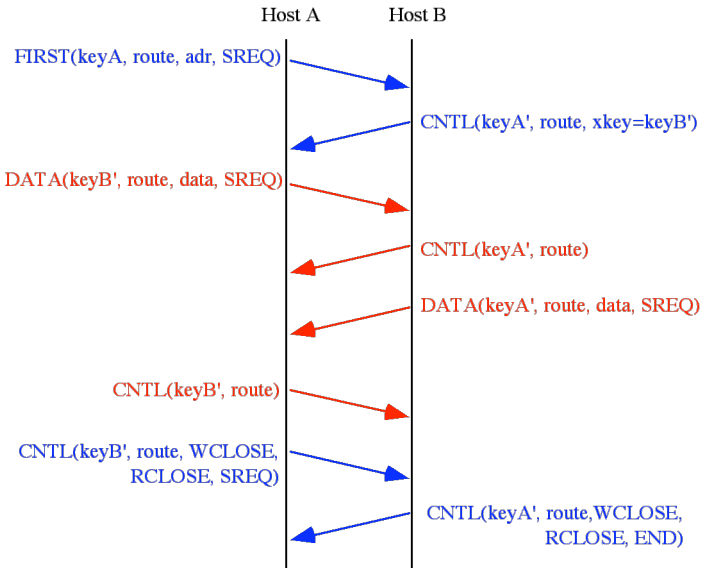
1. Überblick und Taxonomie
   - 1.1 Informationsbegriff (WH)
   - 1.2 Modelle der Kommunikation
   - 1.3 Dienstebegriff
   - 1.4 Verkehrsbeschreibung
   - 1.5 Verbindungen und Topologie
   - 1.6 Namen und Adressen
   - 1.7 Standards

2. Asynchrone Rahmenübertragung
   - 2.1 BSC
   - 2.2 HDLC
   - 2.3 ATM
   - 2.4 Ethernet (WH)



3. Vermittlungsdienste
   3.1 Bit- und Bytevermittlung
   3.2 Zellvermittlung
   3.3 Paketvermittlung
4. Protokolle
   4.1 Aufgaben und Mechanismen
      Synchronisation
      Flußkontrolle
      Verstopfungskontrolle
      Fehlerkontrolle
   4.2 Beispiele
      TCP
      UDP
      RTP
      XTP
   4.3 Protocols considered harmful?





5. Steuerungsdienste
   5.1 ISDN-Signalisierung: Q.931
   5.2 ATM-Signalisierung Q.2931 und ATM-Forum UNI
   5.3 Signalisierungssystem 7 (SS#7)
   5.4 ICMP: Internet Control Message Protocol
   5.5 Netzwerkmanagement SNMP
6. Anwendungsdienste
   6.1 Konsumptive Dienste

   6.2 Kooperative Dienste
      Telefonie
      Visuelle Telefonie
      Messaging
      Sharing

Formales

Termine:

Vorlesung: Dienstag 14:00 - 15:30, HUM-1115
Mittwoch 16:00 - 17:30, HUM-1115

Dramatis Personae:

Prof. Dr. Konrad Froitzheim
Professur Betriebsysteme und Kommunikationstechnologien
frz@tu-freiberg.de

Vorlesungsunterlagen (kein Skriptum):
http://ara.informatik.tu-freiberg.de/Vorlesungen/RN2010.doc

Dipl.-Ing. Mathias Buhr
B.Sc. M. Sc. Frank Gommlich

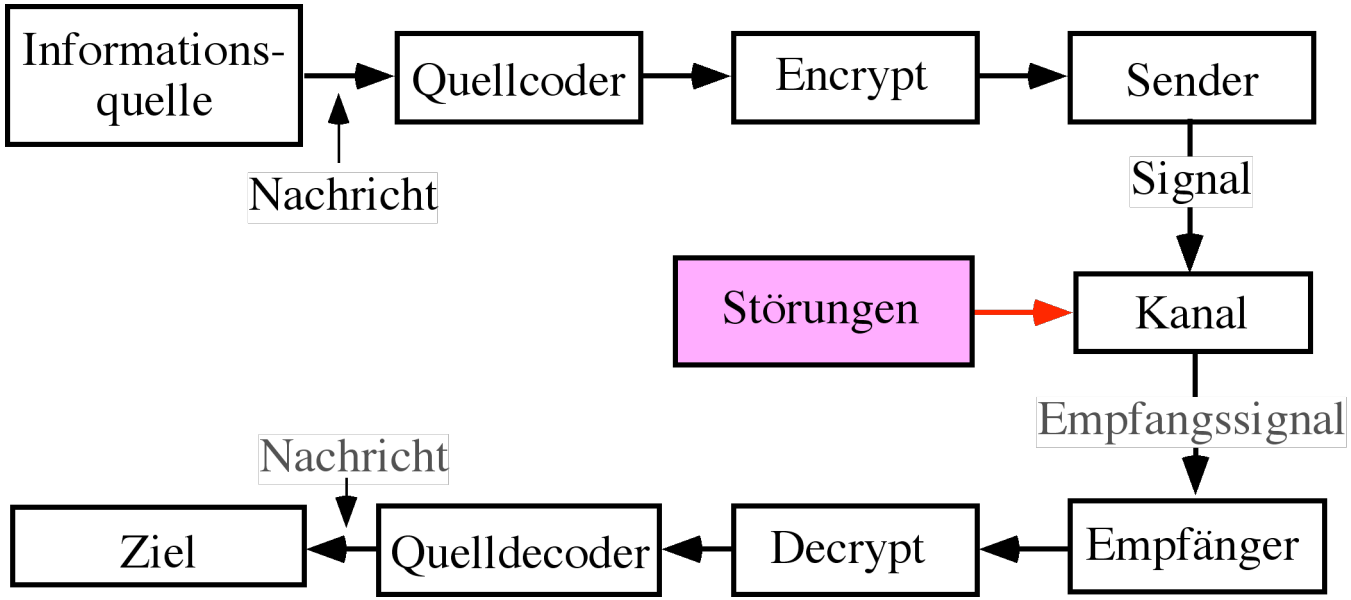
Literatur

- Black, U.: ATM: Foundation for Broadband networks, 1995.
- Black, U.: IP Routing Protocols, 2000.
- Comer, D.: Computer Networks and Internets, 2001.
- Froitzheim, K.: Multimedia Kommunikation, d-Punkt, 1997.
- Halsall, F.: Computer Networking and the Internet, 2005.
- Held, G.: Understanding Data Communications, 1996.
- Krüger, G., Reschke, D.: Lehr- und Übungsbuch Telematik, 2000.
- Minoli, D.: Telecommunication Technology Handbook, 1991.
- Naugle, M.: Illustrated TCP/IP, 1999.
- Pecar, J. et al: Telecommunciations Factbook, 1993.
- Stallings, W.: Networking Standards, 1993.
- Stevens, W.R.: TCP/IP Illustrated, Vol. 1, 1994.

1. Überblick und Taxonomie

1.1 Informationsbegriff

- Information als mathematische Größe [Shannon, 1948]
  - Entropie H = -ld p [bit]
  - Existenzbeweise für Quellkodierung, Kanalkodierung
  - technisches Problem der Kommunikation
- Übertragung

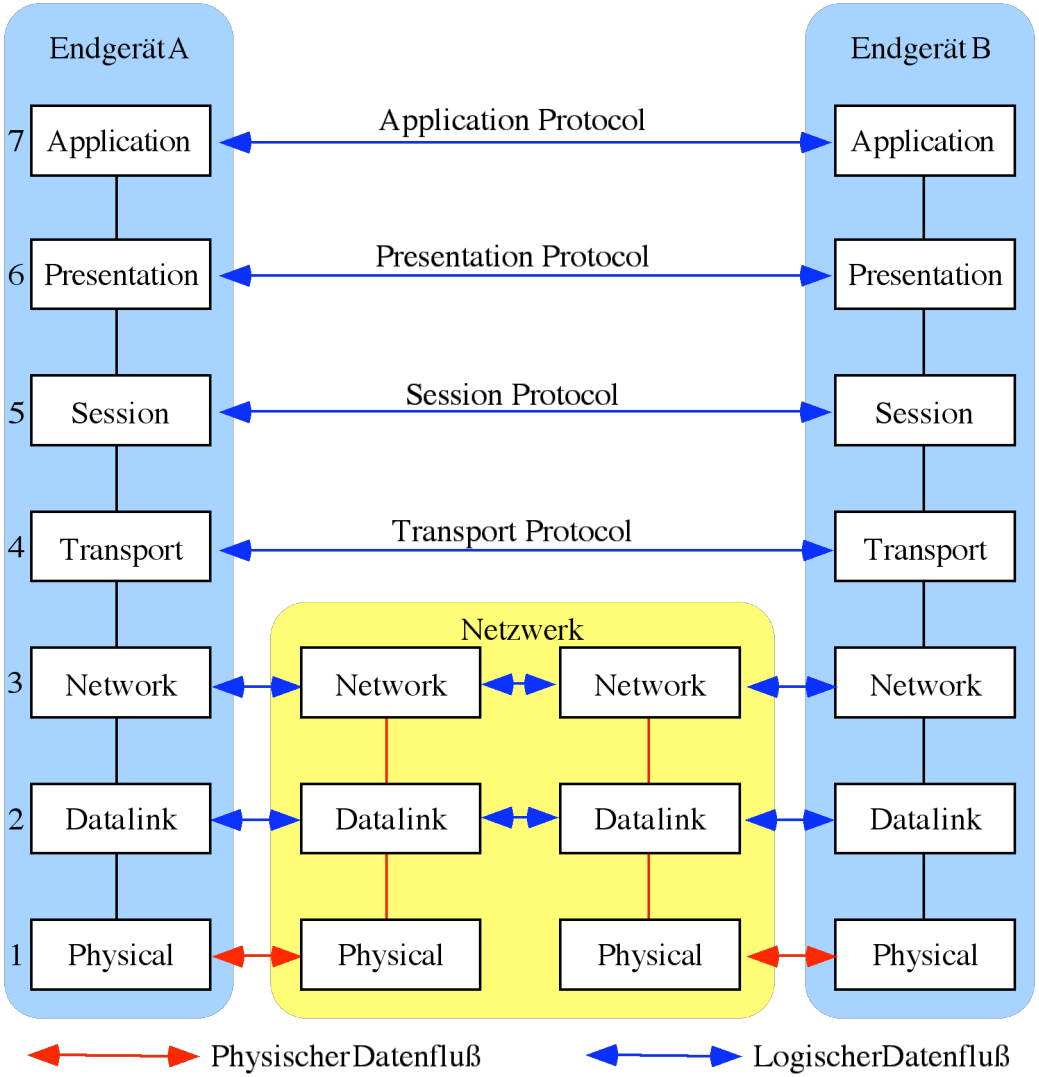


- Kompression siehe Vorlesungen E-Media und Multimedia

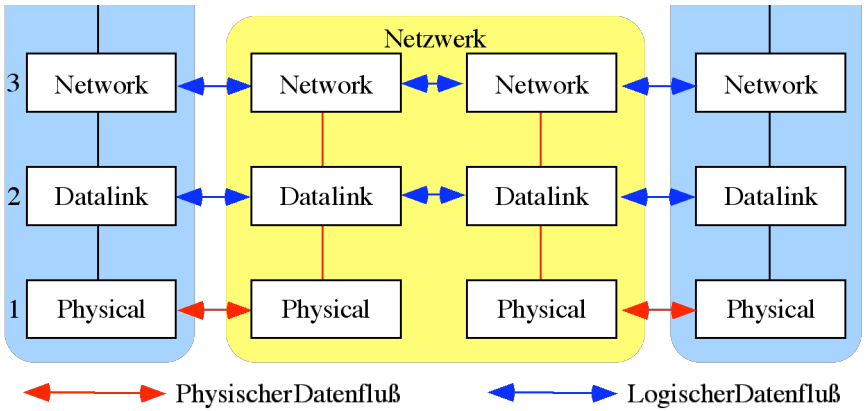
1.2 Modelle der Kommunikation

1.2.1 OSI

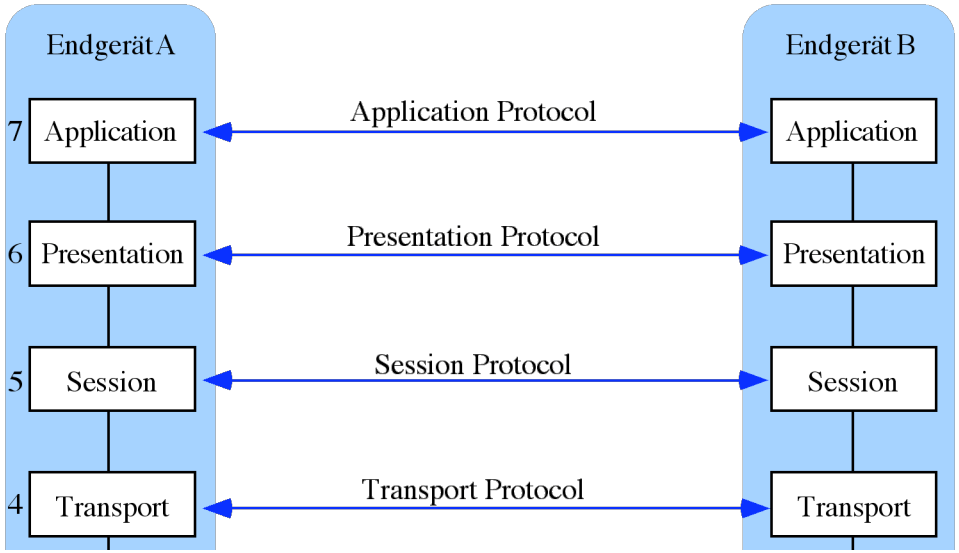
- Open Systems
- definierte Schnittstellen
  - Service Access Point (SAP)
  - Service Data Units
  - Request
  - Response
  - Confirmation
  - Indication
- Peer Protocols
  - Protocol Data Units
  - Verpackung



- Netzwerkabhängige Schichten
- Übertragung (physical layer)
  - ungesicherte Übertragung der Signale über eine Leitung
  - Kanalkodierung, Modulation
  - Bsp: Leitungskodierung
- Verbindungsschicht (data link, Sicherungsschicht)
  - Rahmenbildung (frames)
  - Prüfsummen zur Fehlerentdeckung
  - evtl. Fehlerbehebung
  - Flußkontrolle
  - Verstopfungskontrolle / Lastabwehr
  - Bsp: HDLC
- Vermittlungsschicht (network layer, Netzwerkschicht)
  - Übermittlung zum Kommunikationspartner
  - Routing (Weglenkung)
  - Flußkontrolle
  - Verstopfungskontrolle / Lastabwehr
  - IP, X.25



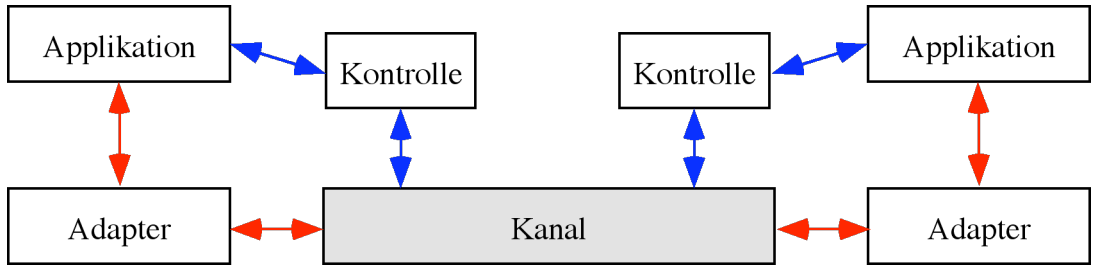
- Netzwerkunabhängige Schichten
- Transportschicht (transport layer)
  - gesicherte Nachrichtenübertragung
  - Adressierung, Numerierung, Prüfung
  - Flußkontrolle
  - Ende-Ende Fehlerbehebung
  - Bsp: TCP, XTP, TP4
- Verbindungssteuerung (session layer)
  - Datenstrom
  - Verbindungsaufbau, Dialogsteuerung
  - Synchronisation und Wiederaufsetzpunkte
  - Bsp: http, ftp, smtp, …
- Darstellungsschicht (presentation layer)
  - Daten als Informationsobjekte
  - Informationsdarstellung, Kompression und Verschlüsselung
  - Transfersyntax zur Datenbeschreibung
  - Bsp: html, sgml, ASN.1, …
- Verarbeitungsschicht (application layer)



- Leistungsprobleme
  - Schnittstelleninflation
  - Marshalling-Overhead und Prozeduraufruf
  - Prozesswechsel
  - schichtinterne Protokolle
  - Verpackungsmüll
- konzeptuelle Probleme
  - Einteilung in Schichten fraglich
  - Funktionsduplizierung
  - Konvertierung der Parameter evtl. verlustbehaftet
  - Semantikverlust im Stack
  - Schichtentimeouts
- Beispiel MAP/TOP, ALF
- Sinnvoll evtl. für Entwurf und Verifikation

1.2.2 Komponentenmodell

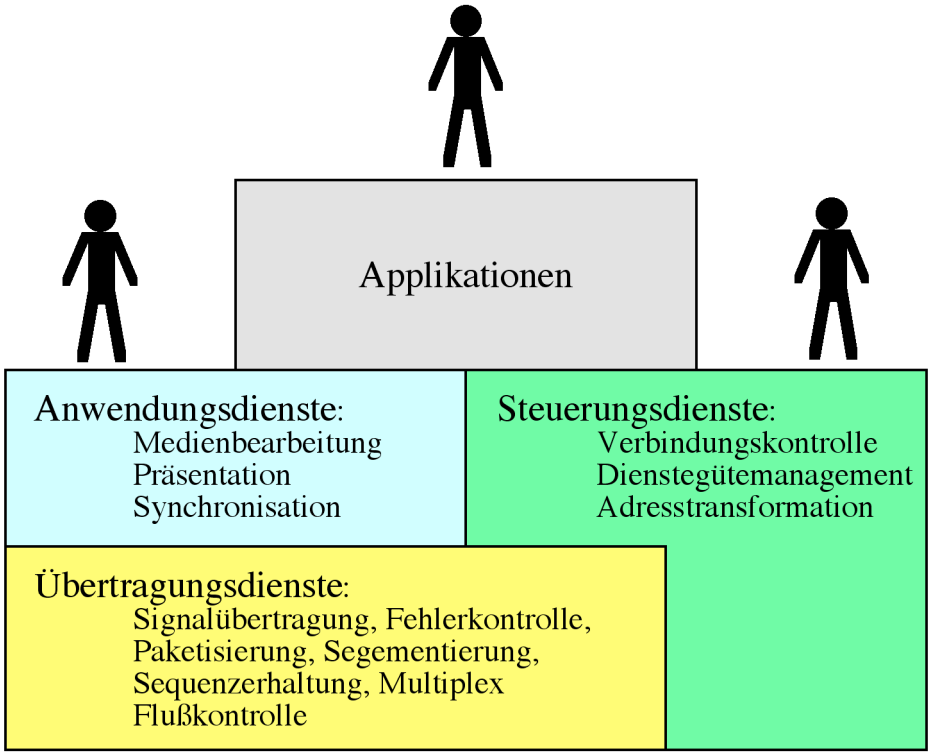
- Komponenten der Kommunikation
  - frei nach Shannon



- Potenzmenge bilden:
  - 16 Mengen
  - Bsp: {Kanal, Kontrolle, Adapter, Applikation}
- sinnvolle Mengen identifizieren. Drei bleiben übrig:
  - {Kanal, Kontrolle, Adapter} - Übertragungsdienste
  - {Kanal, Kontrolle, Adapter, Applikation} - Anwendungsdienste
  - {Kontrolle, Adapter, Applikation} - Steuerungsdienste
- Weitere Unterteilung

  Übertragungsdienste = Übertragung und Vermittlung

- Anordnung der Bereiche

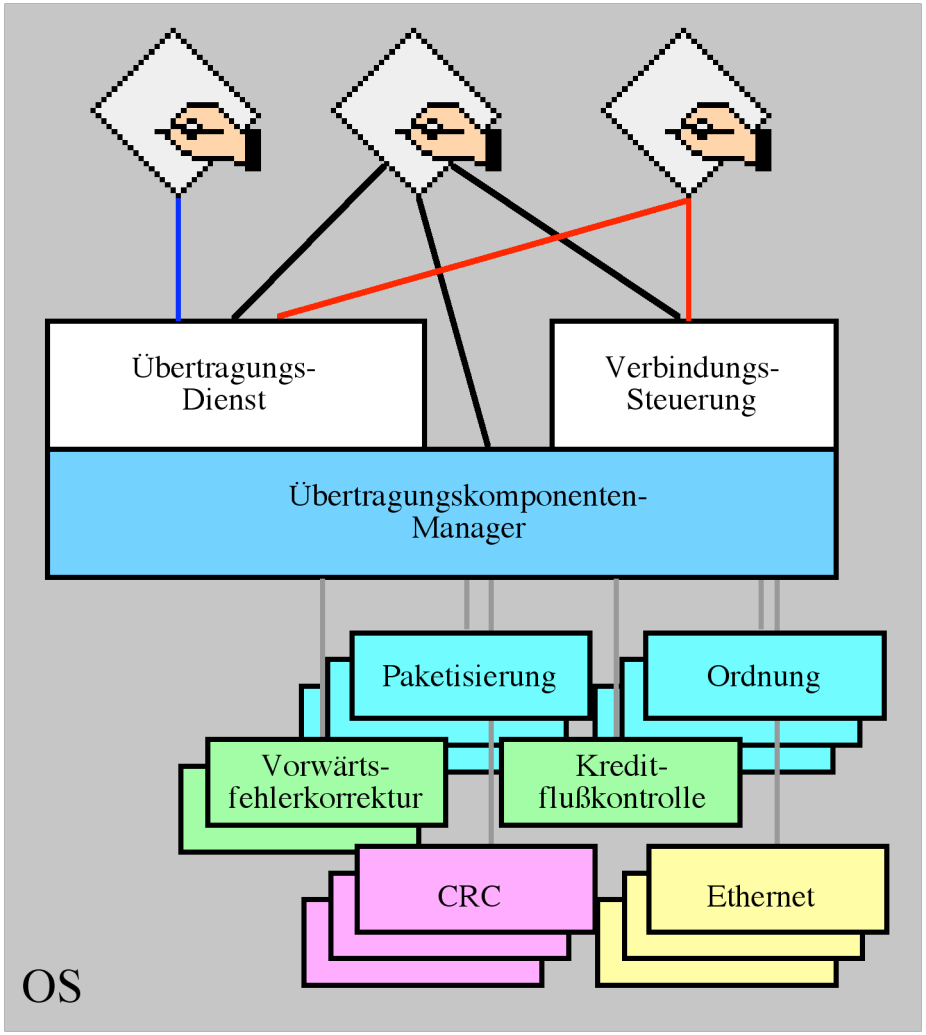


- Übertragungsdienste
  - Signalübertragung
  - Vermittlung
  - Protokolle

- Steuerungsdienste
  - Verbindungsaufbau und -abbau, Features
  - Netzzugang (Autorisierung, …)
  - Netzwerkmanagement?
- Anwendungsdienste
  - kennen Medien
  - End-zu-End Protokolle (http, ftp, …)
  - konsumptive und kooperative Dienste (WWW, Telefon)
  - Anwendungselemente (RPCs, RTP, …)
  - Verzeichnisdienste (DNS, X.500)
  - Endgeräte (Funktelefon)
- Applikationen
  - kennen Semantik
  - verwenden Anwendungselemente und Anwendungsdienste
  - Textprogramme, CSCW, …

- Baukastenprinzip
  - mehrere Abstraktionsebenen
  - Mechanismenklassen
- Klassen
  - Protokollfunktionen
  - Fehlerbehandlung
  - Flußkontrolle
  - Lastkontrolle
  - Traffic-Shaping

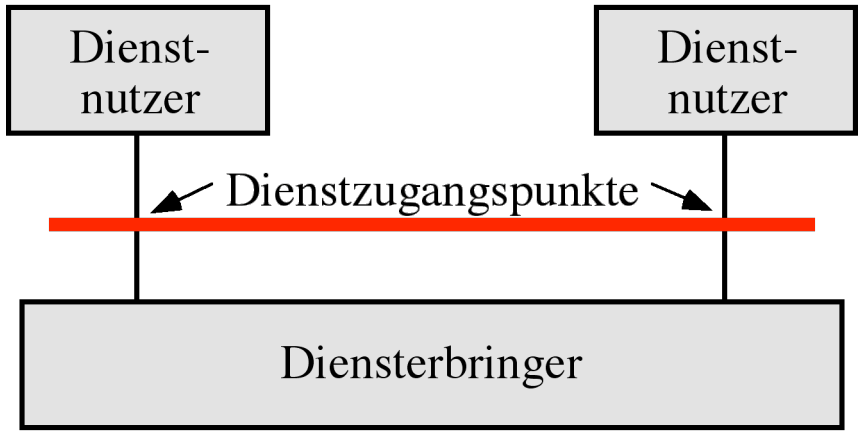
=> konfigurierbare Dienste



1.3 Dienstebegriff

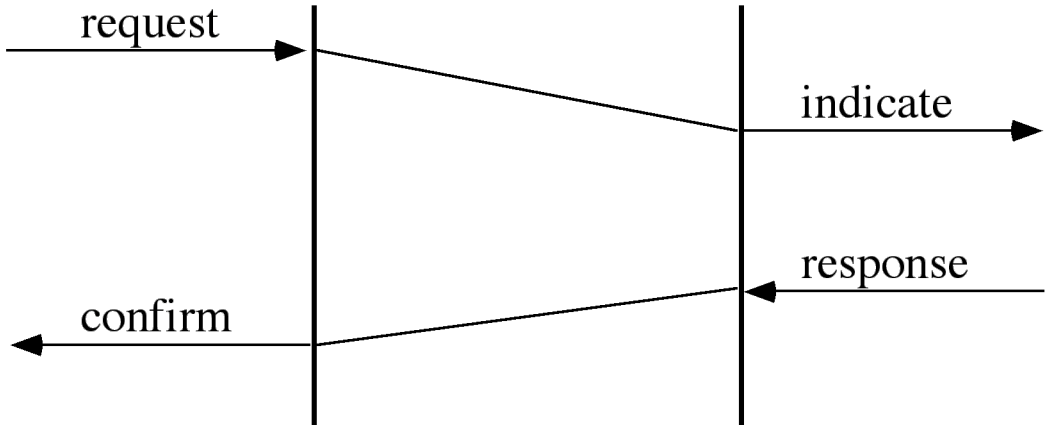
Dienst ist eine Menge von Funktionen, die einem *Benutzer* von einem *Erbringer* zur Verfügung gestellt werden

- Funktionen unterschiedlicher Abstraktion je nach Dienst
  - Dateiübertragung, Datentransport, Read, Write, …
  - Verbindungsaufbau, - abbau, …
  - SendPDU(packType, parameter)
- Service Access Points SAP



  - Dienstschnittstelle, Dienstzugangspunkt
  - im OSI-Modell für jede Schicht (X-SAP)

- Diensteprimitive nach OSI



  - Bezeichnung: Schicht-Objekt.Primitiv
  - Bsp: T-DATA.request
    N-CONNECT-indication
- Verbindungsorientiert
  - X-CONNECT.request X-CONNECT.confirm
  - X-DATA.request X-DATA.confirm
  - X-DISCONNECT.request X-DISCONNECT.confirm
- Verbindungslos
  - X-UNITDATA.request(Quell-, Zieladresse, Dienstegüte, Daten)
  - X-UNITDATA.indication(Quell-, Zieladresse, Dienstegüte, Daten)

- Dienstegüte (Quality of Service, QoS)
  - Geschwindigkeit, Bandbreite, Fehler
  - Wahrscheinlichkeit des Verbindungsaufbaus
  - Anbieter verspricht bestimmte Diensteigenschaften
- Dienstenutzer erwarten bestimmte Eigenschaften
  - QoS an der Benutzungsschnittstelle => Akzeptanz des Dienstes
  - Medien und Interaktivität bestimmen Anforderungen

> Die Dienstegüte ist eine Menge quantitativer Kenngrößen, die die Eigenschaften eines Dienstes beschreibt.

- Abbildung an Schnittstellen
  - Programmabsturz, Rauschen -> Bitfehler
  - Knacken, Krachen -> Paketverlust
  - Bildrate -> Durchsatz, Ruckeln -> Jitter

1.3.1 Dienstegüteparameter

1.3.1.1 Verkehrscharakteristik

- Übertragungsgeschwindigkeit, Bitrate, Bandbreite?
- Schwankungen der Bitrate
- Problem mit Bitrate: nicht alles bitseriell
- Router, Puffer, Protokolle etc.
- Paketrate, Paketgröße, Paketoverhead
- Allgemein:

> **Durchsatz** ist das Verhältnis der Größe einer SDU zu der Zeit, bis die nächste SDU übertragen werden muß (kann).

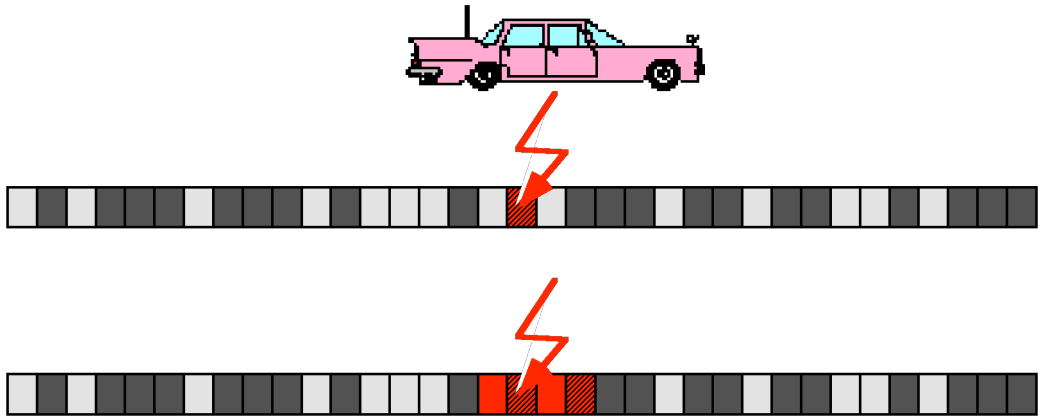
- Beispiele
  - 1 bit / (1 s/64.000) = 64.000 bit/s = 1 byte / (1 s/8.000)
  - 1500 byte / (1 s/100) = 150.000 byte/s = 1.2 Mbit/s

- statistische Angaben problematisch
  - Zeitraum der Messung unbekannt/unterschiedlich

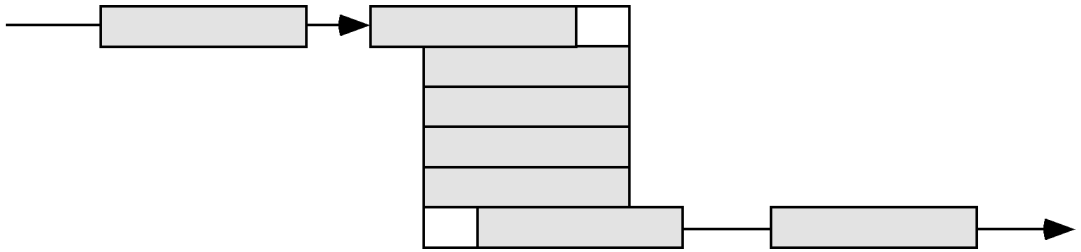
  => Leaky Bucket etc.
- Transit-Verzögerung (Delay)
  - Zeit zum Durchlaufen des Netzes und der Übertragungsprotokolle
  - pro SDU
- Verzögerungsschwankung (Delay Jitter)
  - Schwankungen der Durchlaufzeit
- Besser:
  - Mittelwert (Erwartungswert) = Delay
  - Schwankung (Varianz) = Delay-Jitter
- Oder:
  - minimale und maximale Verzögerung einer SDU
- Anzahl SDUs pro Zeiteinheit schwankt
  - Kompression
  - Burstiness

1.3.1.2 Zuverlässigkeit

- Fehlerarten
  - Bitfehler einzeln oder als Burst



  - Paketverlust (Prüfsumme falsch, Pufferüberlauf)



- Fehlerwahrscheinlichkeit
  - Bitfehler $P(F_B)$ aus Signal-Rauschverhältnis
  - $P(F_{PDU}) = P(F_B) * Länge_{PDU} + P(Pufferüberlauf)$

- Fehlererkennung
  - Paketverlust, Paketduplizerung oder Datenkorrumpierung
  - Prüfsummen auf mehreren Schichten
  - Sequenznummern
- Zuverlässigkeitsinformation vom Kanaldekoder
  - Soft-Output (Viterbi)
  - Wahrscheinlichkeit pro bit: P(0), P(1)
  - Auswertung durch Mediendekoder bzw. höhere Schichten
- Fehlerbehebung durch Wiederübertragung
  - Retransmission
  - selective Retransmit
  - Bestätigungen
    - Automatisch vom Empfänger (positiv, negativ)
    - auf Anfrage vom Sender
  - Zeitüberwachung (timeout)

- Problem Retransmissions bei großen Multicastgruppen

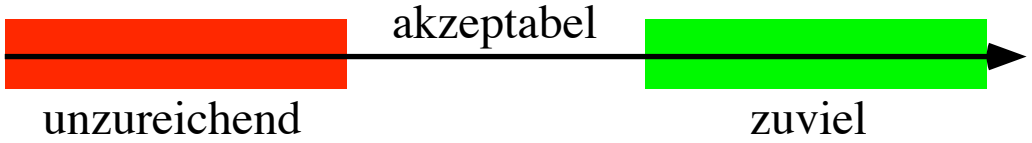
> Eine Verbindung zu n Empfängern ist **k-zuverlässig** ($0 \leq k \leq n$), falls für jede Dateneinheit gilt, daß sie bei mindestens k Empfängern angekommen ist.

  - Unbekannte Gruppen?
- k-deterministisch-zuverlässig
  - bei k spezifizierten Mitglieder angekommen
- Ablieferungszeitpunkt in Mehrpunkttopologien
- Voraus-Fehlerkorrektur
  - Fehlerkorrigierende Codes
  - Blockcodes: Reed-Solomon
  - Faltungscode
- Wechselwirkung mit Verkehrscharakteristik
  - Burst bei Retransmission
  - höhere angebotene Last durch Voraus-Fehlerkorrektur

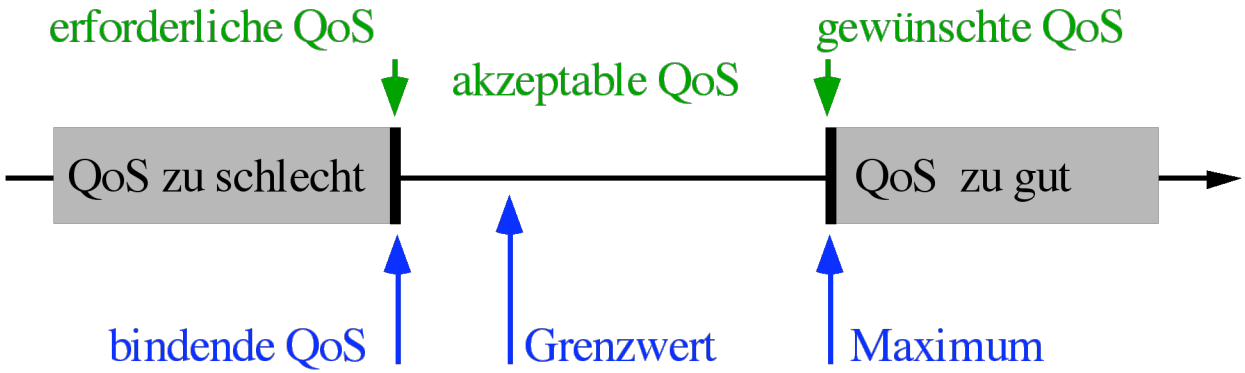
1.3.1.3 Transaktionsbeschreibung …

1.3.2 Dienstegütesemantik

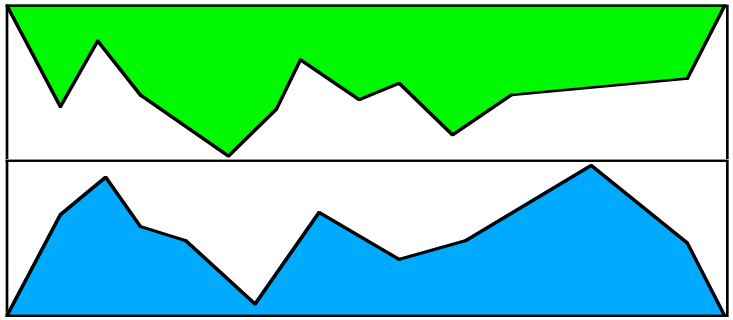
- Best Effort
  - LANs, Internet
- QoS an der Benutzungsschnittstelle => Akzeptanz des Dienstes
- Medien und Interaktivität bestimmen minimale Anforderungen
- Aufwand (Preis) bestimmt maximale Anforderung

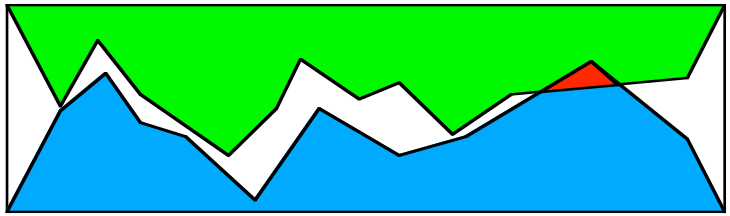


- Aus der Sicht der Kontrahenden



- Garantierte Dienstegüte
  - Pessimistische Annahmen Systemverhalten
  - Sehr zuverlässig
  - Überreservierung => schlechte Ressourcennutzung
  - Bsp. Sprachkanal im Telefonsystem





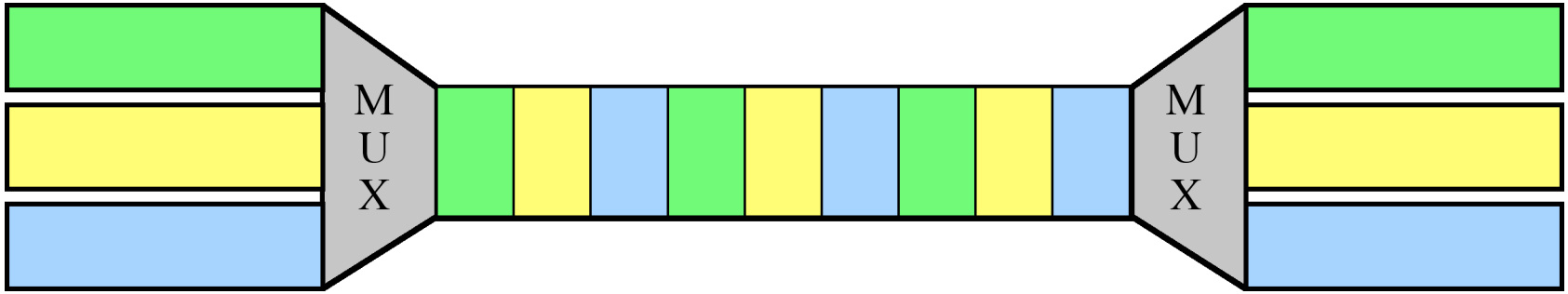
- Statistische Dienstegüte
  - Stochastische Ausnutzung der angeforderten Dienstegüte
  - keine Garantie …
  - Bsp. Verbindungsmanagement im Telefonsystem
- Dienstegütemanagement im Netz schwierig
  - Überwachung
  - weitere Zustände
  - 'Rerouting'

1.3.3 QoS-Anforderungen

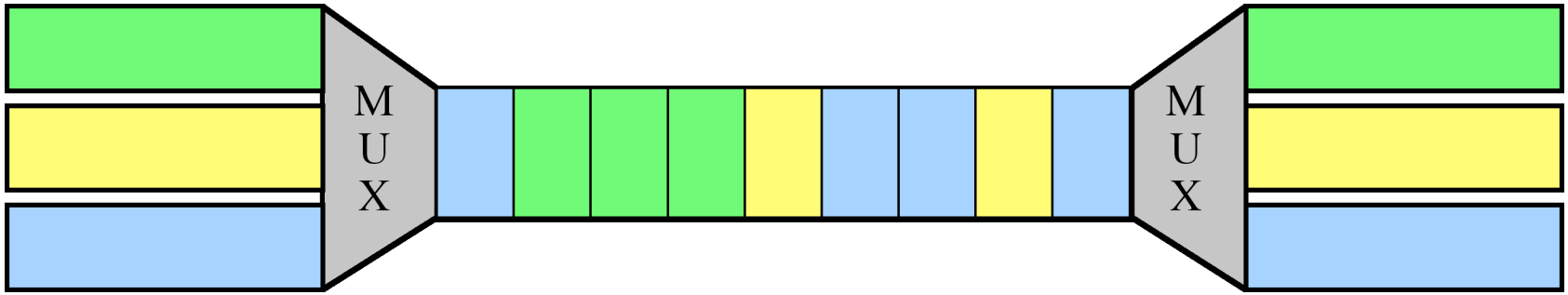
Medium	QoS-Parameter	Werte	Qualitätsvergleich
Audio	Verzögerung	≤ 200 ms	Satellitenverbindung
	Fehler	≤ 10^{-2} Pakete	Internet 10^{-1}
	Durchsatz	16 kbit/s	Telefon-ADPCM
		192 kbit/s	CD
Video	Verzögerung	≤ 200 ms	
	Fehler	≤ 10^{-2} Pakete	unkomprimiert
		≤ 10^{-11}	MPEG-komprimiert
	Durchsatz	1,5 Mbit/s	Videorekorder komprimiert
		140 Mbit/s	TV unkomprimiert
Text	Fehler	≤ 10^{-3}	
	Durchsatz	14 kbit/s	interaktiv
Grafik	Fehler	≤ 10^{-3}	unkomprimiert
	Durchsatz	28 kbit/s	interaktiv
Bild	Fehler	≤ 10^{-2}	unkomprimiert
		≤ 10^{-11}	JPEG-komprimiert
	Durchsatz	64 kbit/s	JPEG, interaktiv

1.4 Verkehrsabschätzung

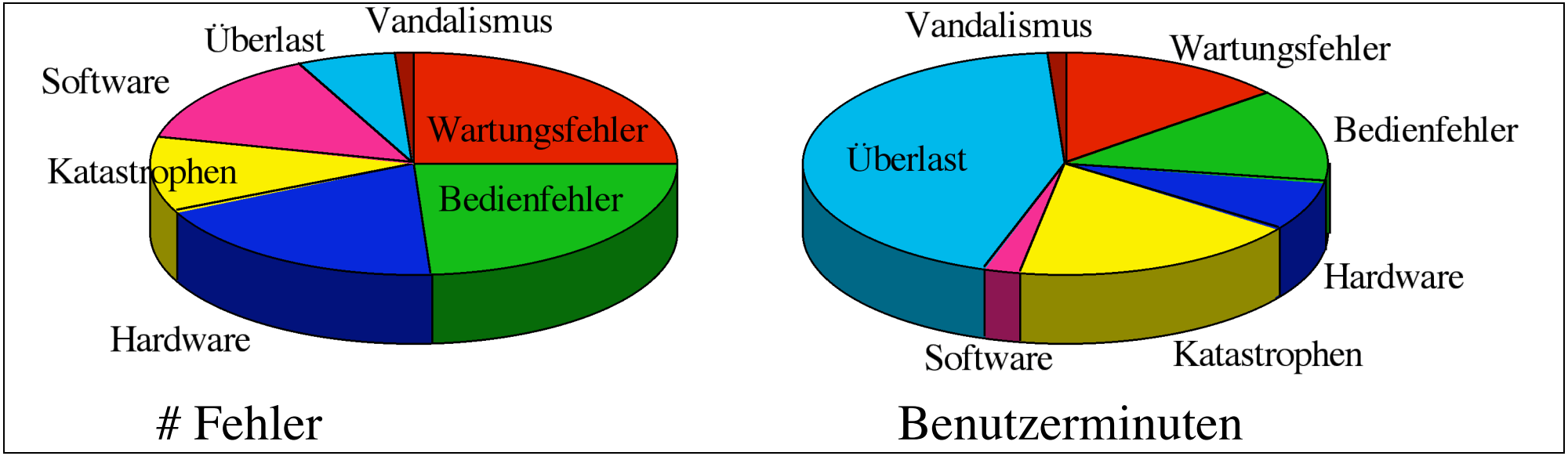
- gemeinsam genutzte Ressourcen: Leitungen, Vermittlungen
  => Multiplex
- feste Multiplex-Schemata
  - Verbindungen im Telefonsystem, ISDN-Teilnehmeranschluß
  - Ressourcenverschwendung



- statistischer Multiplex
  - Ethernet, 'call attempts' im Telefonsystem, Internet



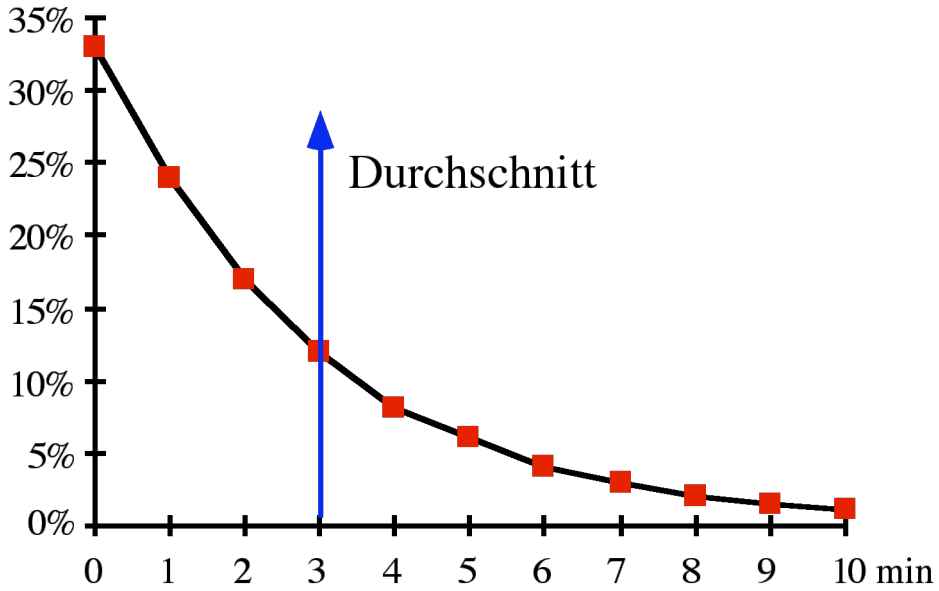
- Probleme im US-Telefonsystem 4/92 - 3/94



=> traffic engineering

- Abschätzung von Multiplex-Effekten
  - Eigenschaften der Einzelströme
  - zeitliche Kombination
  - Auslegung der Ressourcen (Durchsatz, Puffer, Prozessor, …)
- Warteschlangen -Theorie (queueing-theory)

- Annahme: wichtige Verteilungen mit Poissonverteilung
  - negative Exponentialverteilung
  - Nachrichtenlänge
  - Ankunftsrate



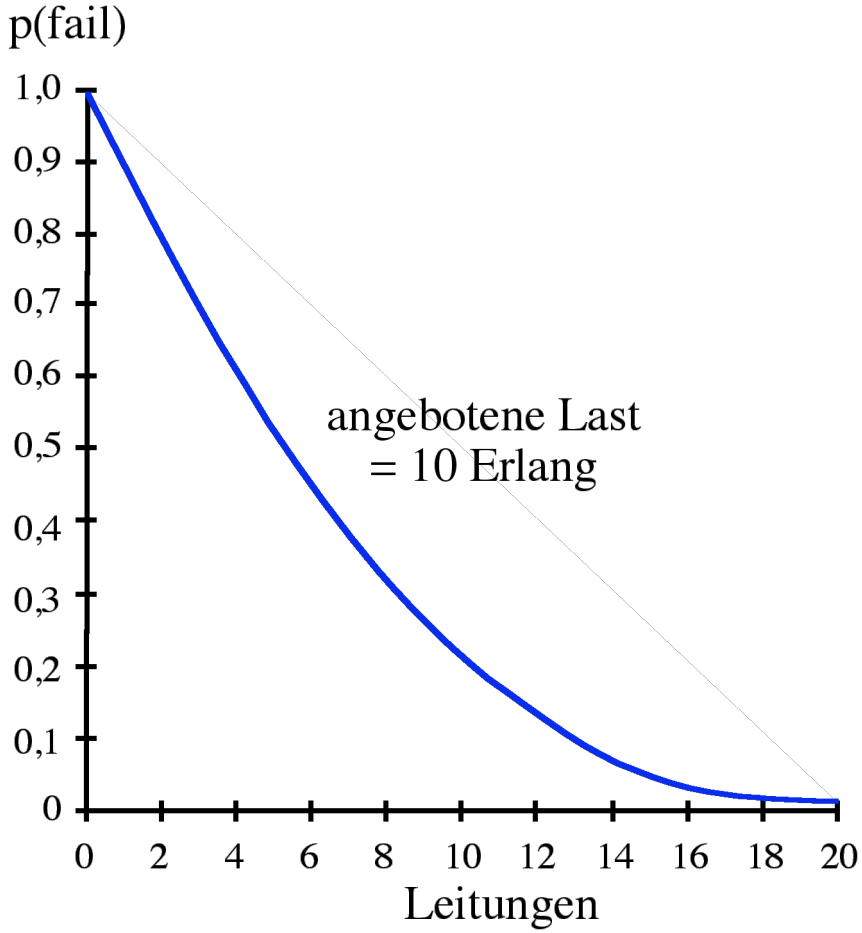
- Mittelwerte
  - λ für mittlere Ankunftsrate
  - μ für mittlere Länge/Bearbeitungsdauer

- Angebotene Last = Ankunftrate*Dauer [Erlang]
  - 100 [Benutzer] * 3 Anrufe/Stunde * 0,05 Stunden/Anruf = 15 Erlang
- A.K. Erlang 1918
  - Anzahl der Anrufe poisson-verteilt
  - Dauer der Anrufe poisson-verteilt

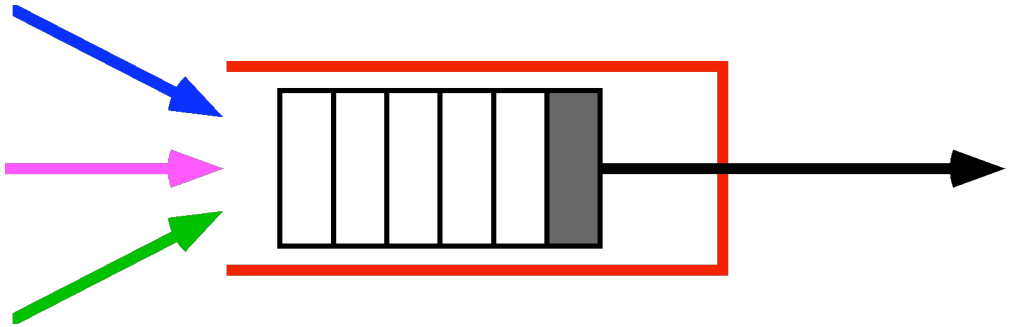
=> Blockierwahrscheinlichkeit

$$B(a,n) = \frac{a^n}{n! \sum_{i=0}^{n} \frac{a^i}{i!}} = \frac{a * B(a,n-1)}{n + a * B(a,n-1)}$$

- a angebotene Last in Erlang
- n Leitungen
- iterieren bis B(a,n) klein genug



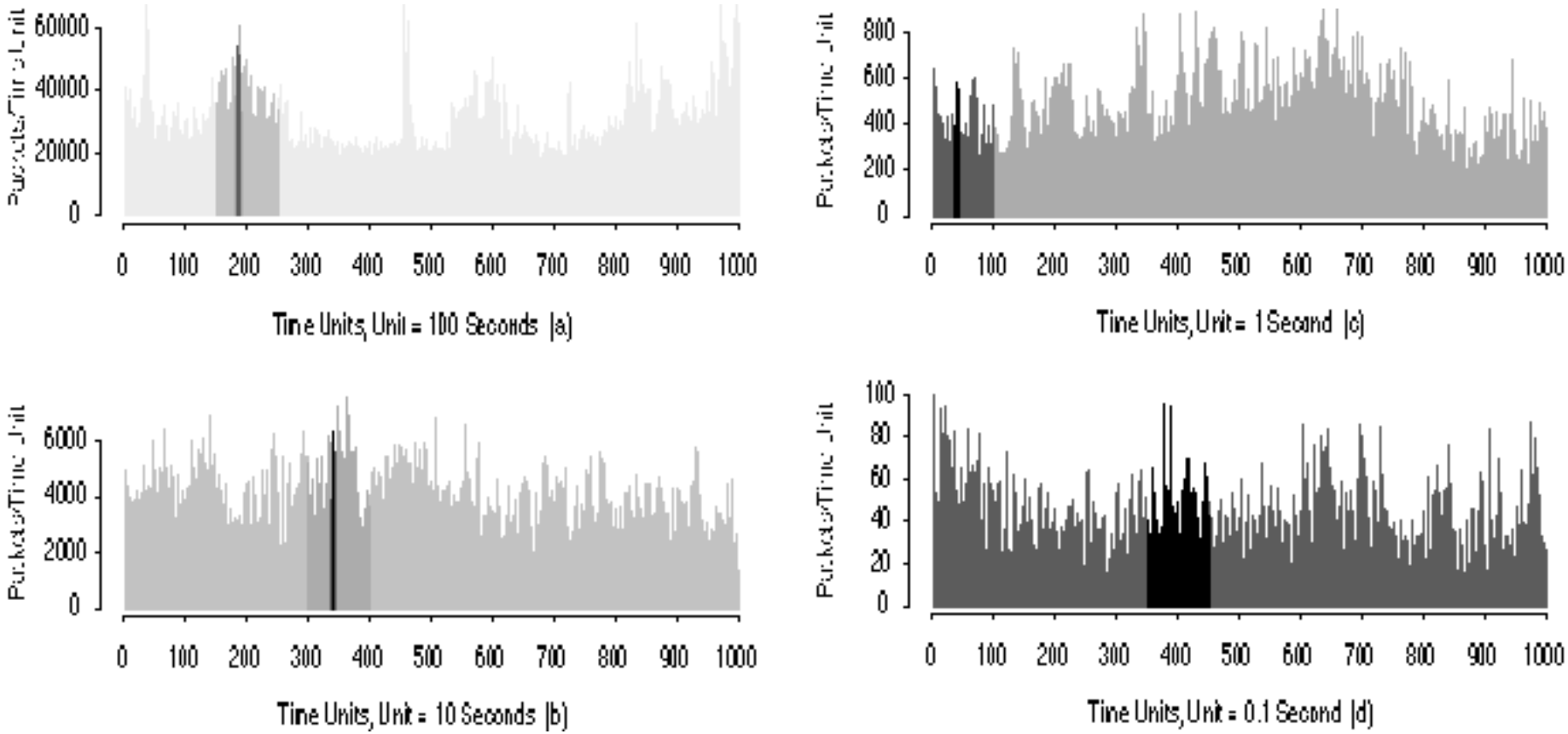
- Warteschlangen
  - ankommende Nachrichten kommen in Queue
  - werden einzeln abgearbeitet

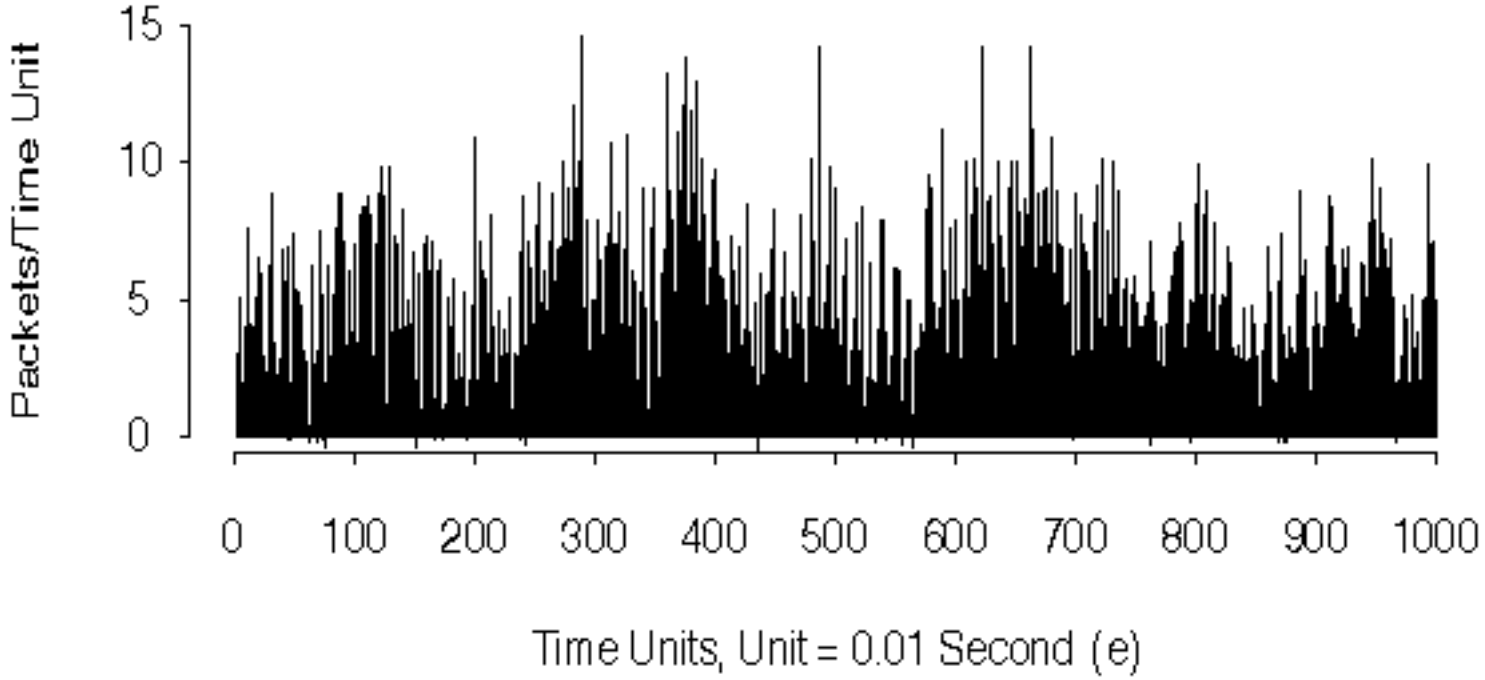


- Anzahl Schlangen vs. Anzahl Verarbeitungseinheiten
- Warteschlangenmodelle
  - Ankunftsmodell/Dienstdauermodell/# Server - Länge Warteschlange
  - M = gedächtnisloses Modell (memoryless, Poissonverteilung)
  - D = deterministisches Modell
  - G = allgemeines (general) Modell
  - M/M/1 abgeforscht
  - G/G/m-k?

- Sind Ankunftsintervalle poissonverteilt?
  - Poisson: große Anzahl unabhängige Benutzer
  - Videokompression, MPEG
  - TCP's Vegas-Algorithmus reagiert auf Verstopfung
  - FTP und neue 'Releases'
- Ist Servicedauer poissonverteilt?
  - FTP-Server für wenige Dateien
  - Telefonauskunft
- Sind Verkehrsquellen unabhängig?
  - Protokolle erzeugen Abhängigkeit (Request-Response)
  - Überlast, Verlust und Paketwiederholungen
  - besondere Ereignisse (Fußballspiel, Sylvester)
- Fraktales Stromverhalten beobachtet
  - MPEG-Video ohne VBV [Garrett, Willinger, 1994]
  - Verteilung in verschieden langen Zeiträumen ähnlich

- On the Self-Similar Nature of Ethernet Traffic
  - Messungen über mehrere Jahre in echten Netzen
  - Bellcore Morristown Research Centre
  - [Leland, Taqqu, Willinger, Wilson, 1994]

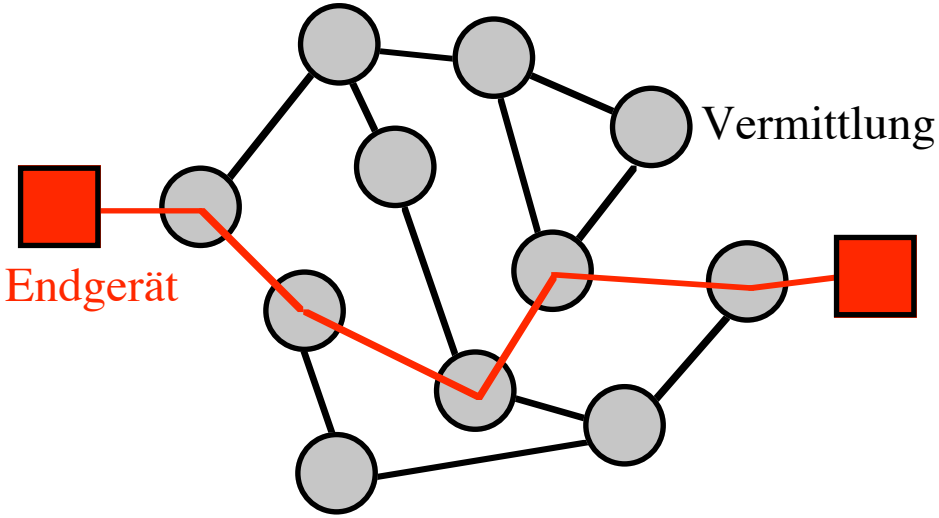




- Selbstähnlich
  - gleichgültig wie stark vergrößert: Bilder ähnlich
  - burst, flacheres Gebiet, burst
  - Poisson-Modell unzutreffend für Ethernet
- Folgerungen/Maßnahmen
  - selbstähnliche Quellen konstruieren
  - Meßmethoden für 'Burstiness' entwickeln
  - Schluß vom Gesamtverkehr auf Quellen möglich

1.5 Verbindungen und Topologie

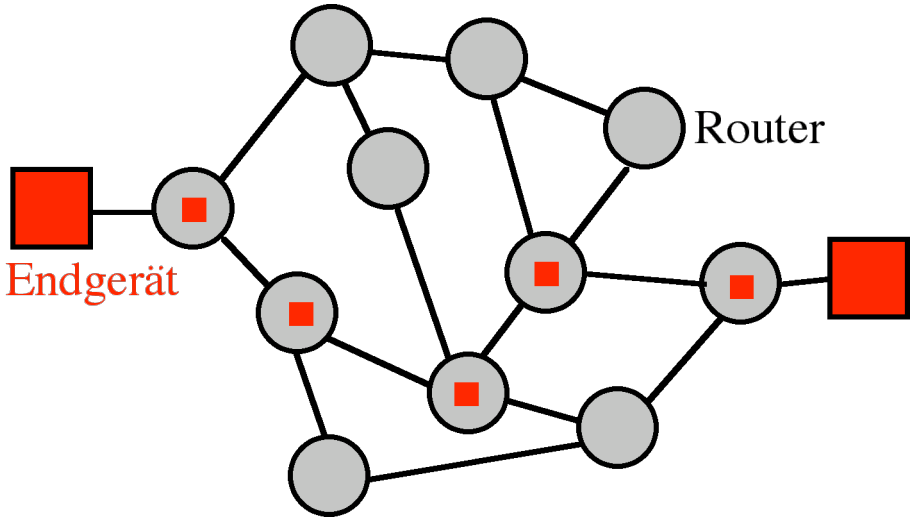
- Verbindung: Abstraktion der Shannon'schen Kanäle
  - Kontext der Informationsübertragung
  - fest
  - statisch geschaltet
  - dynamisch geschaltet
  - virtuell



- Verbindungslose Dienste?
  - Dateneinheiten enthalten komplette Zielbeschreibung
  - Erreichen des Zieles ohne Kontext
  - implementierungsbezogene Definition

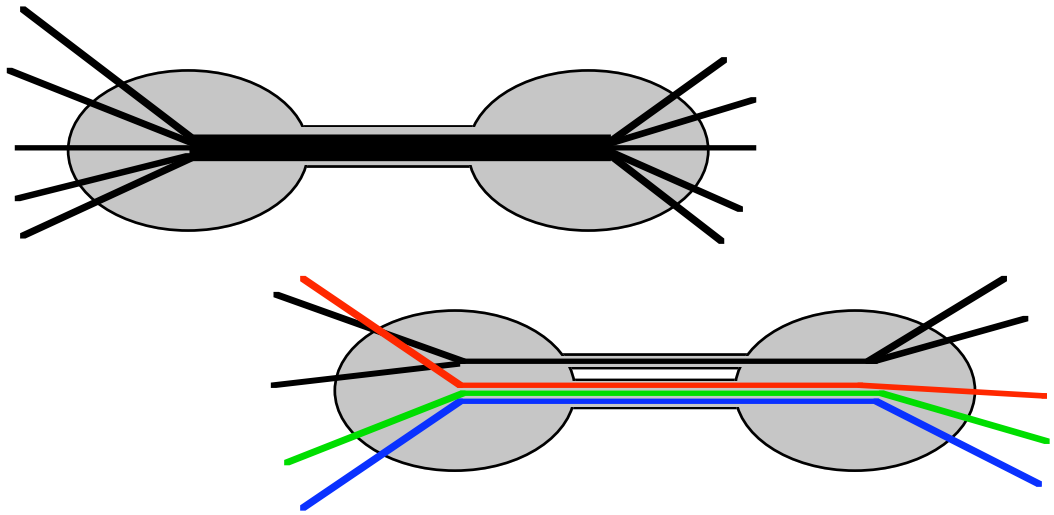
- Technik verbindungsloser Dienste
  - Weglenkung bei jeder SDU neu
  - ATM wird aber auch als verbindungsorientiert bezeichnet
  - Router verwenden Adress-Cache zur Beschleunigung

  => Eintrag in caches entlang des Pfades = Kontext



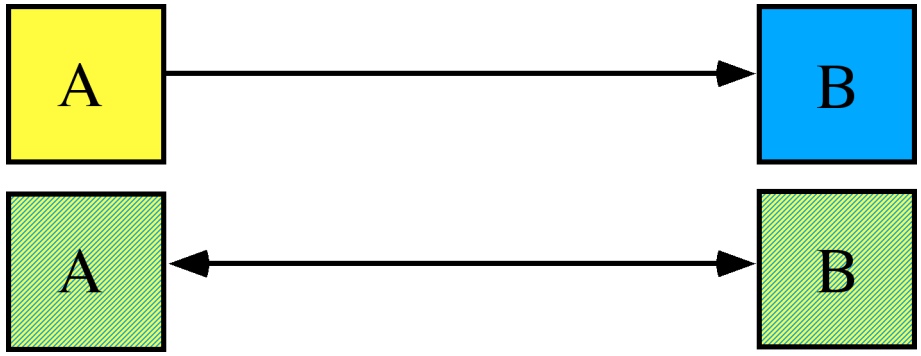
- Benutzungssemantik
  - Warten auf Bestätigung durch Benutzer
  - Kontext um Antworten einzuordnen (Listen von Requests)
  - Protokolle auf höheren Ebenen

- QoS in verbindungslosen Diensten?
  - Identifikation von Paketen als Teil eines 'flows'
  - Reservierung für Flows
  - Reservierungsprotokol

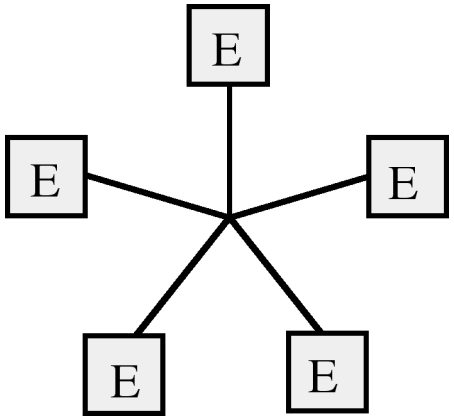


- Verkehrsklassen (Internet, ATM, …) brauchen Kontext
  - konstante Bitrate
  - variable Bitrate
  - verbleibender Verkehr
- Leitungsvermittlung und Paketvermittlung
- hard state und soft state

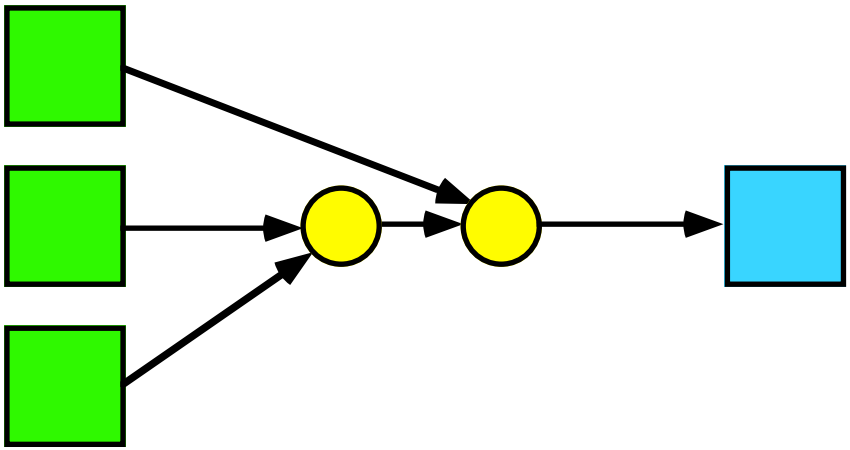
- Verbindungstopologie
  - gerichtet (Funk, Fax, ftp, Telefon, …)
  - bidirektional (ISDN, V.24, telnet, …)
  - Richtungsumschaltung möglich



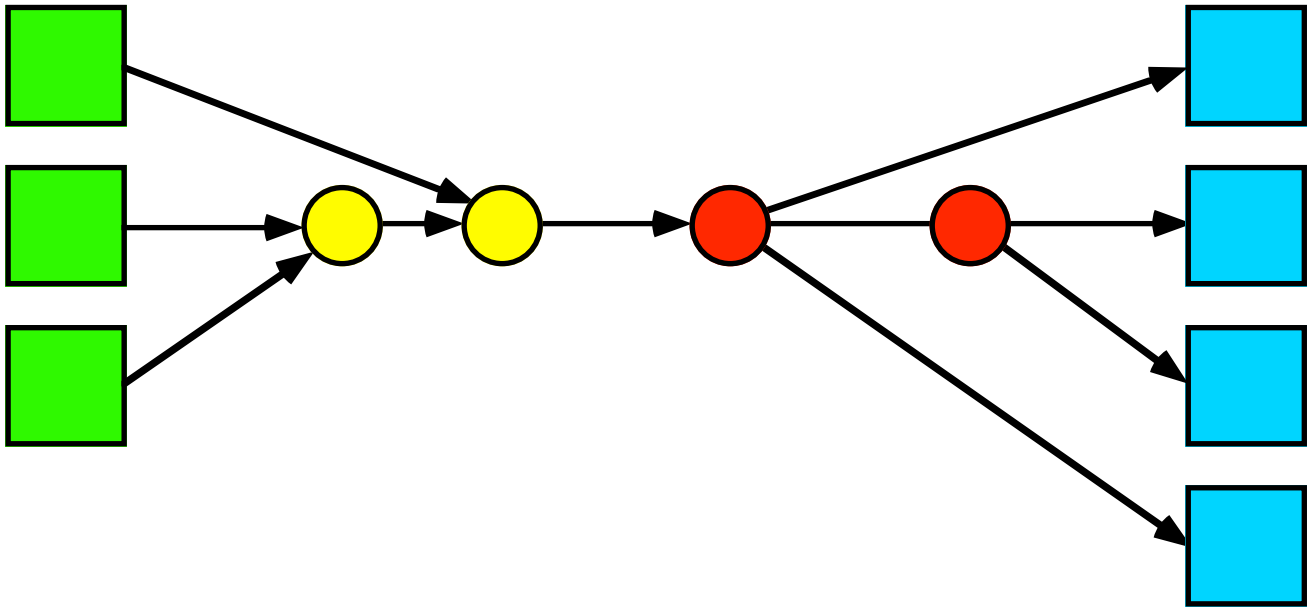
- asymmetrische Verbindung
  - unterschiedliche Dienstegüte je nach Richtung
- Zweipunkt und Mehrpunkt



- Unicast n:1

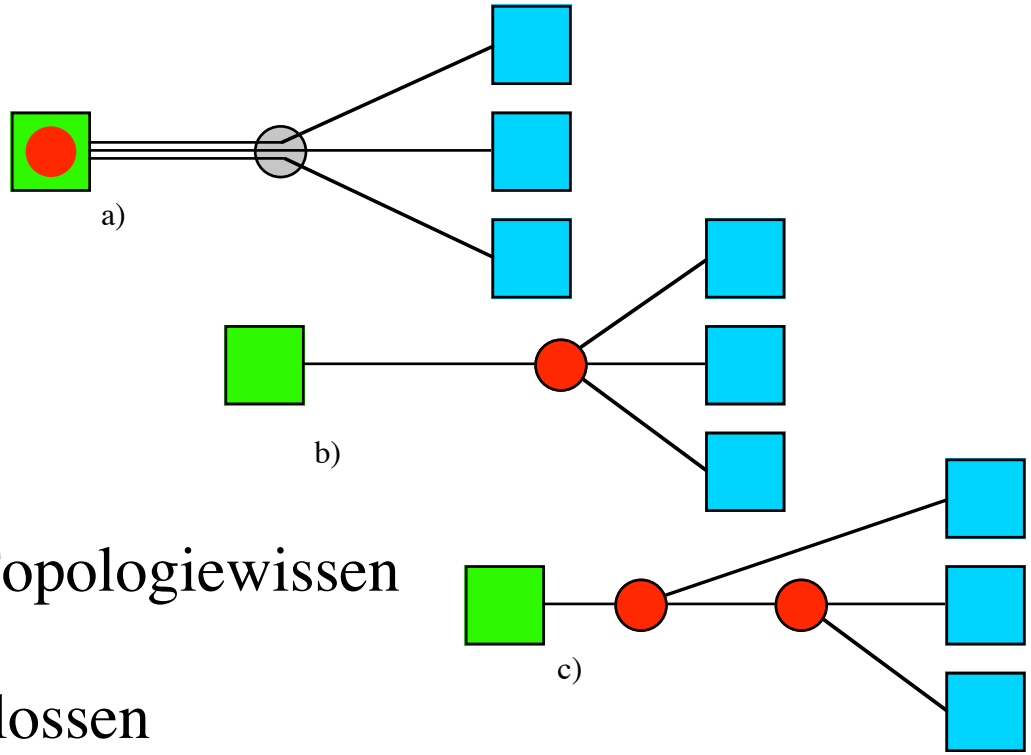


- Multicast n:m
  - n-facher Mehrfachzugriff auf Verbindung

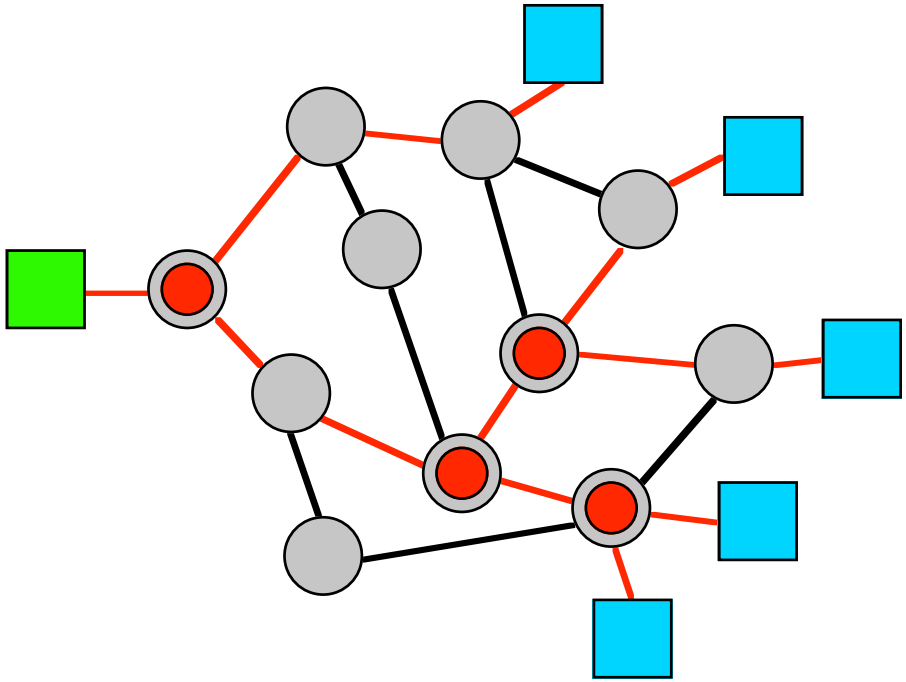


- Nachrichtenreplikation
  - in der Quelle
  - im Netz
  - explizit oder implizit

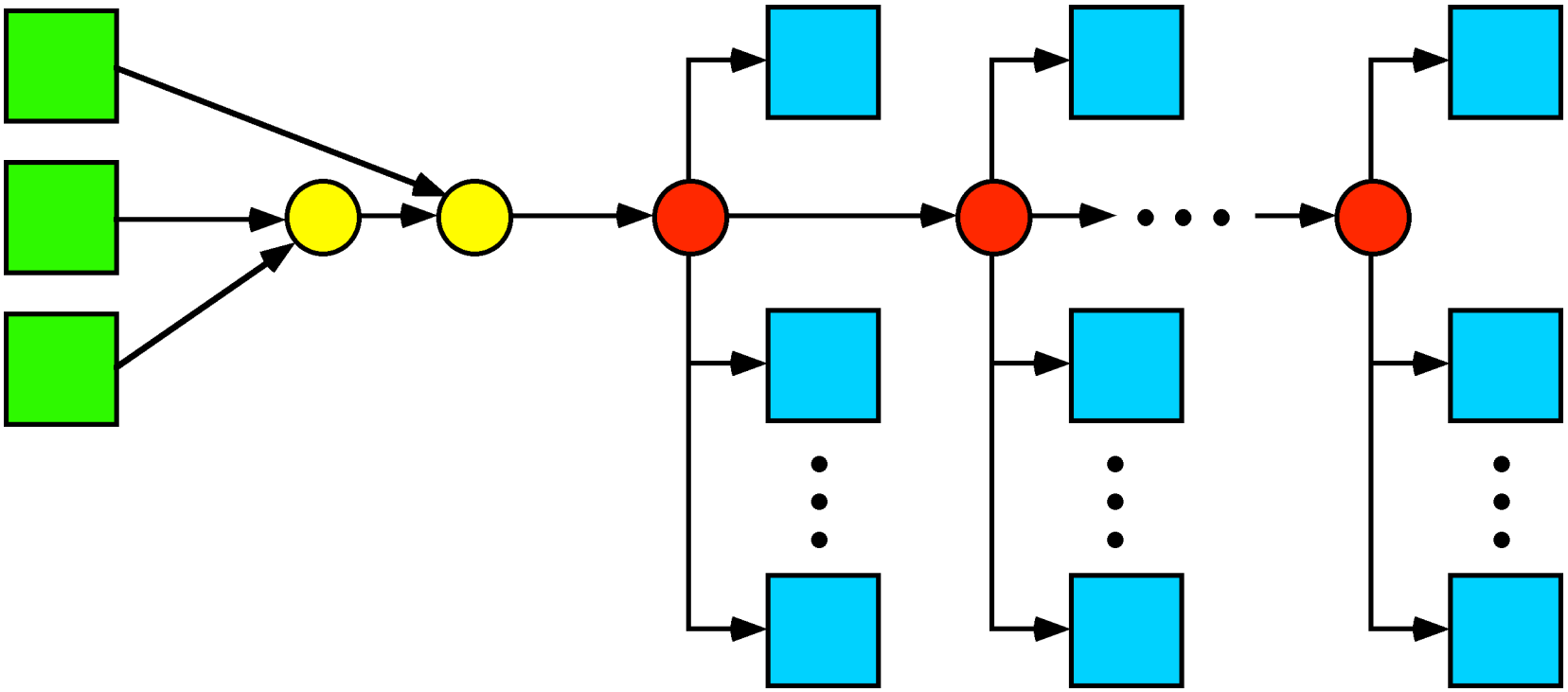
- Gruppenverwaltung
  - statisch/dynamisch
  - deterministisch = vollständiges Topologiewissen in einem Gerät
  - Zugangsregeln: offen oder geschlossen



- Mehrpunkttopologie im Netz
  - normal in LAN's (Ethernet, Token-Ring, PhoneNet, …)
  - Mehrfachadressen
  - explizite Replikation in vermaschten Punkt-zu-Punkt-Netzen (z.B. Internet)
  - Mehrfachversand



- Broadcast
  - AppleTalk NBP, Ethernet ARP
  - verteiltes Rechnen



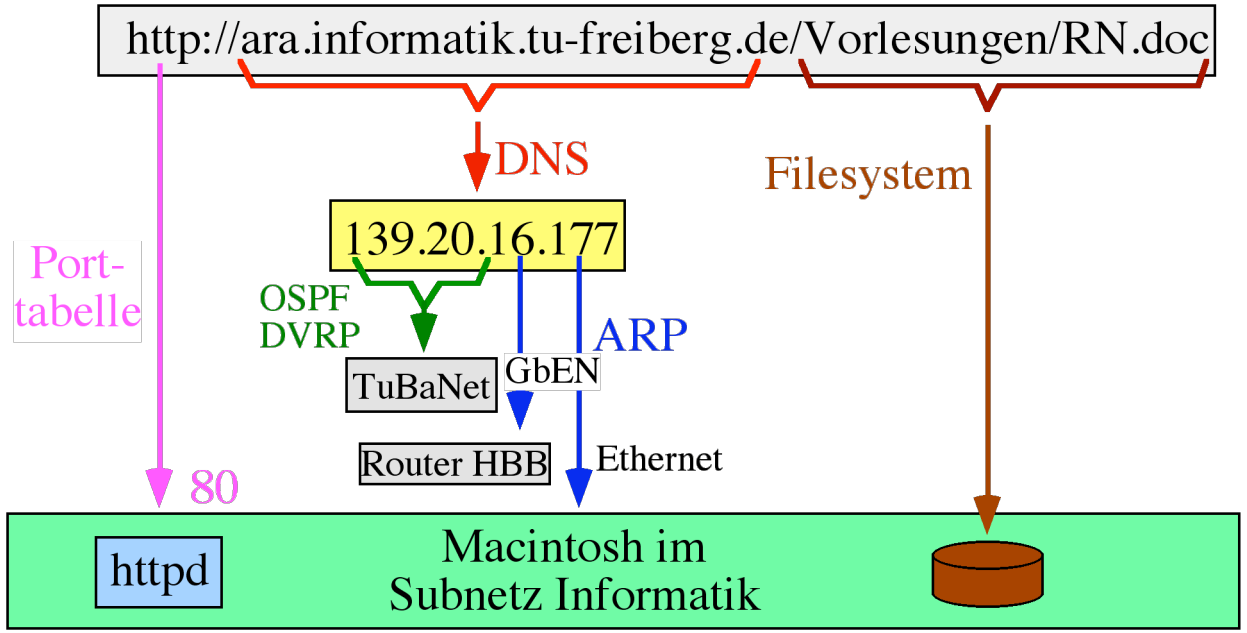
- Skalierbarkeit
  - Mehrstufigkeit
  - Begrenzung der Reichweite
  - MBone, Usenet-News

- Multiplexen und Mischen
- Mehrfachzugriff verursacht Überlagerung
  - Sequentialisierung (Multiplexen)
  - Mischen
- Multiplexen
  - feste Kanalzuordnung (Telefonsystem, SONET, …)
  - Verfahren mit dynamischer Zuordnung
  - Bsp. Unterhaltung, Konferenz (soziale Protokolle)
- Beispiele dynamischer Verfahren
  - Reservierung mit Anmeldung
  - Token
  - Aloha
  - CSMA mit Varianten
- Mischen = Konferenzschaltung

1.6 Namen und Adressen

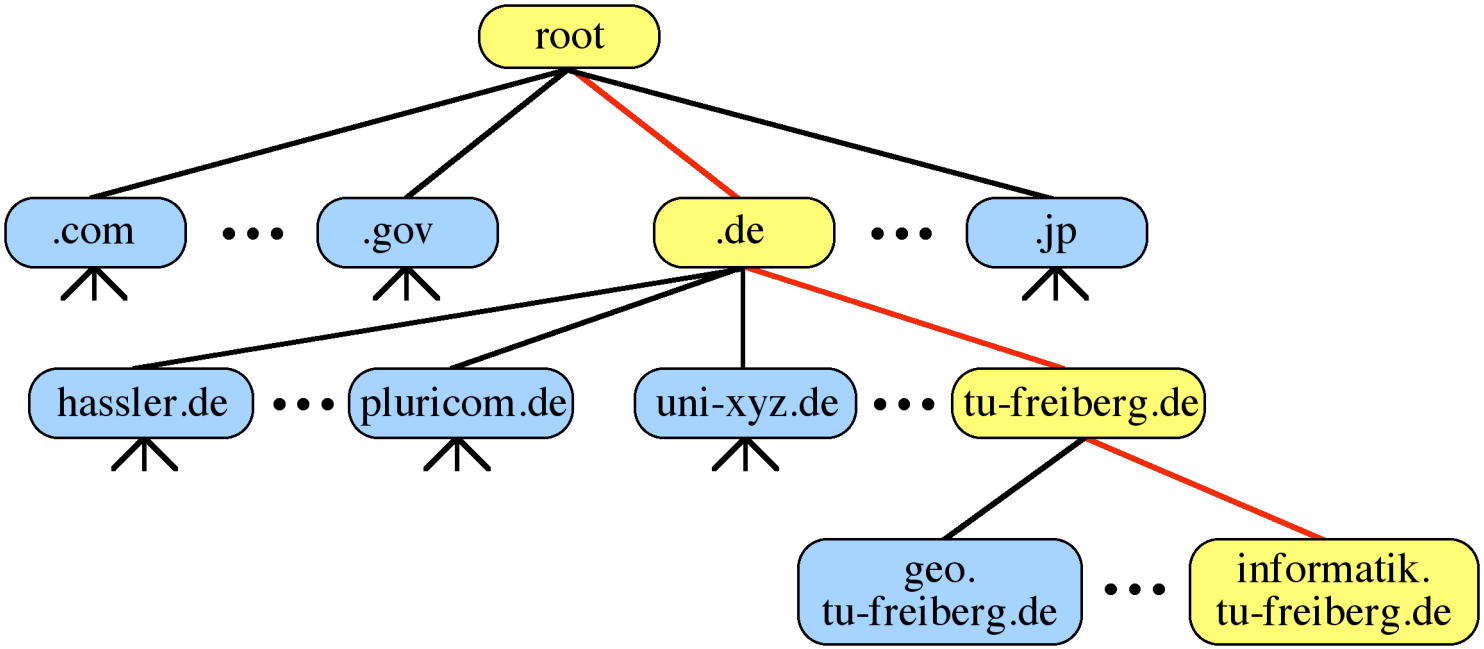
- Endpunktbezeichnung
  - Personen, Dienste, Geräte
  - Postanschrift ist Tupel
  - (Name, Bestimmungsort, Leitangaben, Zustellangaben)
  - menschliche Bezeichnung vs. Weglenkungsinformation
- flache und hierarchische Adressen
  - Bsp. Pass-Nummer vs. DNS-Name
- Besser: 3 Abstraktionsebenen
  - Kommunikationsadressen
  - Netzwerkadressen
  - Anschlußpunktadressen
- Kommunikationsadressen
  - von Menschen benutzbar
  - konzeptuelle Beschreibung
  - (Dienst/Person, Beschreibung)
  - beschreiben Person oder Konzept
  - Bsp. URL

- Netzwerkadressen
  - technische Beschreibung des Kommunikationszieles
  - (Länderziffer, Ortsnetz, lokale Nummer)
  - Bsp.: Telefonnummer, IP-Adresse
- Anschlußpunktadressen (Service Point Access)
  - eindeutig in einem Kontext
  - Hardware-Beschreibung
  - (Gerätenummer, Diensteidentifikation)
- Adreßbindung = Abbildung auf untere Klasse



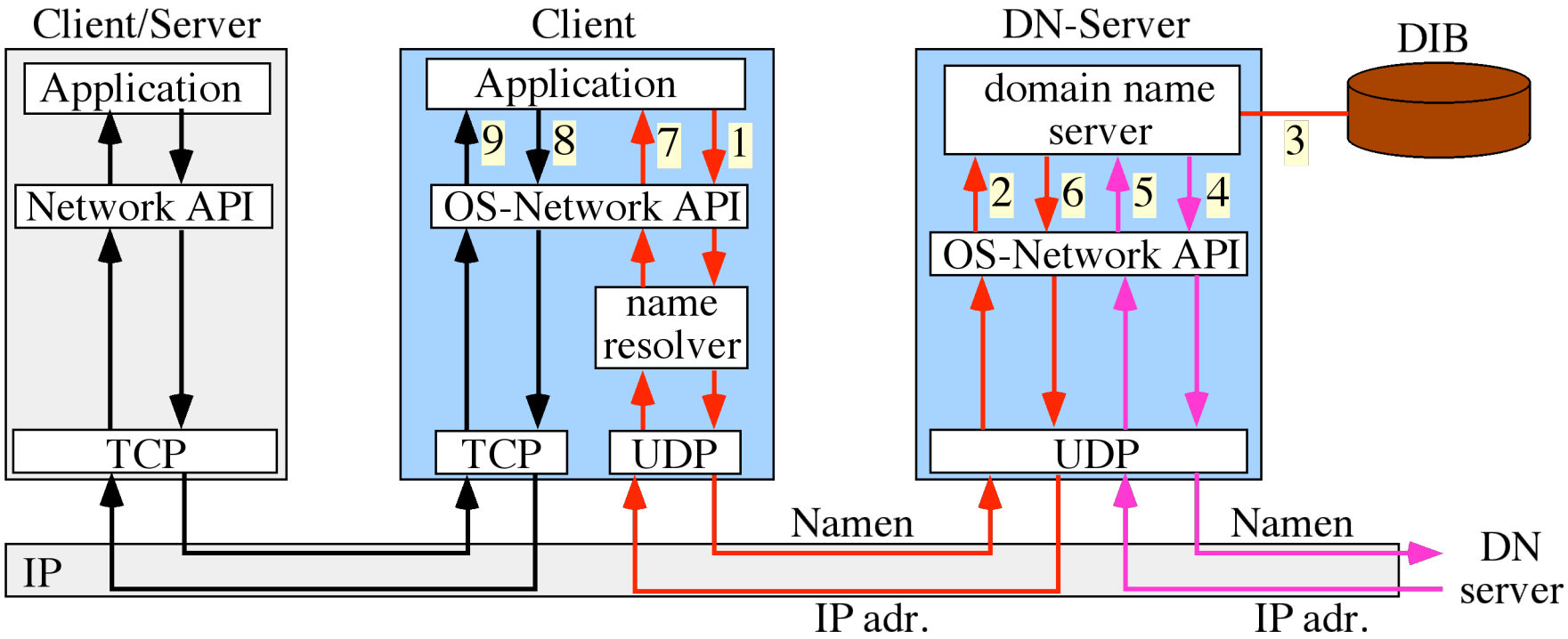
- Kommunikationsadressen
  - (Dienstename, Kontextname, Gerätename)
  - konzeptuelle Beschreibung
  - verständlich für Menschen
- URL Uniform Resource Locators
  - Dienst://Rechnername[/Pfad][/Datei]
  - RFC1738
  - http://rr.informatik.tu-freiberg.de/rr.html
  - ftp://ara.informatik.tu-freiberg.de/vorlesungen/RN.doc
  - Ausnahme: mailto:mueller@informatik.tu-freiberg.de
- Dienst: ftp, http, …
- Rechnernamen sind DNS-Namen
- Pfad und Dateiname entsprechend OS-Konvention
- Oft Dienst auch im Namen wiederholt
  - http://www.joeboxer.com
  - ftp://ftp1.apple.com/
  - besser: <dienst>://info.itu.ch

- Domain Name System (DNS) Namen
  - Bsp: info.itu.ch, altavista.digital.de
  - Namenssyntax RFC1034
  - hierarchisches System

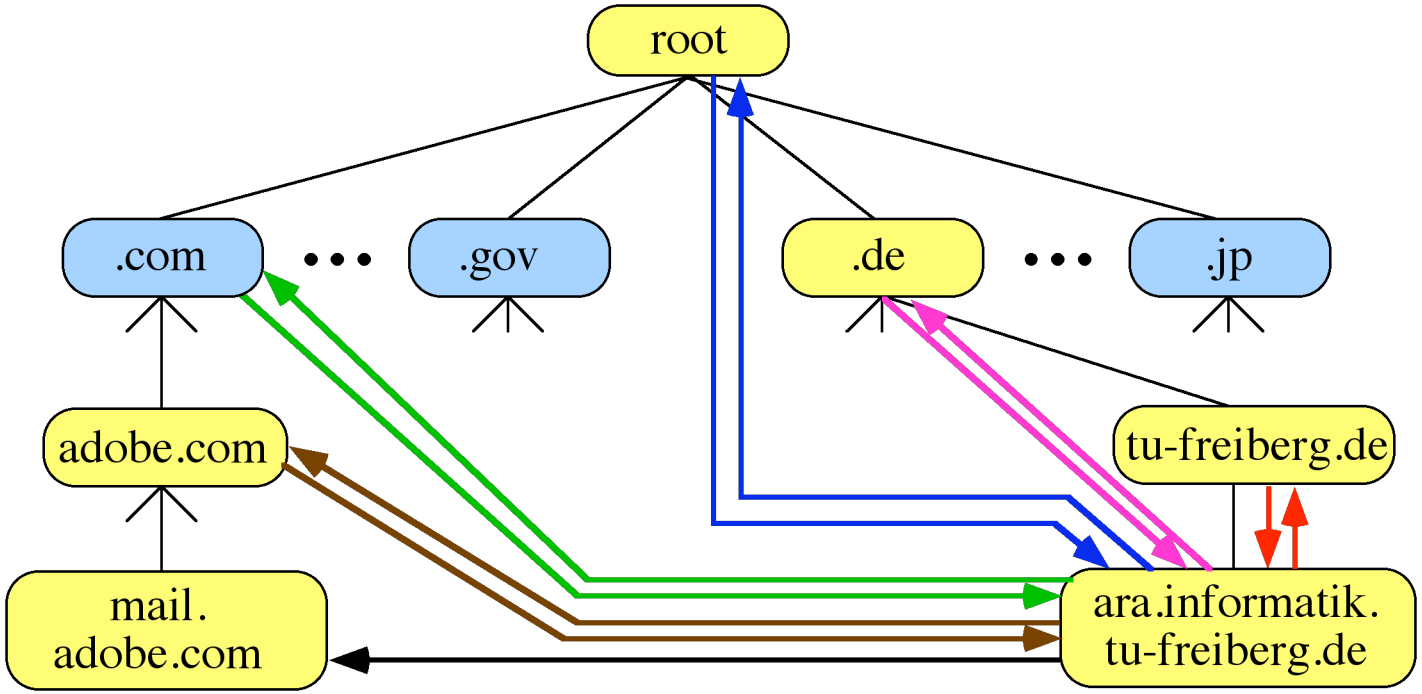


- Länderkennzeichen oder .com, .org., .net, .gov, .mil, .edu, .int
- Dauerstreit um Domainnamen
  - McDonalds.com, bau.de
- Neue Domains: .pers, .store, …

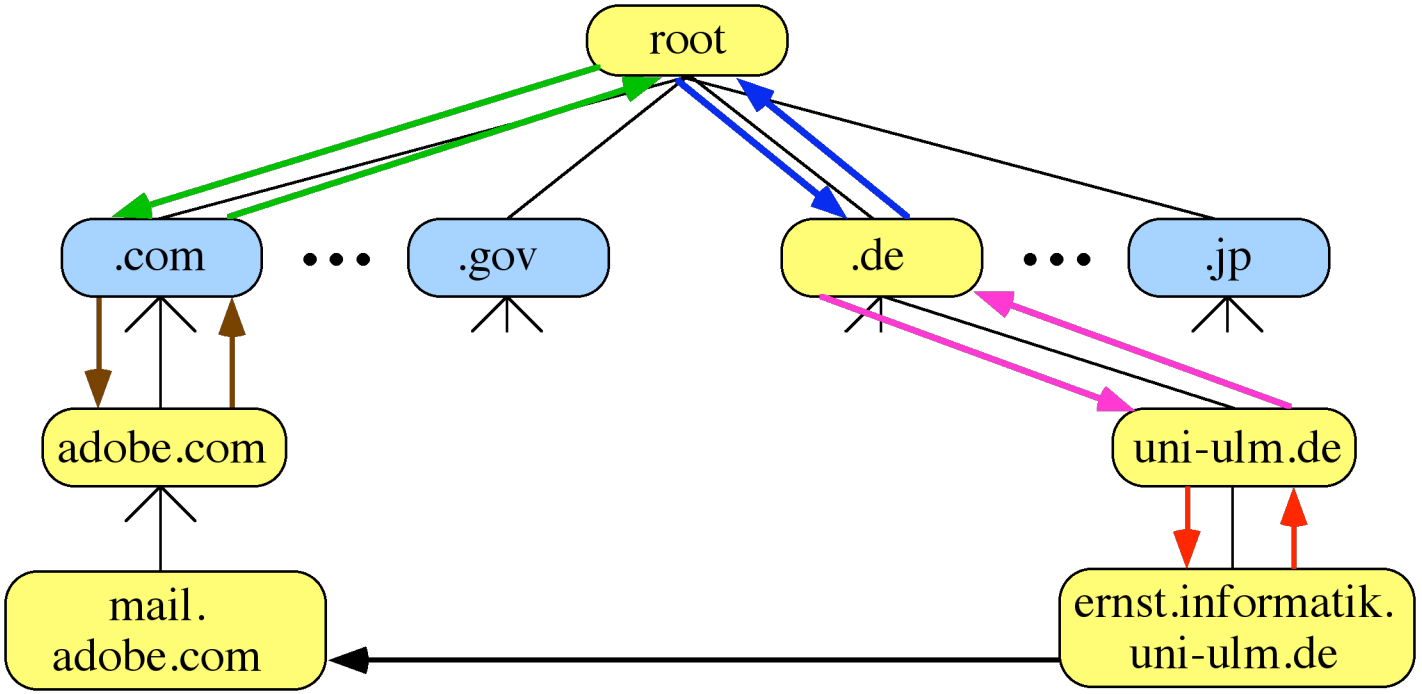
- DNS-Adressbindung
  - Anfrage an wohlbekannten Nameserver (IP-Nummer)
  - dynamisch: erst beim Verbindungsaufbau
  - Auskunft vom DNS aus den Servern statisch
  - statische Registrierung
  - Domain Information Base (DIB)



- name resolver
  - Software im Klientenrechner
  - verwendet TCP/IP
- Verteilter Algorithmus in RFC1035
  - Adressbindung DNS-IP
  - Lokalität der Namensbindung
  - nichtlokale Anfragen: referral
  - iterativ (Nameserver antwortet mit anderer Serveradresse)



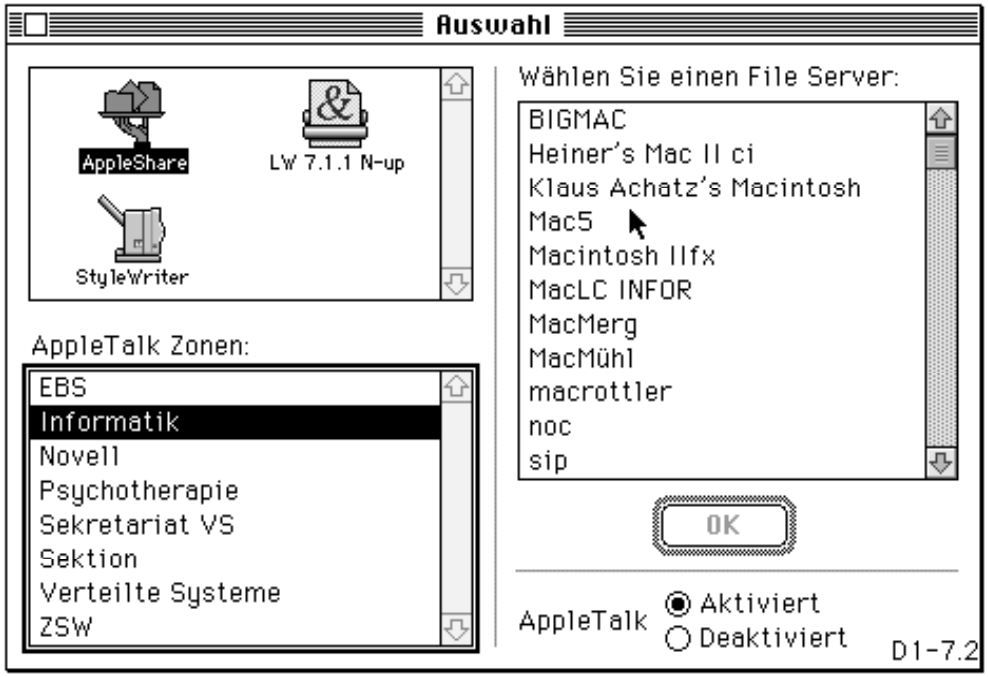
- rekursiv (Server fragt andere DNS-Server)



- name cache

• ITU X.500
  - Objekt hat Attribute (Attributtyp, Wert)
  - Directory Information Tree
  - Search Port (yellow pages), Read Port (white pages)
  - Welt, Nation, Organisation, Unterorganisation, Personen

- AppleTalk Namen
  - <object>.<type>.<zone>
  - object: frei wählbare Zeichenkette für Knoten
  - type: Dienst, z.B. LaserWriter, AFS-Server
  - zone: logisches Netzwerksegment
- Name Bindung Protocol NBP
  - Resource-Lookup
  - Abbildung direkt auf Anschlußpunktadresse
  - =‚LaserWriter‚* findet alle lokalen PAP-Drucker

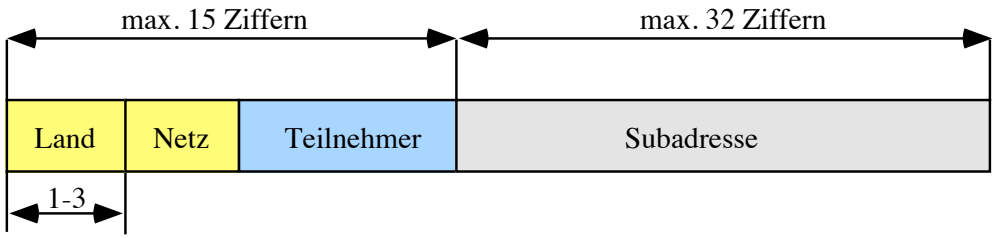


- Netzwerkadressen
  - technische Beschreibung des Kommunikationszieles
- Nummern-Tupel
  - Dienst (oft implizit durch Geräteauswahl)
  - Kontext (Subnetz oft impliziert)
  - Subnetz {Subnetz}
  - lokals Gerät/Leitung
- Telefonsystem

  [<Art>] [<Land>] [<Vermittlung>] [<Teilnehmer>] [<Nebenstelle>]
  - Art = {externlFernverbindunglinternationallDienst}
  - lokal unterschiedlich
  - Anpassung bei mobilen Geräten oft schwierig (siehe GSM)
- Beispiel Telefonnummer

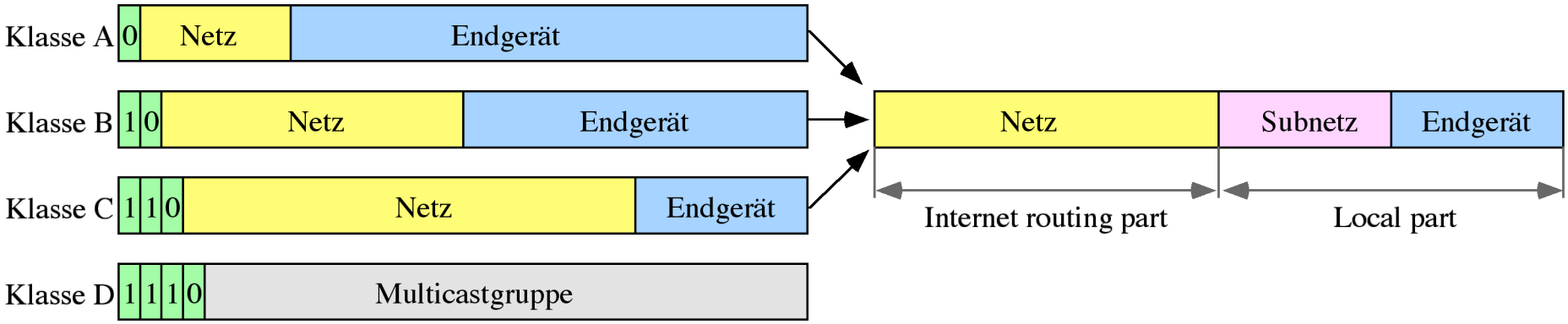
-				3939	in der Universität
lokal			39	3939	in Freiberg
Fernverbindung		3731	39	4146	in Deutschland
international	49	3731	39	4146	im Ausland

- statische Aufteilung in USA und Kanada: 415-653-9516
  - area code bzw. besondere Dienste (800, 900) dreistellig
  - Vermittlungscode dreistellig
  - Teilnehmer vierstellig
  - PBX (Centrex, virtuelle Nebenstellanlage) in Nummernplan integriert
- Deutschland
  - Vermittlungsbereich 2-5-stellig (89, 351, 3731, 34491)
  - Rufnummer
- ITU Norm E.164
  - abgeleitet von Telefonnummern
  - Länge durch Wahlbewertungshardware eingeschränkt
  - Netzkennziffer (Bsp: 161, 171, 172, 177, …)
  - Subadresse
    - Endgeräteauswahl im ISDN
    - Tln. in Nebenstellenanlage
- E.123 Schreibweise: +49 3731 39 3939
- RFC 3966 tel: +49-3731-39-3939



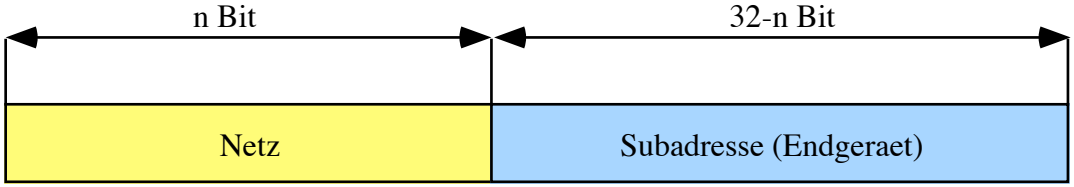
oder (03731) 39 3939

- Internet Protocol: IP-Adressen
  - IP Version 4 Header: 32 Bit-Adressen
  - (netid, hostid)
  - (netid, subnetid, hostid): 139.20.16.177 (8B1410B1)



- Adressklassen
  - Klasse identifiziert Semantik und Format
  - Punktnotation zur Lesbarkeit
  - Klasse F reserviert
- Probleme
  - zu wenig verwendbare Adressen
  - Größe der Routingtabellen begrenzt
  - schnelle Cache-Invalidierung durch große benutzte Adressmenge
  - hierarchische Weglenkung steigert Effizienz

- CIDR: Classless Inter-Domain Routing [RFC1519]
  - flexiblere Längeneinteilung der IPv4 Adressen -> mehr Netze
  - Zusammenfassung von Netzbereichen in Routern (Hierarchie)
- Schreibweise: A.B.C.D/n
  - Präfix: n Bits (von links)
  - 139.20.0.0/16 : tu-freiberg.de
  - 139.20.16.0/24: informatik.tu-freiberg.de



- Flexible Netzwerkgrößen
  - kleine Netze möglich: z.B. /27
  - oder mehrere Class-C Netzwerke (/24) zusammenfassen: /22 etc.
- Aggregation
  - Router fassen Netzwerke mit gleichem Präfix zusammen
  - Tabellen kleiner im Speicher und in Paketen
  - longest match algorithm
- VLSM: Variable Length Subnet Mask

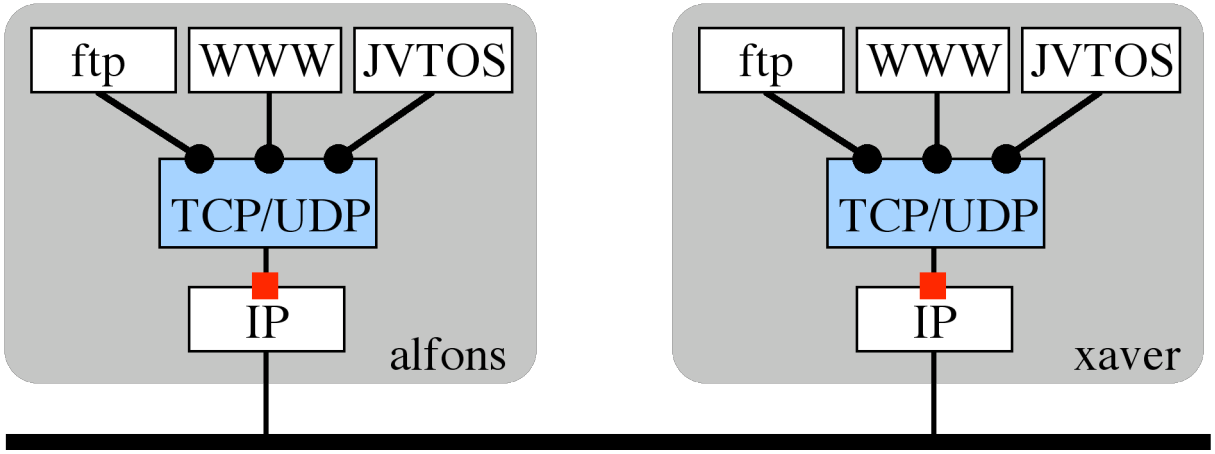
- IPv6: neuer Header
  - Quell- und Ziel-Adresse je 128 bit
  - weitere Felder (flow-label, etc) siehe Kapitel 3.5
  - 1500 bis $3*10^{17}$ Adressen / m^2
- Präfix 3 - 10 Bit unterteilt in Klassen
  - Unicast, Multicast (1/256)
  - Local-Use
  - IPX, NSAP-Adressen für ISO CLNP
  - 85% future use
- Unicast-Adressen
  - 12,5% des Adressraumes
  - 4 Stufen

3	n	m	o	p	125-n-m-o-p
010	Reg	Prov	SubSc	SubNet	Interface

  - auch als Anycast-Adressen verwendet
  - gekapselte IPv4-Adressen

80	16	32
0000.....0000	0000/FFFF	IPv4

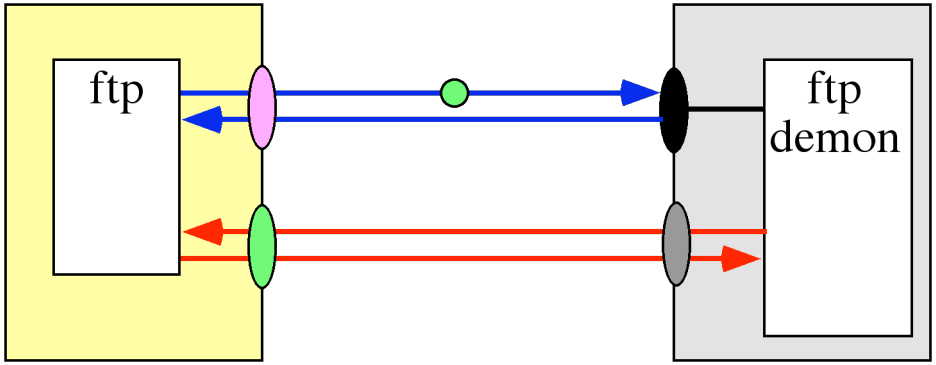
- Portkonzept
  - mehrere Dienste in einem Rechner
  - mehrere Verbindungen zwischen zwei Rechnern
  - Identifikation der Prozesse



- Well-known-ports für wichtige Internet-Dienste

21	ftp control port
23	Telnet
80	httpd
111	Sun Remote Procedure Call

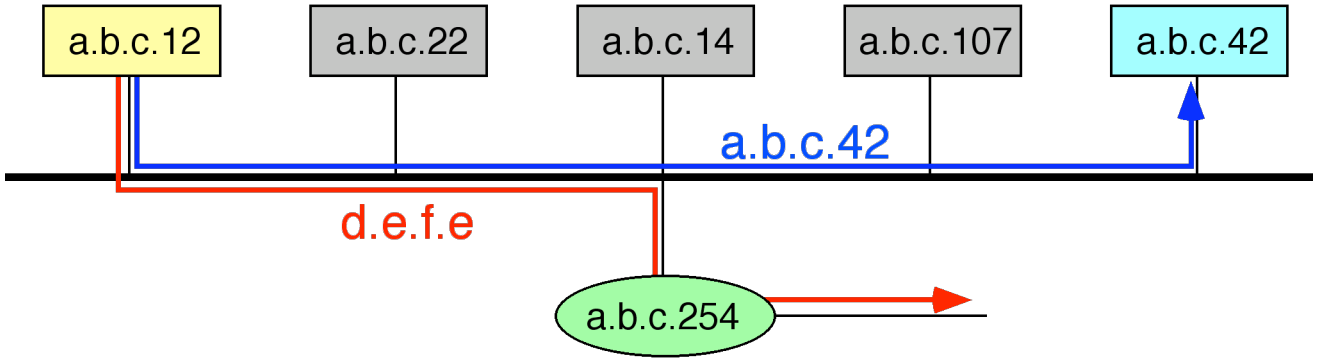
- Mehrfacher Service auf einem Port
  - Kontaktaufnahme über'Anrufport'



  - Datenverbindung auf dynamisch zugewiesenem Port
  - 'passive ftp'
- Wie finden andere Computer den richtigen Port?
  - Anfrage bei zentraler Instanz: port-mapper
  - port-mapper vermittelt Dienst und Anfrage
  - unterhält Verzeichnis (Dienst, Programm)
  - sucht freien Port
  - teilt Quelle und Ziel den Port mit
  - hat feste Portadresse

- Anschlußpunktadressen
  - ohne Bindung von der Hardware verwendet
  - (Dienstadresse, Kontextadresse, Geräteadresse)
  - (Telefonnummer), Pointer, I/O-Portadressen, …
- IEEE 802.2
  - 16 bit oder 48 bit (in LANs)
  - Multicast: 01 00 5E + 24 bit
- z.B. Ethernet
  - 22 bit Organisationskennung (Hersteller)
  - 24 bit Seriennummer
  - im Ethernet-Chip oder ROM
  - Bsp: 08:00:07:1E:5F:8B
- dynamische Festlegung (z.B. LocalTalk)
  - beim Einschalten des Gerätes
  - zufällig Adresse auswählen
  - Paket an diese Adresse schicken
  - falls Bestätigung kommt: Adresse bereits vergeben
  - falls Timeout: Adresse verwenden

- Adress Resolution Protocol ARP (RFC826)
  - welche Ethernet-Adresse gehört zu 134.60.77.7?
  - Senden im lokalen Netzwerk nicht über Gateway
  - Abbildung muß auch ohne Gateway funktionieren



- Entscheidung mit Netzmaske
  - im subnetz/24 255.255.255.0

```
if ((Netzmaske AND own_IP) == (Netzmaske AND dest_IP))
   sendto(ARP(dest_IP,own_IP));
   else sendto(ARP(gateway_IP,ownIP));
```

- ARP-Cache mit Tupeln (IP,802.2)

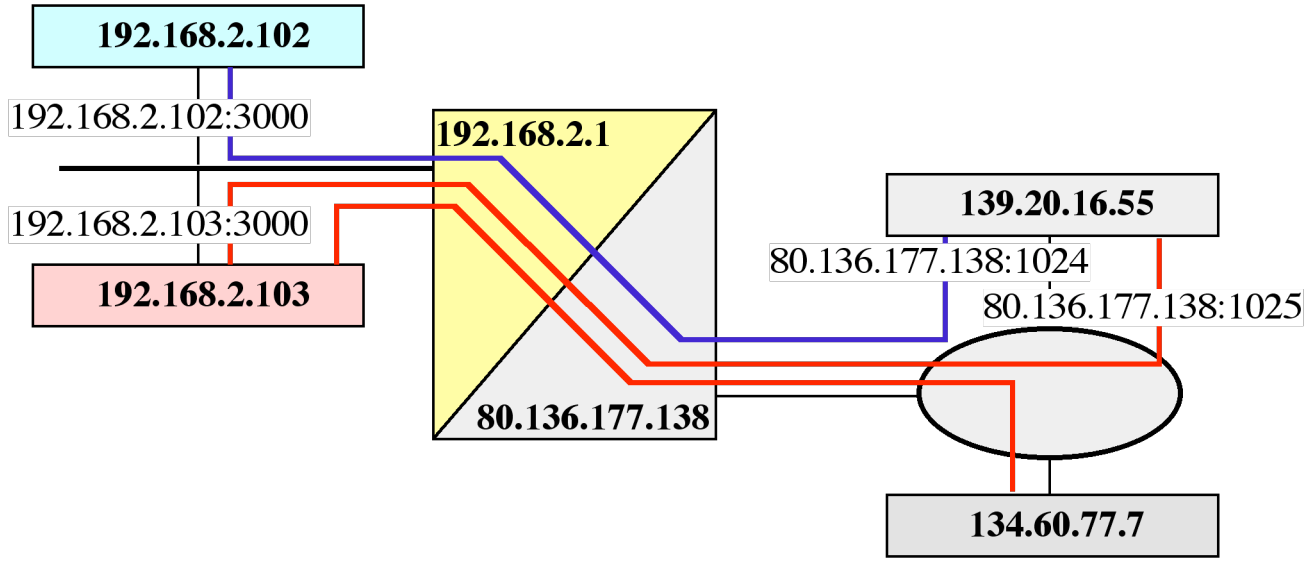
IP-Adresse	802.2-Adresse
134.60.77.99	08:00:08:14:11:2B
134.60.77.77	08:00:07:1E:5F:10
134.60.77.56	08:00:07:1E:5E:AB
134.60.77.10	08:00:07:1E:AF:FE

  - cache wie üblich klein <-> Suchzeit
  - LRU etc.
- Falls IP-Nr nicht im Cache
  - Ethernet Broadcast (FF:FF:FF:FF:FF:FF)
  - ARP-Request(gesuchte IP-Adr, eigene 802.2-Adr, eigene IP-Adr)
  - ARP-Reply(IP-Adr, 802.2-Adr)
  - Eintragen in lokale Tabelle
  - Timeout: Gateway-Adr eintragen

- Effizienz von Adress-Systemen
- Technische Einschränkungen
  - Hierarchische Weglenkung
  - Nummernbereiche identifizieren Vermittlungseinheiten
  - 03731 39 XXXX
  - kann durch komplexere Adressbindung verbessert werden
  - 2010: 769 M Hosts, 2^{32} Adressen => ~18,7%
- Administrative Einschränkungen
  - Zuordnung von Gruppen an Verwaltungseinheiten
  - Wachstum der Bedarfsgruppe ungewiß
  - 139.20.nnn.mmm blockiert 2^{16} Adressen
  - Herstellerid im Ethernet
- H-Ratio [Huitema]
  - Maximum 0,30103
  - Sättigung zwischen 0,14 und 0,26
  - Telefonsysteme: Frankreich: 0,26, USA: 0,24
  - Ethernet: Annahme 10^{10} Adapter - 0,21
  - IPv4 2010: 0,28 bei 769 Millionen Hosts

$$H = \frac{\log(Anzahl\ \ Objekte)}{Adressbits}$$

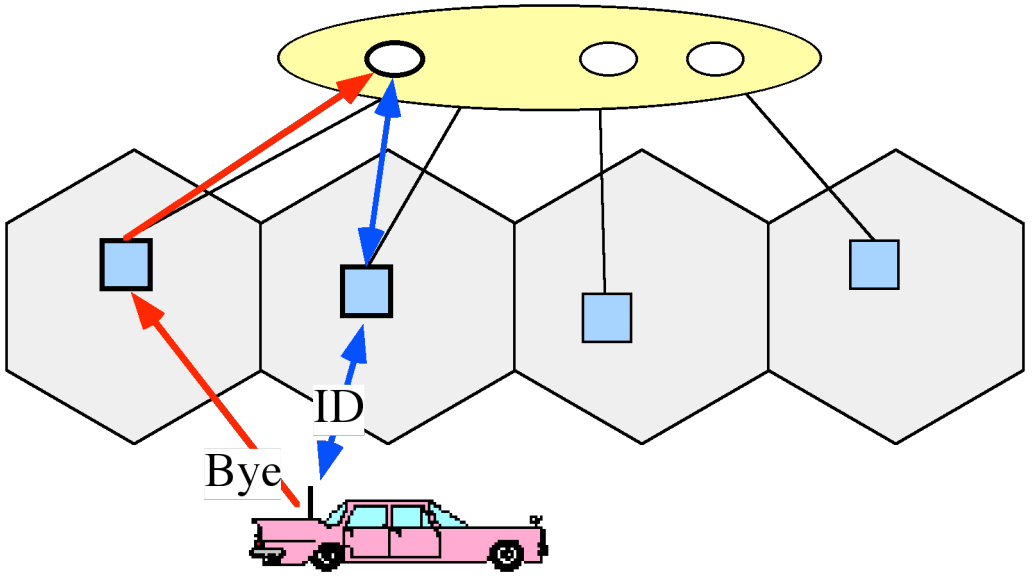
- NAT: Network Address Translation
  - auch IP-Masquerading
  - Gateway hat eine öffentliche IP-Nummer
  - viele nicht-öffentliche IP-Nummern
  - nur lokale Signifikanz
- Multiplex vieler n-IP-Nummern auf einer ö-IP-Nummer
  - Gateway vermittelt



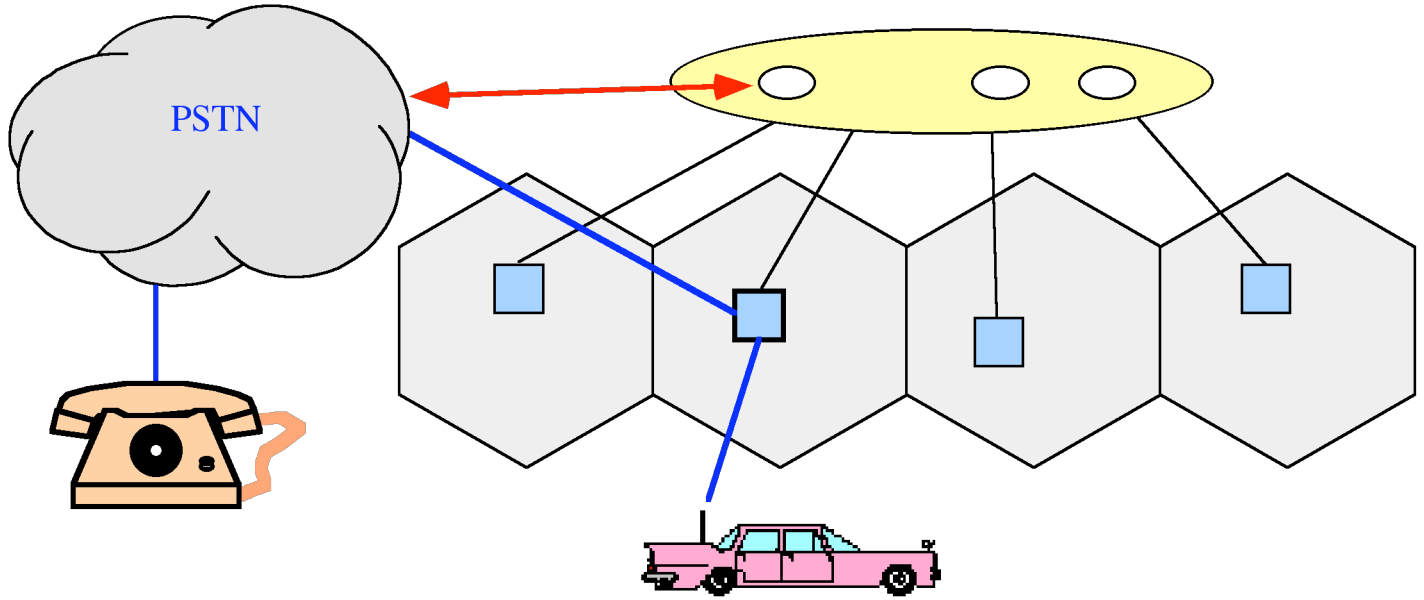
  - Portnummern als Adresserweiterung
  - evtl. Konflikte mit Portnummernvergabe
  - wohlbekannte Ports im privaten Netz?

- Eigenschaften der Adreßbindung
  - Zeitpunkt
  - Algorithmus und Antwortzeit
  - Flexibilität und Fehlertoleranz
  - Wartbarkeit
  - Aktualität und Verfügbarkeit
  - Hochleistungs-Datenbank
- statische Adressbindung
  - fester Zeitpunkt
  - hoher Aufwand für die Aktualisierung der Datenbasis
  - Voraussetzung: hierarchisches System
  - Telefonnummern
  - ungeeignet bei mobilen Teilnehmern
  - DNS: Eintrag in Nameserver
- dynamische Adressbindung
  - beim Verbindungsaufbau
  - fortlaufende Adressbindung
  - suchende Adressbindung

- überwachende Adressbindung
  - System führt Wissen ständig nach
  - Heimat-Server und Netzzugangs-Verzeichnis
  - für mobile Geräte
  - Registrierung
  - Ab- und Anmelden beim Ortswechsel
- Anmeldung
  - beim Netzzugang
  - Auswahl einer zugangslokalen Identität



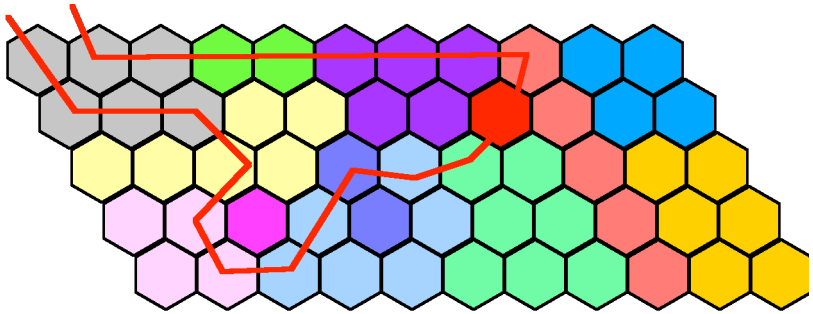
- Datenaustausch zwischen Zugang und ????
  - (Identität, Netzzugang) an Heimat-Verzeichnis
  - Eintrag des neuen Tupels in der Datenbasis
  - Autorisierung etc.



- Verbindungsaufbau
  1. Heimat-Verzeichnis mit statischer Adressbindung finden
  2. Bindungsanfrage an Heimat-Verzeichnis
  3. Ruf an Zugang schicken

- GSM
  - home-location and visitor-location registry
  - Zellwechsel führt zu Registriertransaktion
- ITU: Universal Personal Telephony
  - Smartcards
  - Lesegeräte in allen Telefonen
  - drahtlose Registrierung
- Probleme
  - viele Update-Transaktionen bei hoher Mobilität
  - präzise Bewegungsprofile bei piko-zellulären Netzen

- suchende Adressbindung
  - Aufbau der Orts-Information just-in-time
  - Suchmeldung über Broadcast-Kanal (Funk)
  - Verbindungsannahme => Standortmeldung beim Netz



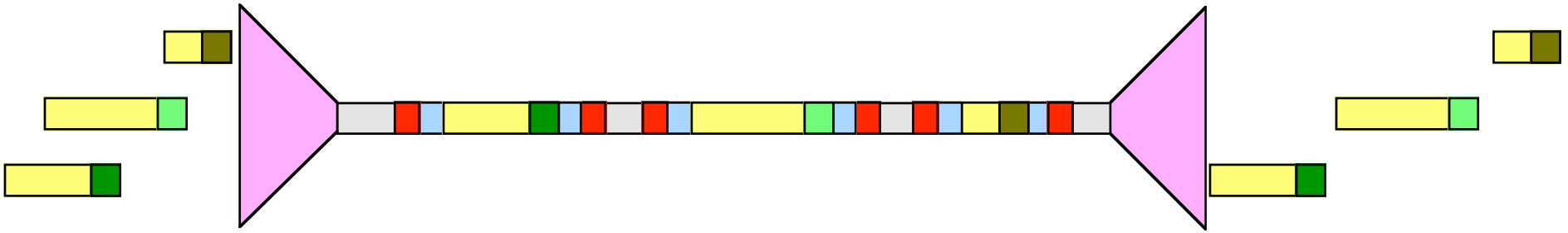
1.7 Standardisierung

- ISO (International Standards Organization)
- IEC International Electrotechnical Commission
- ITU (International Telecommunications Union)
  - ITU-T (CCITT)
  - ITU-R (CCIR)
  - Zusammenschluß der 'Provider'
- CEPT Conference Européenne des Administration des Postes et des Telecommunications
  - BTX-Standard
- ETSI
  - European Telecommunication Standards Insitute
  - GSM, EuroISDN, Hyperlan, Tetra
- IEEE (Institute of Electrical and Electronics Engineers)
- Industrievereinigungen
  - ATM-Forum
  - ECMA (European Computer Manufacturers Association)
- IETF - Internet Engineering Taskforce

2. Asynchrone Rahmenübertragung

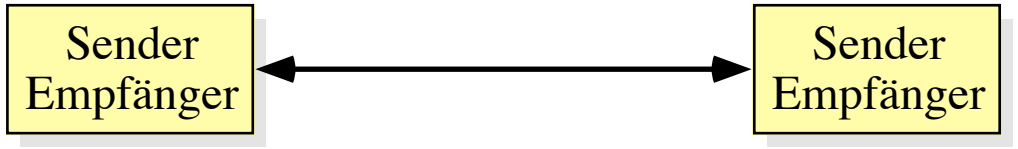
- Synchrone Leitung - asynchroner Datenstrom
  - Rahmenkennzeichen

  - Füllsignale
    - Synchronisation, …
  - beliebig Bitanzahl
  - statistischer Multiplex

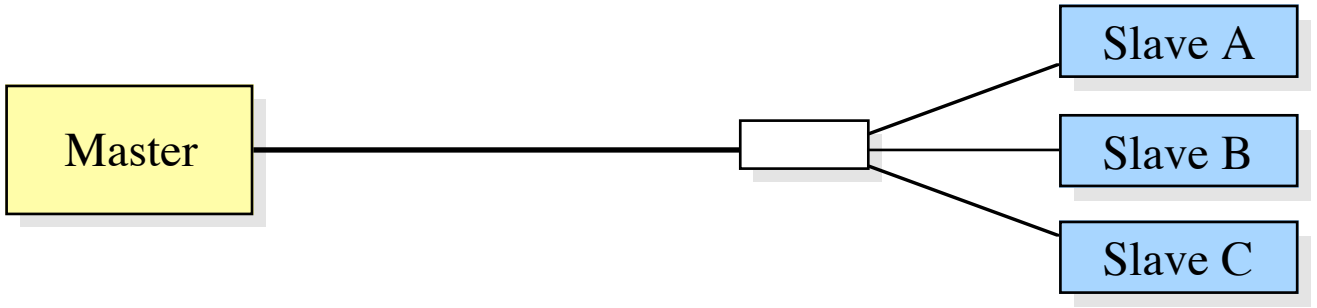


  - Verbindungskontrolle
  - Fehlererkennung
    - Wahrscheinlichkeit unentdeckter Fehler, CRC
  - Fehlerkorrektur: Wiederholung, FEC
- Andere Namen: Zellen, Pakete

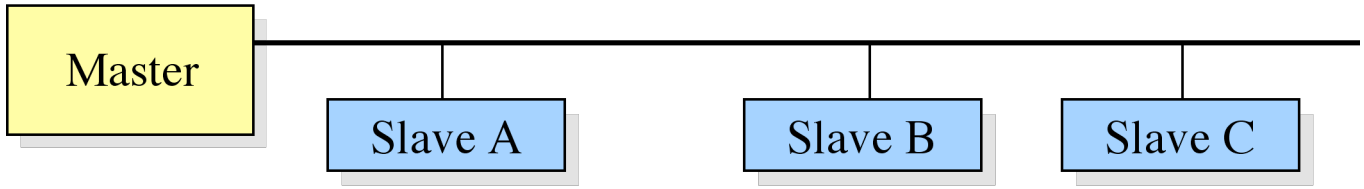
- Topologien
- Punkt-zu-Punkt



- Multipoint



  - Teilstrecke auf einem Übertragungskanal mit Multiplex
  - Adressierung
- Multidrop



  - ein Übertragungskanal, mehrere Zuhörer
  - Adressierung

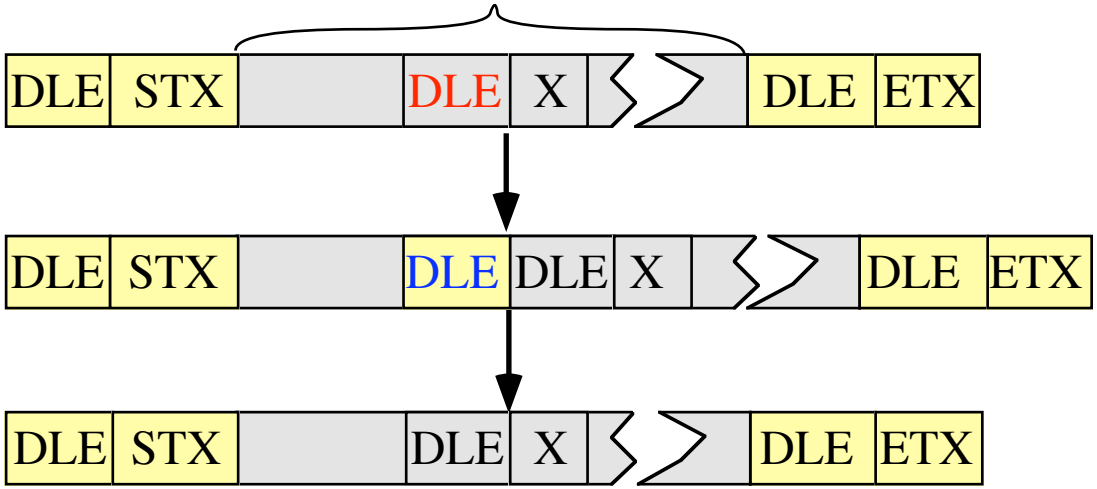
- Exemplarisches Rahmenformat
  - "Link-Layer Protocol Data Unit"
  - bit- oder byteorientiert

- Flag: Oktettsynchronisierung und Rahmenbegrenzung
- Addr: Adressierung für LAN- und Mehrpunktkonfigurationen
- Control Art der Meldung
  - Open, Close
  - Bestätigung, Flusskontrolle
  - Informationstransport ...
- Seq##: Laufende Nummerierung
  - evtl. verlorene Meldungen festzustellen
  - Bestätigungen ebenfalls mit Sequenznummer.
- Daten: Von der Sicherungsschicht nicht weiter interpretiert
- CRC: Prüfsumme für Bitfehler (Cyclic Redundancy Check)

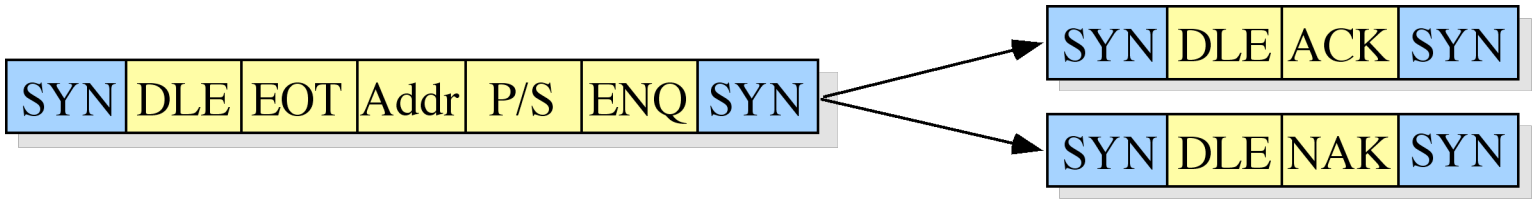
2.1 Zeichenorientierte Rahmenübertragung: BSC (Bisync)

- IBM
  - Terminalansteuerung
  - EBCDIC
  - ISO basic mode mit ASCII
- Spezialbuchstaben für Rahmenbildung und Kontrolle
  - SYN als Lückenfüller
  - SOH: start of header
  - STX: start of text
  - ETX: end of text
  - EOT: end of transmission
  - ACK/NAK: (negative) acknowledge
  - DLE: data link escape
- Zeichenorientierter Rahmen
  - Daten durch (DLE, STX) und (DLE, ETX) eingerahmt
  - Paketanfang (DLE, SOH, Identifier, Stationsadresse)

- Transparenz der Datenübertragung:
  - Daten eine (DLE, . . .) Gruppe als Inhalt => Missverständnis
  - "Zeichenstopfen": ein zusätzliches DLE-Zeichen wird eingefügt
  - vom Empfänger wieder entfernt

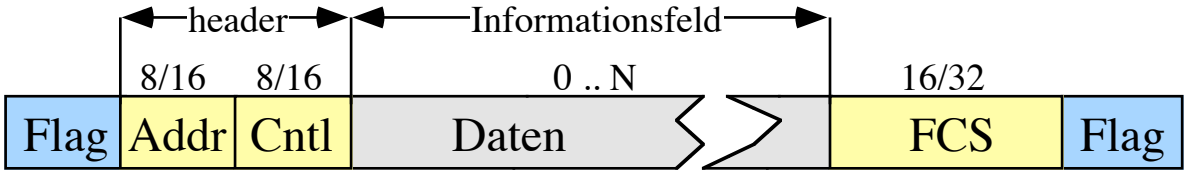


- Poll+Adresse: Master bietet Slave die Möglichkeit, Daten zu senden
- Select + Adresse: Master hat Daten für den Slave



2.2 Bitorientierte Rahmenübertragung: HDLC

- High Level Data Link Control [IBM]
  - LAP-B (ISO 4335)
  - LAP-D (ITU Q.921) Link-Protokoll für ISDN-Signalisierung
  - LLC (Logical Link Control, IEEE 802.2) für LANs



- HDLC-Rahmen
  - beliebige Bitanzahl im Rahmen (Paket)
  - Flags 'rahmen' Daten ein
  - Kontrollinformation am Paketanfang (header)
  - CRC-Prüfsumme am Paketende ($x^{16}+x^{12}+x^5+1$)

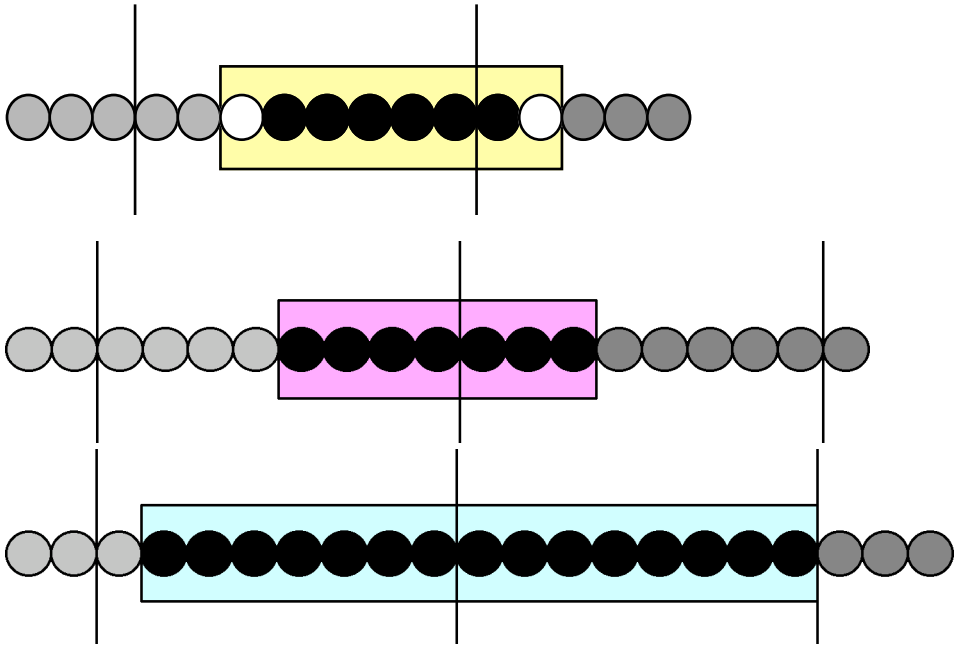
• Besondere Bitmuster

Flags, ' 0111 1110 '

- evtl. auch als Idle-Muster
- Bitstuffing

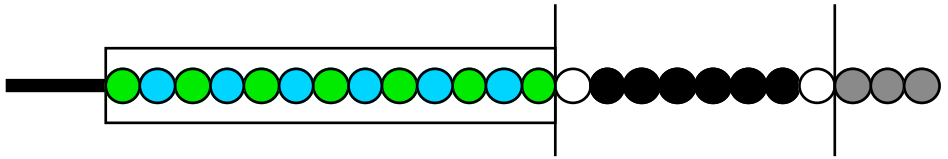
Abbruchmuster (abort), 7 mal '1'

Idle-Muster, 15 mal '1'

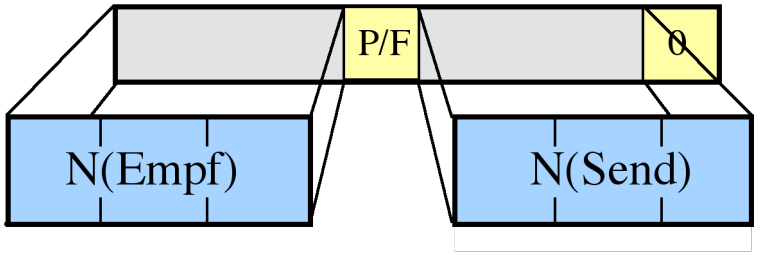


Taktsynchronisation

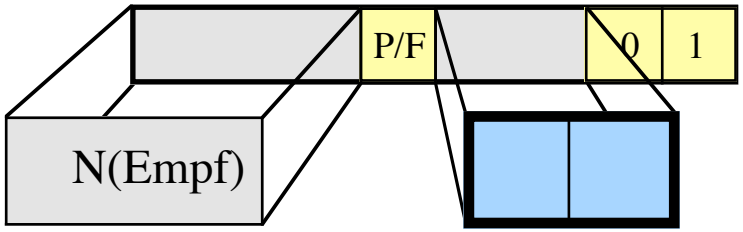
- zusätzliche Präambel
- falls kein Takt vom Modem
- evtl. >100 Bits,
- gehört zur physikalischen Ebene



- I-Rahmen (Information):
  - Nutzdaten im Informationsfeld.
  - 'Huckepack' Bestätigung f. Gegenrichtung.
  - Evtl. 16 Bit anstatt 8 Bit Kontrollfeld.

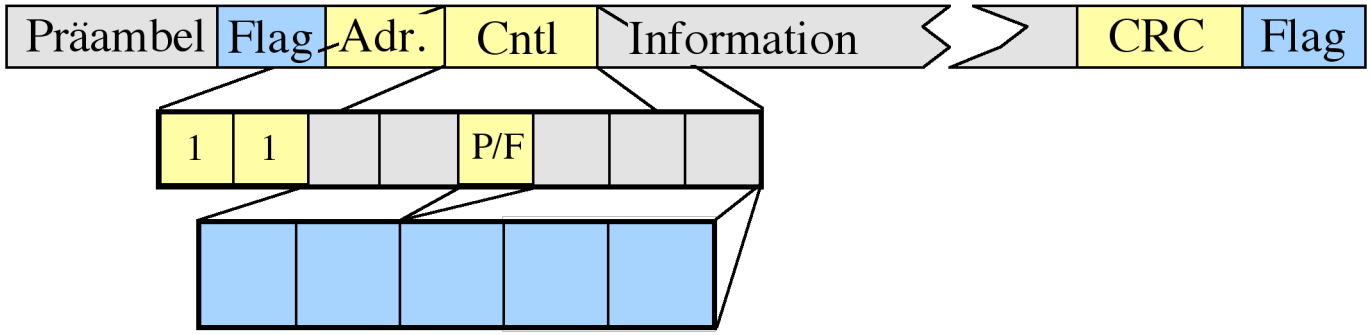


- S-Rahmen (Supervisory):
  - Stop & Go Flusskontrolle
  - Bestätigungen, falls nicht Huckepack
  - Anfordern von Wiederholungen



00	RR	C/R	Receive Ready
10	RNR	C/R	Receive Not Ready
01	REJ	C/R	Reject, letzte N( ) Rahmen
11	SREJ	C/R	Selective Reject (optional)

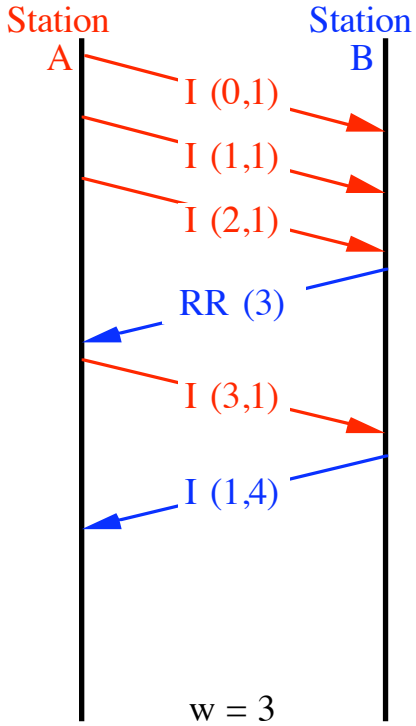
- U-Rahmen (Unnumbered)
  - bitorientiertes Kommandofeld



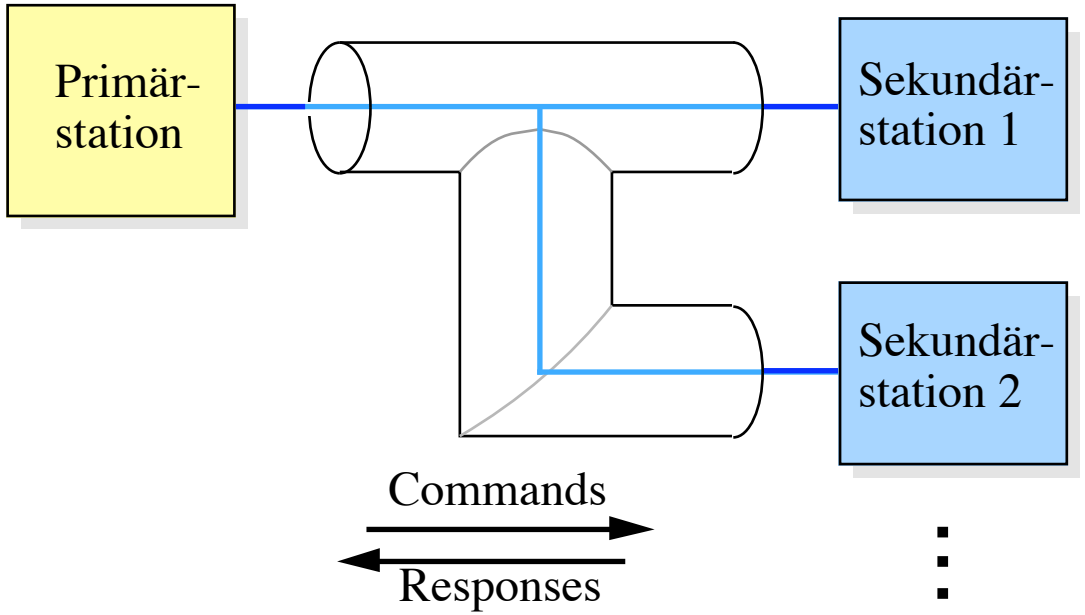
- Kontrollfeldwerte (Pakettypen)

00001	SIM	C/-	Set Initialization Mode
00001	RIM	-/R	Request Initialization Mode
00011	SARM	C/-	Set Asynchr. Resp. Mode
00011	DM	-/R	Disconnected Mode
01000	DISC	C/-	Disconnect
01000	RD	-/R	Request Disconnect
10000	SNRM	C/-	Set Normal Resp. Mode
00000	UI	C/R	Unnumbered Information
00100	UP	C/-	Unnumbered Poll
00111	SABM	C/-	Set Async. Bal. Resp. Mode
01100	UA	-/R	Unnumbered Acknowledge
10001	CMDR	-/R	Command Reject (NRM)
10001	FRMR	-/R	Frame Reject (ABM)
10111	XID	C/R	Exchange Identification

- Sequenznummern
  - Fehlerkontrolle und Flußkontrolle
  - Fenstermechanismus
- Jede Station unterhält zwei Zähler:
  - Z(Send): Sequenznummer des beiliegenden Pakets
  - Z(Empf): Sequenznummer des erwarteten Paketes
  - Bestätigung für Paket[N-1] in Gegenrichtung
  - Zählung Modulo 8 oder Modulo 128
- Regeln
  - bestätigte Rahmen im Sender freigegeben
  - unbestätigte behalten bzw. wiederholt
  - höchstens w Rahmen dürfen unbestätigt sein (w= Window Size)
  - Zurückhalten von Bestätigungen bremst Datenfluß

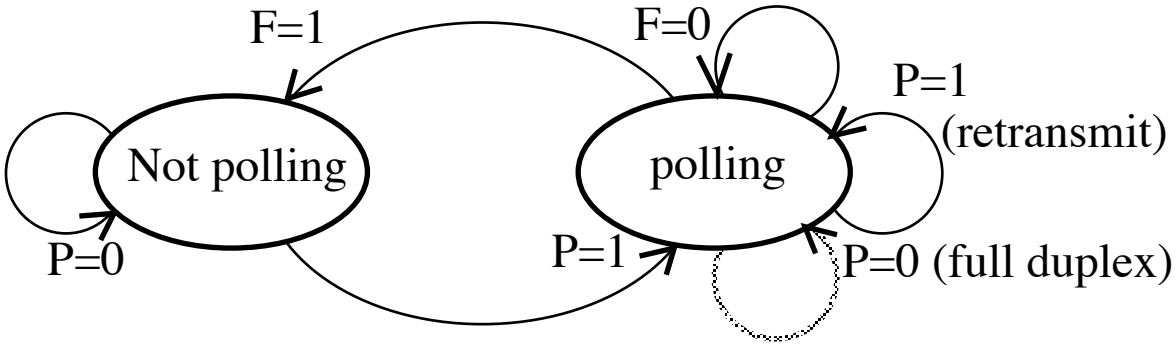


- Normal Response Mode NRM
  - Mehrpunktkonfiguration
  - Zentralrechner bedient mehrere blockorientierte Terminale

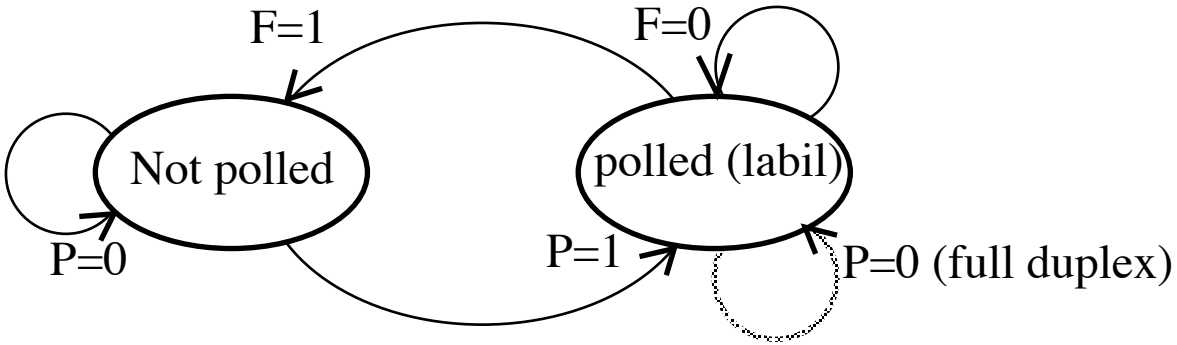


  - Sekundärstationen senden nur nach Aufforderung
  - Poll/Final Bit im Meldungskopf
  - Poll: Sendeerlaubnis von Primärstation
  - Final: Sekundärstation ist fertig

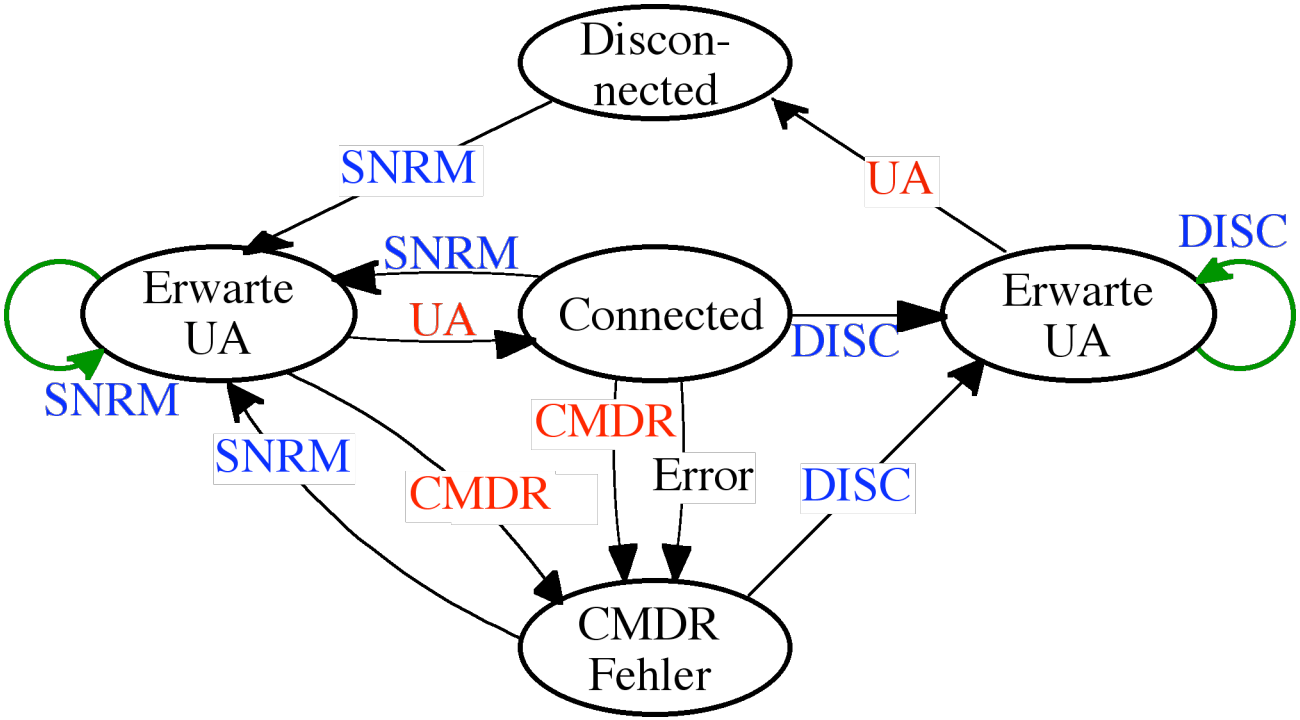
- Zustandsautomat für Poll/Final Bit
- In der Primärstation
  - P senden
  - F empfangen



- In der Sekundärstation
  - P empfangen
  - F senden

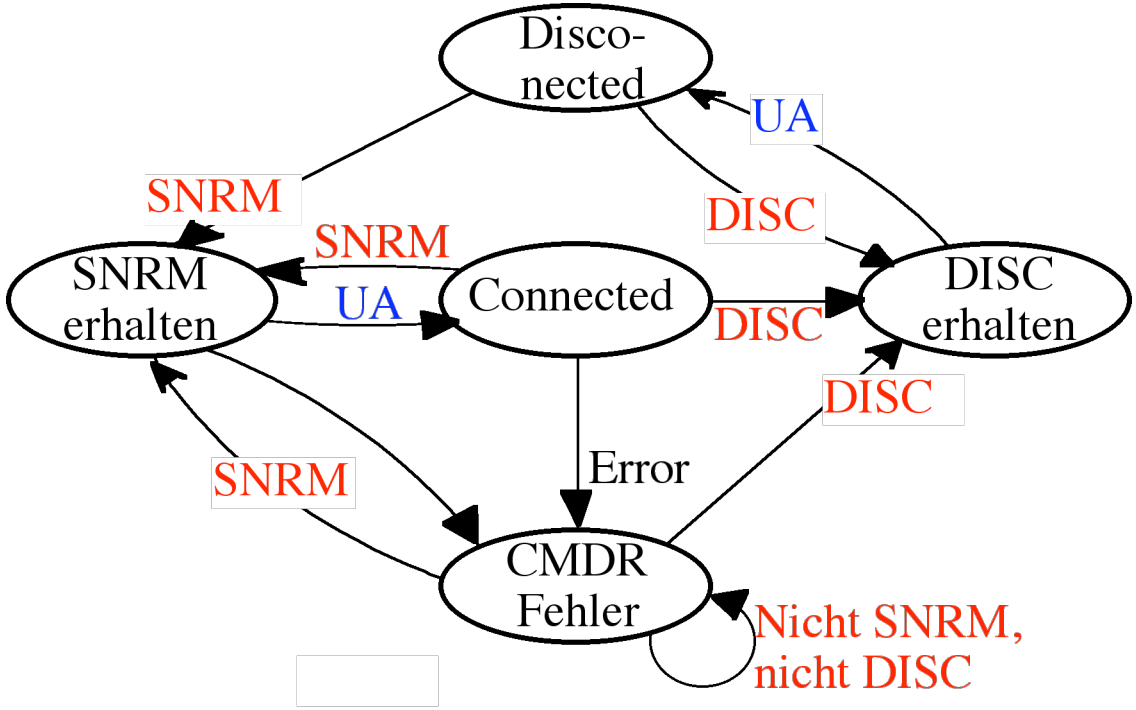


- Verbindungsaufbau und -abbau für Normal Response Mode
  - Zustandsautomat für Primärstation



  - response empfangen
  - command gesendet
  - commands eventuell nach timeout senden

- Zustandsautomat für die Sekundärstation



- command empfangen
- response gesendet
- eventuell nach timeout

• Protokollablauf NRM

B,RR-P(0)	->		Pollen ob es Daten gibt
	<-	B,RIM-F	Guten Morgen, brauche Initialisierung
B,SIM-P	->		Initialisierung bereit
	<-	B,UA-F	OK Set
	...		Download Initialisierung
B,SNRM-P	->		NRM setzen, Reset Ns
	<-	B,UA-F	OK NRM
B,RR-P(0)	->		Nochmal pollen
	<-	B,RR-F(0)	gibt nichts
B,I(0)-P(0)	->		Datenpaket 0 zu B
	<-	B,I(0)- (1)	Datenpaket 0 zu A Bestätigung A0
	<-	B,I(1)-F(1)	Datenpaket 1 nach A
B,I(1)-P(2)	->		Datenpaket 1 + Bestätigung von B0,B1
	<-	B,I(2)-F(2)	Datenpaket 2 an A, Bestätigung für A1
B,RR-P(3)	->		Bestätigung von Datenpaket B2
	<-	B,RR-F(2)	gibt nichts
B,RR-P(3)	->		B pollen
	<-	B,I(3)-F(2)	Übertragung zu A
	*** CRC Error ***		Rahmen verloren
B,RR-P(3)	->		B pollen
	<-	B,I(3)-F(2)	Wiederholung B3
B,RR-P(4)	->		B3 bestätigen und pollen
	<-	B,RR-F(2)	gibt nichts
B,DISC-P	->		Abbau
	<-	B,UA-F	Abbau OK

- Asynchronous Response Mode (ARM)
  - spontane Übertragung von der Sekundärstation
- Initialisierung mit SARM für jede Richtung

B,**SARM**-P	->		ARM, Reset Ns
	<-	B,UA-F	OK ARM
	<-	A,**SARM**-P	ARM, Reset Ns
A,UA-F	->		OK ARM
	...		Duplex DÜ

- Asynchronous Balanced Mode (ABM)
  - gleichberechtigte Übertragung
  - Computer-Computer Konfigurationen
  - Huckepack Bestätigungen möglich
- Initialisierung mit SABM für eine Richtung genügt

B,**SABM**-P	->		ABM, Reset Ns
	<-	B,UA-F	OK ABM
	...		Duplex DÜ

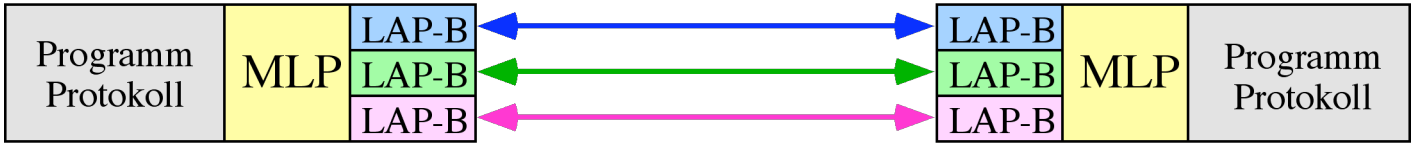
- Adressierung nur mit Adresse der Sekundärstation
  - Primärstation sendet Commands
  - empfängt Replies mit Partneradresse
  - Sekundärstation empfängt Commands
  - sendet Replies mit eigener Adresse
- Commands und Replies anhand der Adresse unterscheiden
  - Senderadresse => Reply
  - Empfängeradresse => Command
- Fehlersituationen
  - Rahmen mit falscher Prüfsumme oder mit Abbruchsequenz verwerfen
  - Überlasteter Empfänger verwirft Rahmen
- Fehlerkorrektur durch erneute Übertragung nach:
  - Time-Out
  - Reject (REJ) nach Empfang des Rahmens N+1 anstatt N
  - Selective Reject SREJ (optional)

- Erneute Initialisierung wird durch Primärstation eingeleitet nach
  - Command Reject (CMDR) im NR Modus
  - Frame Reject (FRMR) im AB Modus
  - internem Fehler
- Optional

DM	Verbindungswunsch ablehnen
RSET	Sequenznummern auf Null
XID	Knotenident. austauschen
RIM	Request Initialization Mode
SIM	Set Initialization Mode

- Erweitertes Kontrollfeld mit SNRME, SARME, SABME.
- Feste Vereinbarung möglich für
  - 16 Bit Adress- & Kontrollfeld
  - maximale Paketlänge
  - Fenstergrösse
  - Initialisierungsbedürfnisse

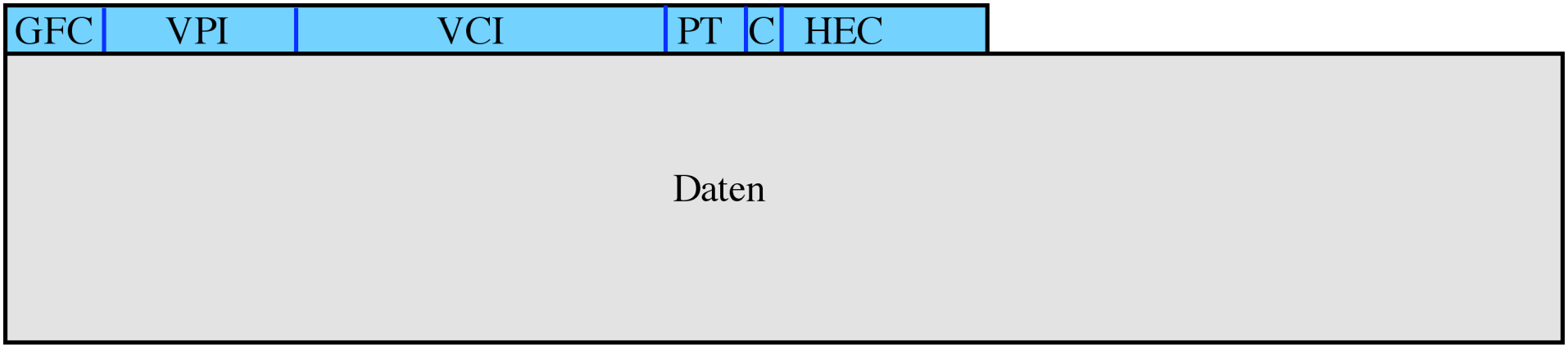
- MLP - Multilink Procedure
  - mehrere physikalische Leitungen für eine logische Strecke
  - Verteilung der Pakete auf mehrere Links
  - Multilink Control (MLC) Feld in jedem Rahmen
  - transparent für Klienten und Single Link Procedure (SLP)



- LAP-M für Modems:
  - X-ON/X-OFF Flußkontrolle mit Unnumbered Information Frame
- LAP-D, Link Schicht der ISDN-Signalisierung
  - Service Access Point Identifiers (SAPI) - Signalisierungsinstanzen
  - Terminal Identifier (TEI)
    - Vergabeprozedur beim Einschalten
    - Adressfeld
  - nur SABME
  - Adressfeld mindestens 16 Bit, Erweiterungsmechanismus
  - verwendet bei Frame Relay

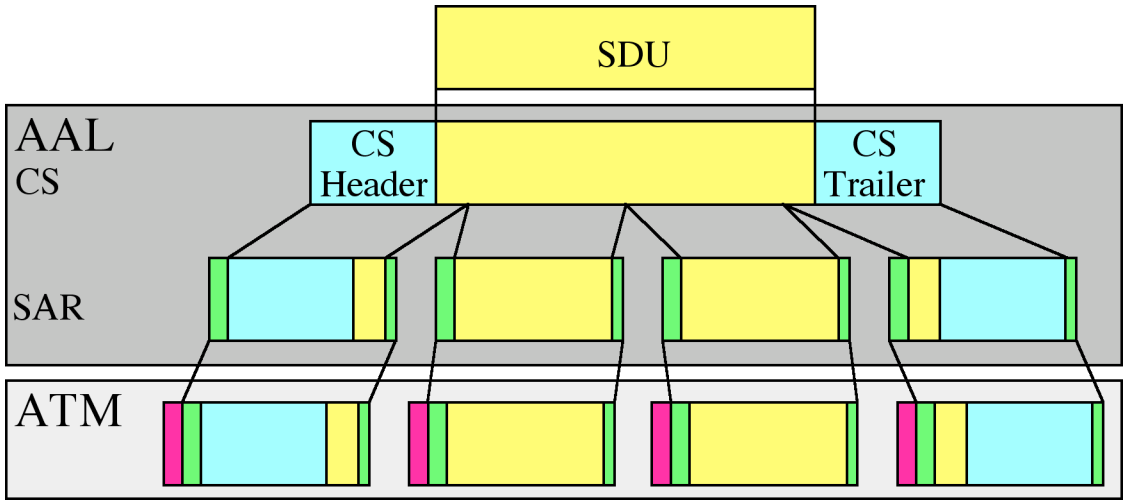
2.3 Übertragung mit fester Rahmengröße: ATM-Zellen

- 53 Bytes = 48 Byte Nutzlast + 5 Byte Verwaltung



- Problem: Aufteilung der Nutzdaten auf Zellen
  - Anwendungsdienst (=> application layer framing)
  - zwischengeschaltete Übertragungsdienste (AAL)
  - Ströme: Ton, Video
  - SDUs: Datenpakete
- AAL: ATM Adaptation Layer
  - Verteilung größerer SDUs auf den Nutzlastbereich der ATM-Zelle
  - evtl. Fehlerkorrektur, Framing, etc.

- Convergence Sublayer: Rahmenbildung

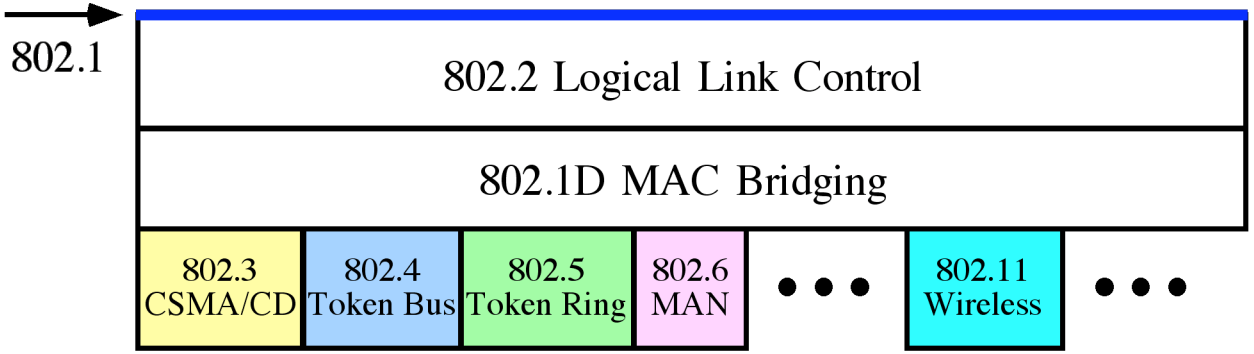


- Segmentation and Reassembly Sublayer
  - bei 140 MBit/s aufwendig

	AAL 1	AAL 2	AAL 3/4	AAL 5
Verkehrstyp	CBR	VBR	ABR/UBR	ABR/UBR
Medien	H.261, G.711	MPEG	Daten	Daten
Nutzlast	46 oder 47	45	44	48
Header in Zelle	Seq#	Seq#		-
Fehler	CRC, FEC?	CRC, FEC?	CRC, Segmentinfo	
CS-Header			Puffergröße	Mngt, CRC

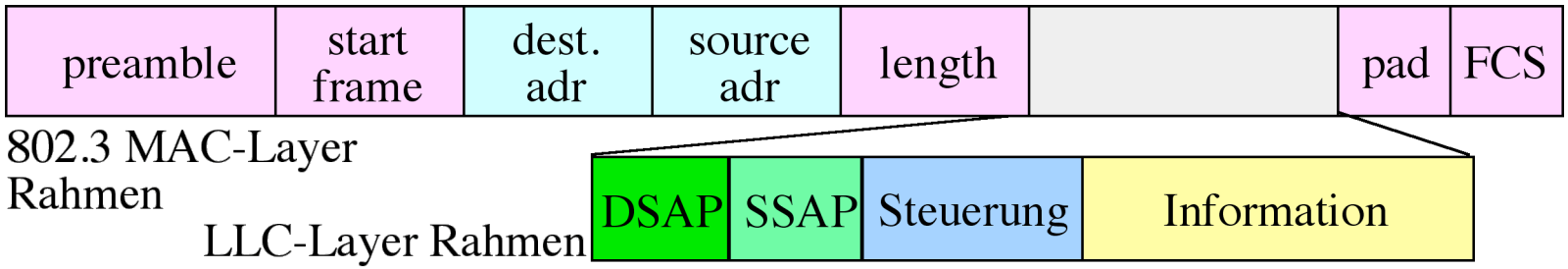
2.4 Ethernet

- Broadcast-Topologie
  - einer sendet, alle hören zu
  - Adressauswertung im *Empfänger* entscheidet über Speicherung
- IEEE Standards 802.x
  - MAC: Medium Access control



- 802.1 Interface
  - "verbindungslos", unbestätigt
    - L_DATA.request, L_DATA.indication (auch DL_UNITDATA)
  - "verbindungslos", bestätigt
    - L_DATA_ACK.[*], L_DATA_ACK_STATUS.indication
  - "verbindungsorientiert"
    - L_CONNECT, L_DATA_CONNECT, L_DISCONNECT

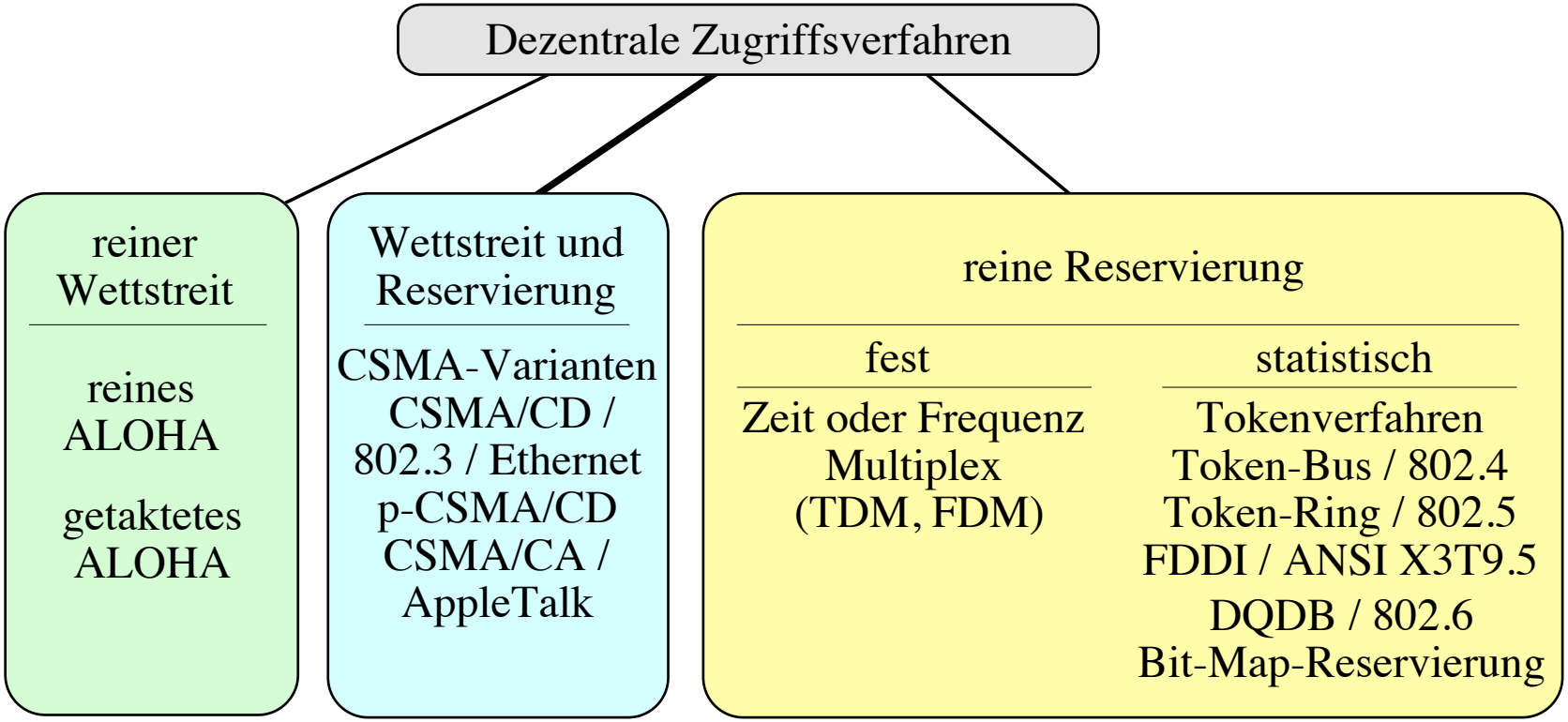
- LLC- und MAC-Rahmen



- Steuerfeld ähnlich wie bei HDLC, LAPB, LAPD …
  - Quell- und Zieladresse
  - 1 oder 2 Byte Steuerfeld
- Verbindungsorientierte Protokoll-Varianten mit 7 Bit Sequenznummern
- Rahmentypen ähnlich HDLC
  - nur symmetrische Konfigurationen
  - I, RR, RNR, REJ, UI, UA, DISC
  - SABMF, XID, TEST, DM, FRMR
  - UI-Frame für typischen Datagramm-Transfer
- MAC-Services
  - MA_UNITDATA.[requestlindication]
  - MA_UNITDATA.confirm bestätigt Übertragung
  - in machen LANs auch Empfang

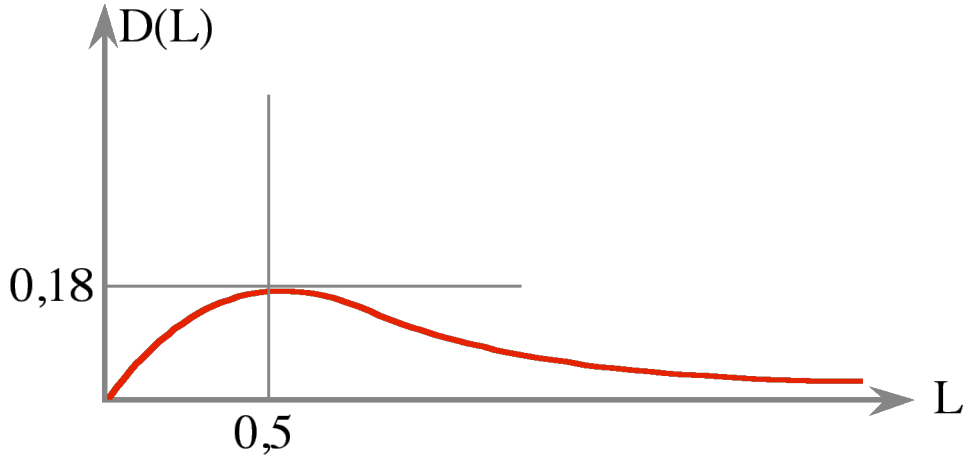
2.4.1 Zugriffssequentialisierung

- Reservierungsverfahren
  - Anmeldung im Reservierungsrahmen
  - Stationen senden in aufsteigender Folge
  - Verhältnis Reservierung/Datenübertragung ungünstig bei vielen Teilnehmern und niedriger Aktivität.
- konzeptuell dezentrale Verfahren

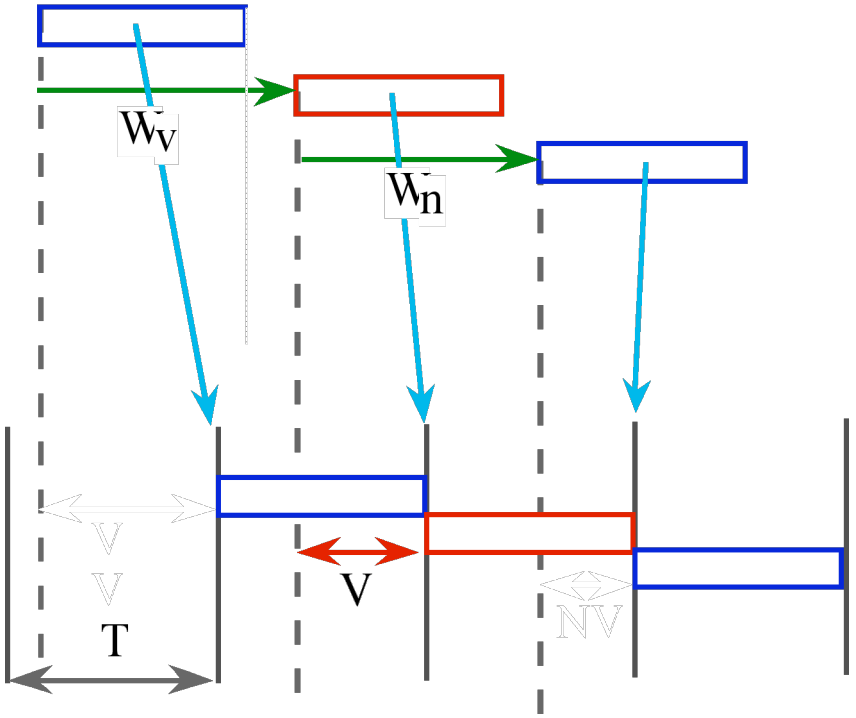


Statistische Verfahren

- Aloha
  - Paket sofort senden, wenn es bereit ist
  - Überlagerung/Kollision wird nicht bemerkt
  - höhere Schichten warten auf Bestätigung
  - falls Timeout erneute Übertragung
- Unter Annahme der Poissonverteilung D< 18%

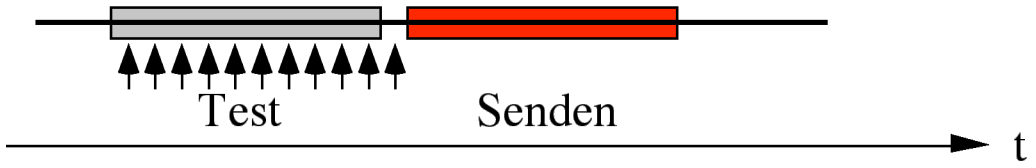


- slotted Aloha
  - Zeitschlitze (Slots) der Dauer ~T
  - Paketlänge fest
  - Übertragung nur zu Beginn eines Slots

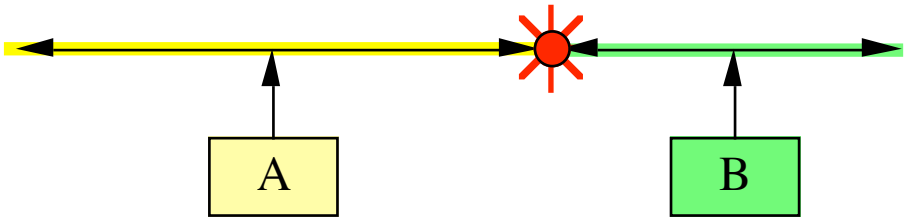


  - entweder komplette Überlagerung oder keine
- Annahme Exponentialverteilung: D < 37%

- CSMA: Carrier Sense Multiple Access
- Vor Übertragung hören, ob Medium belegt
  - wenn frei: Senden
  - wenn belegt: warten und nochmal versuchen



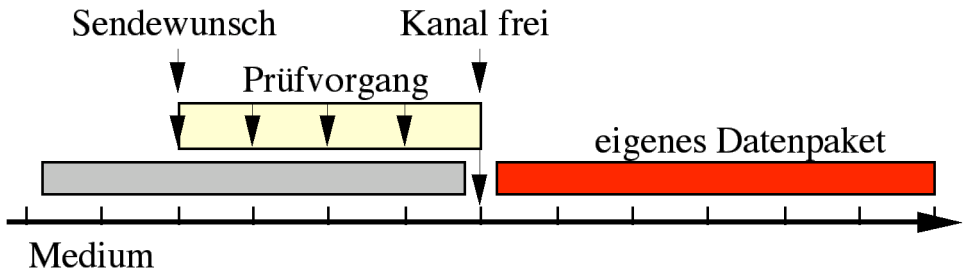
- Problem falls zwei (fast) gleichzeitig testen und dann senden
  - Signallaufzeit bis zu allen anderen ist Risikoperiode



=> p(Kollision) für CSMA ist proportional zur Signallaufzeit
- langes Kabel erhöht Kollisionsgefahr

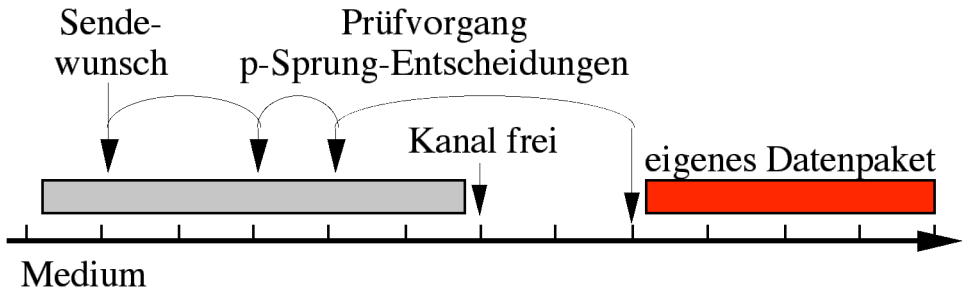
- Persistent CSMA
  - sofort senden, wenn der Kanal frei ist

```
while carrier
      do  { persist };
Send;
```



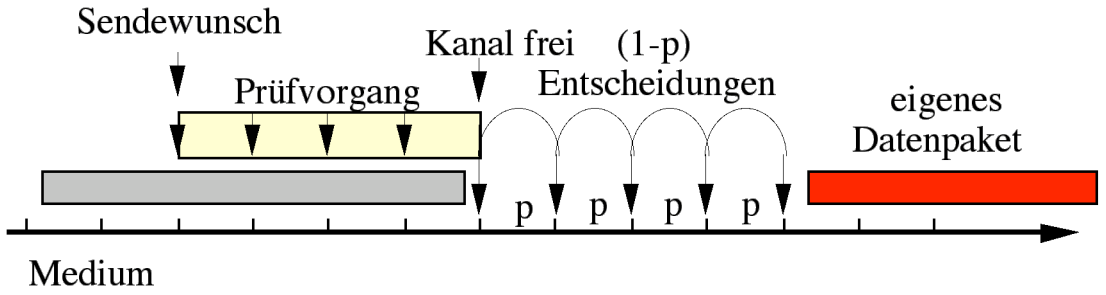
- Non-persistent CSMA:

```
while carrier
      do Wait(random);
Send;
```

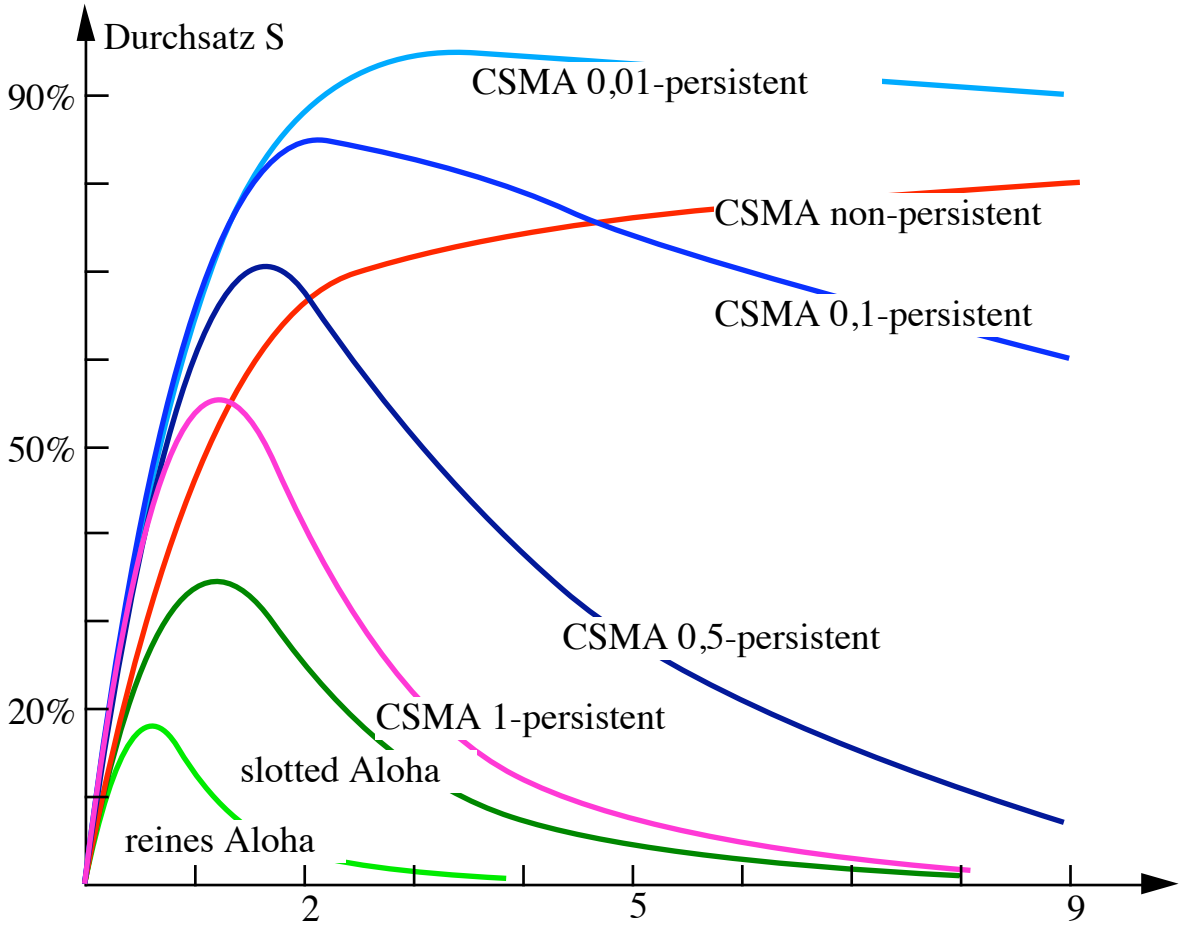


- p-persistent CSMA
  - Medium wird dauernd beobachtet
  - nach dem Freiwerden des Mediums Zufallsentscheidung
  - nur mit Wahrscheinlichkeit p wird sofort gesendet
  - sonst wieder von vorn

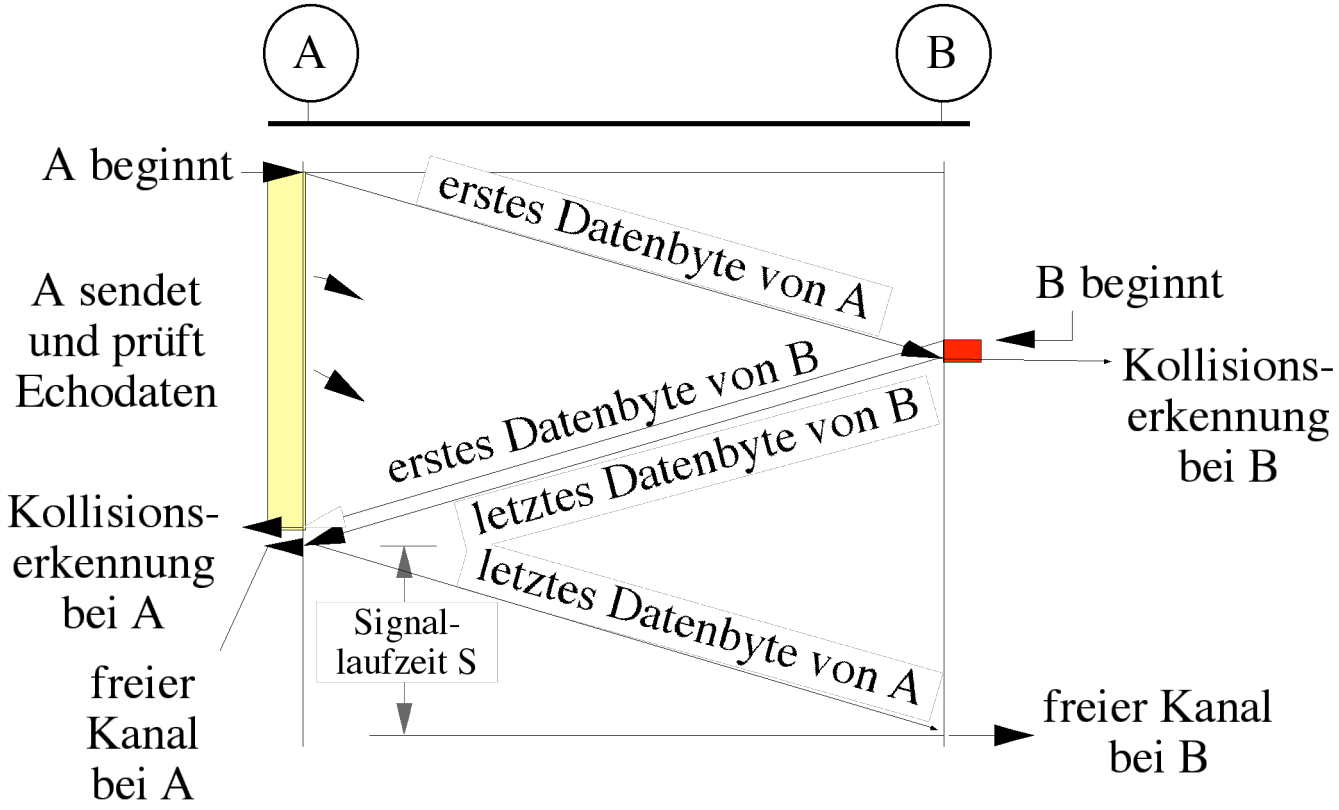
```
while (not p or carrier)
      do Wait(1);
send;
```



- Poisson-verteilte Last



- CSMA/CD: Carrier Sense Multiple Access with Collision Detection
  - Kollision frühzeitig erkennen.
  - Transceiver beobachtet Leitung
  - Beobachten des Mediums während der Übertragung
  - Kollision: Senden abbrechen und Störimpuls senden (Jamming)



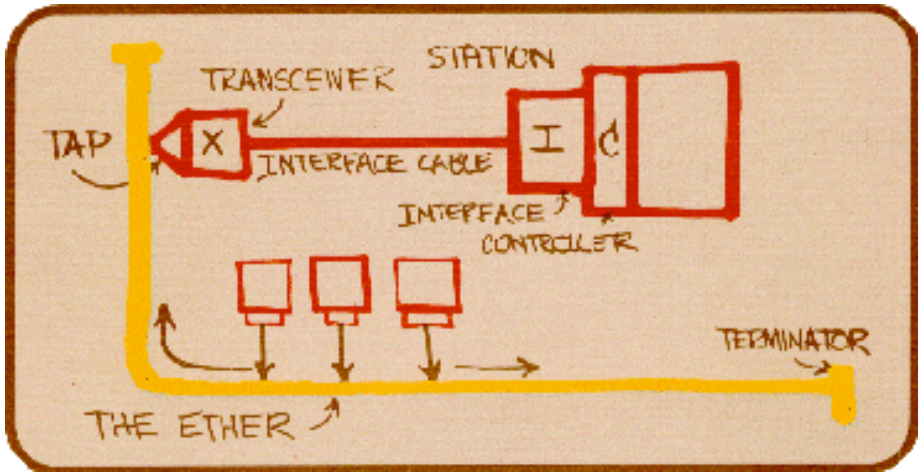
- Kollisionsfenster: slot-time = 2 * Laufzeit + Sicherheitszeit

- Exponential Backoff
  - vor Wiederholung nach einer Kollision wird die Wartezeit verdoppelt
  - Begrenzung auf 2^{10} Slots
  - Anpassung an die Netzlast
  - trotzdem oft lange Wartezeiten
- CSMA/CA: Carrier Sense Multiple Access with Collision Avoidance
  - Probepaket an Empfänger senden (RTS)
  - Auf Antwort warten (CTS)
  - kommt CTS ungestört: Informationspaket senden
  - sichert auch, daß Empfänger bereit
  - RTS/CTS + Wartezeiten verhindert Kollision der Datenpakete
  - RTS kann mit anderem RTS kollidieren

  -> Sender muß nicht mithören, weniger Hardwareaufwand
  - siehe LocalTalk

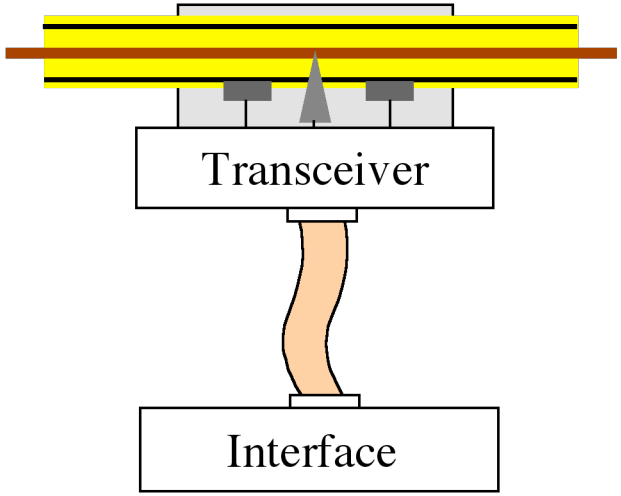
2.4.3 Ethernet

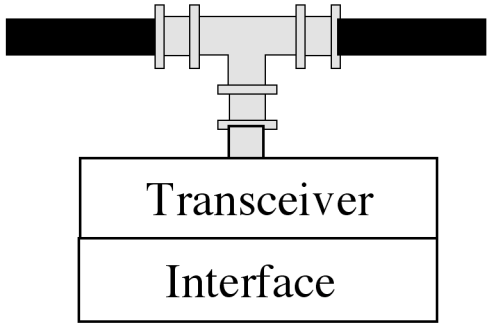
- Bob Metcalfe, Xerox PARC



- CSMA/CD, Kollisionserkennung
  - zwei Pakete auf dem Medium = Signalüberlagerung
  => Manchester - Code - Verletzung
  - |Gleichspannungsanteil| > U_{th} => Kollision
  - DC ohne Kollision > -1.293 V => -1,448V ≤U_{th} ≤ -1,59V
- Größenbeschränkung des Kollisionsgebietes
  - 2500 m => Signallaufzeit maximal 23 μsec
  - nach round-trip von 46,4 μsec (= 464 bit) keine Kollision mehr
  - Paketlänge max. 12.144 bit

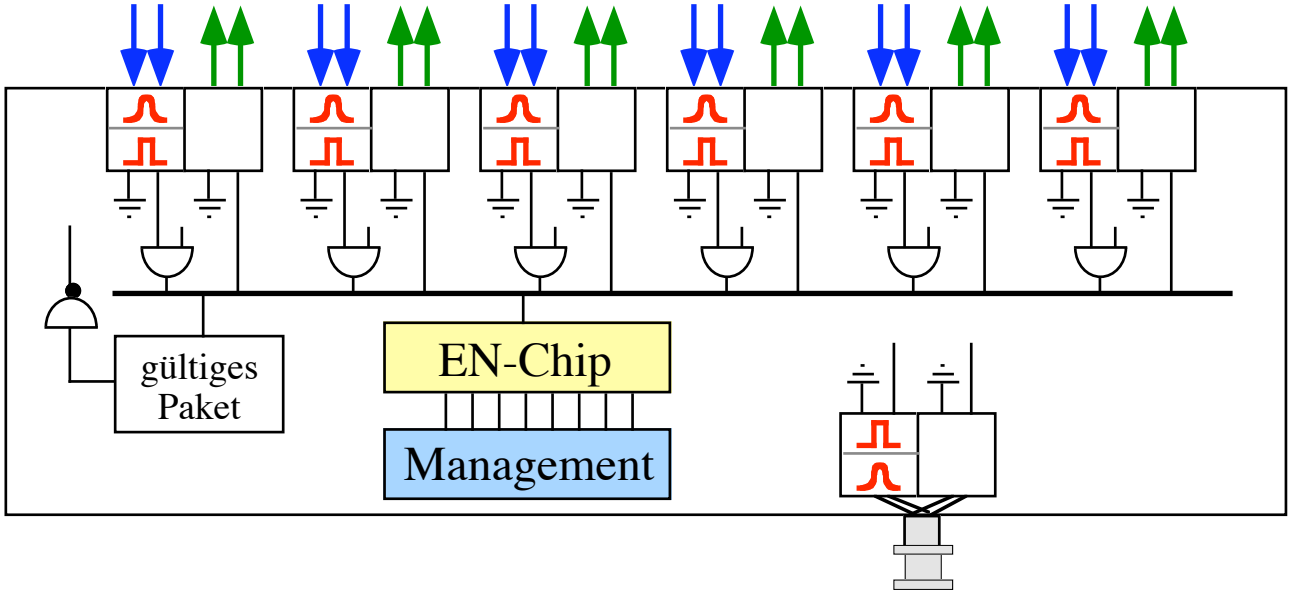
- AUI
  - Interface am Computer mit AUI-Stecker
  - Transceiver am Kabel mit Tap
- 10Base5 (ThickNet, Yellow Cable)
  - 50 Ω Koaxialkabel, 10 mm, starr, 500 m





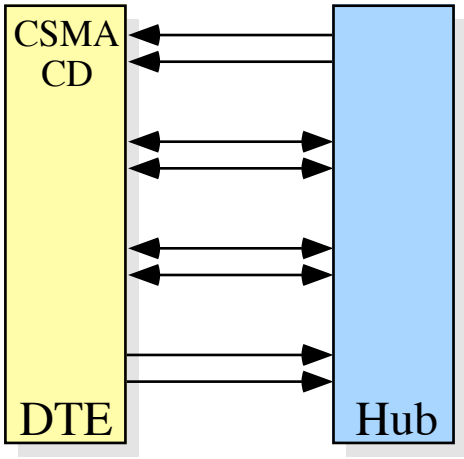
- 10Base2 (ThinNet, Cheapernet)
  - 50 Ω Koaxialkabel, 5 mm, RG58A/U, BNC-Stecker
  - 200 m
- 10Broad36 [Sytec]
  - baumstrukturiertes Kabelverteilnetz, Koaxialkabel 75 Ω
- 10BaseT

- 4 Drähte twisted pair: 2 senden, 2 empfangen
- Verkabelung wie Telefonsystem
- Sternkoppler im 'wiring cabinet'

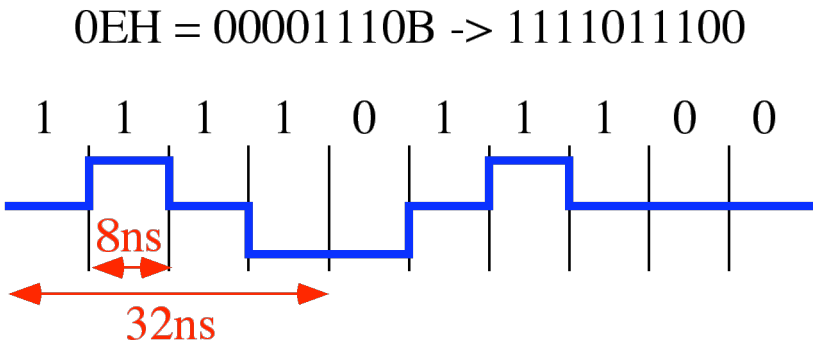


- Gültiges Paket => andere Rx-Leitungen abschalten
- Kollisionen = Carrier-Sense auf dem Empfangspaar AND Senden

- Fast Ethernet (802.3μ)
  - 100 Mbit/s und GigaBit
- CSMA/CD funktioniert bei dieser Geschwindigkeit nicht mehr dezentral
  - Sterntopologie
  - zentrales CSMA/CD
  - Sternkoppler bietet Möglichkeit der Vermittlung
- 100VGAnyLAN (IEEE 802.12)
  - Demand Priority statt CSMA/CD
  - Baumstruktur
- 100BaseT4
  - 4 Adernpaare bis zu 100 Meter, cat 3, 4, 5
  - 8B6T, Symbolrate 25 MHz

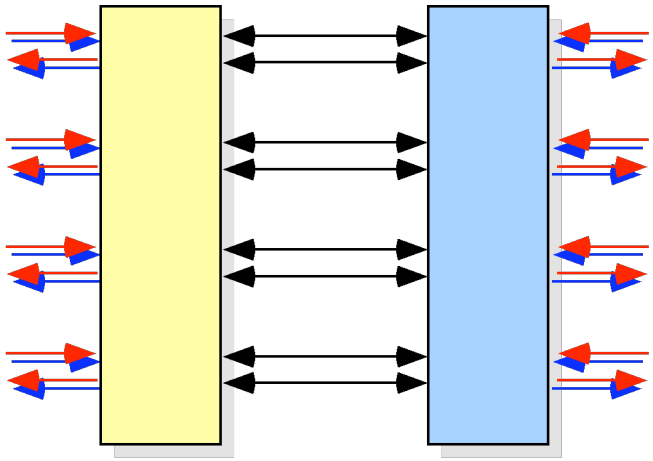


- 100BaseTX
  - 2 Adernpaare, cat 5
  - 4B5B
  - manchesterkodiert 125 MHz: nicht bei cat 5
- Leitungskodierung MLT-3 (NRZI-3)
  - reduziert Frequenz
  - reduziert Gleichstromanteil
  - 11111 -> 32 ns = 31,25 MHz



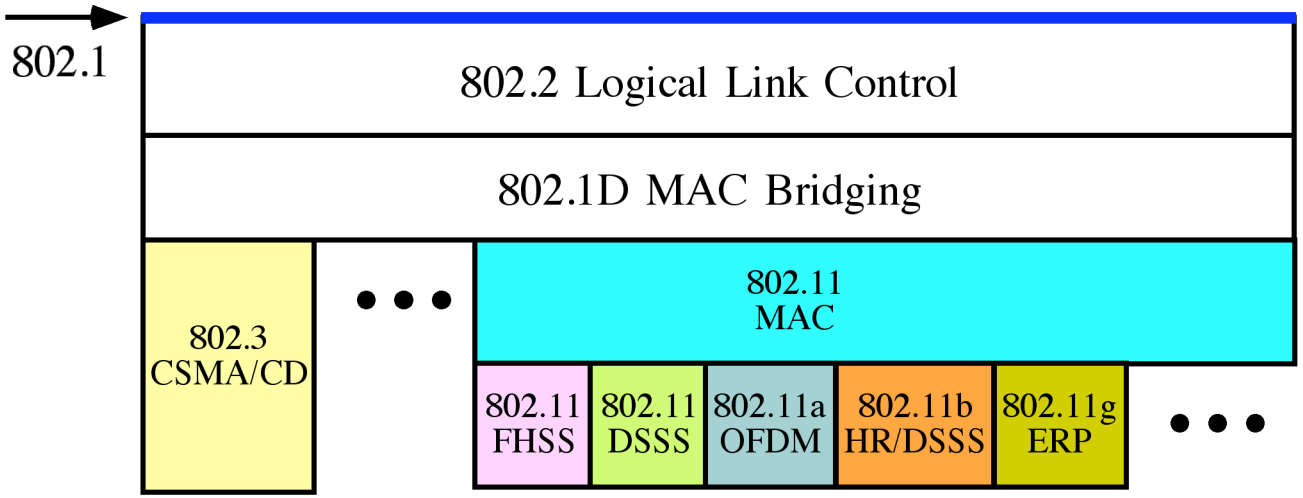
- 4B5B
  - 32 mögliche Kombinationen
  - 16 für Nutzdaten mit mind. 2 Levelwechseln pro Symbol -> Taktableitung
  - Start_of_Frame, End_of _Frame, Idle

- Gigabit-Ethernet (802.3z)
  - T: UTP, cat 5, 100m
  - Gradienten-Multimode Glasfaser 500m
  - SX: (850 nm, max. 5km); LX: (1300nm, max. 500 m)
  - CX: 25 m, "TwinAX"
- 1000Base-X
  - kleine Pakete vs. Collision-Domain (64 bytes ~ 20 m)
  - minimale Frame-Größe 512 bytes
  - Padding, Extension, Frame Bursting (folgende kleine Frames sofort senden)
- 1000Base-T
  - 4 Paare vollduplex, Echounterdrückung
  - 8 Bit vom Interface = 4*2 Bit
  - 125 MSymbole/s pro Paar (125 Mbaud)
- 4D-PAM5 (8B/10B)
  - 2 Bit –> 5 Werte
  - 256 Werte in 625 Symbolwerten kodieren
  - 512 Werte für Trellis-Kodierung, 113 für Kontrolle
  - Viterbi-Decoder



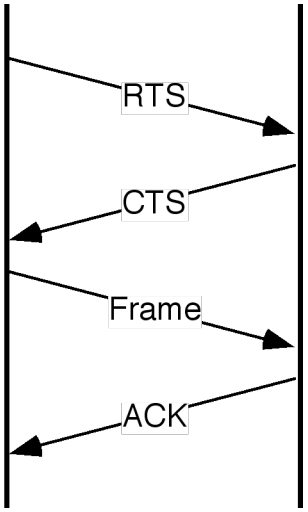
2.7.4 Wireless LANs - 802.11

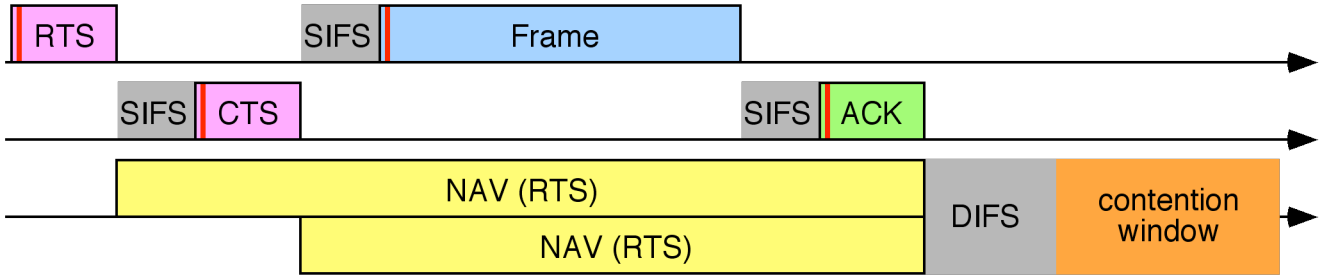
- Integration von lokalen Funknetzwerken in IEEE 802
  - neuer MAC-Layer 802.11
  - drahtlose PHY-Layer



- ISM-Band (industrial, scientific, and medical): 2.4 GHz
- Modulationsverfahren
  - FHSS: frequency hopping spread spectrum
  - DSSS: direct sequence spread spectrum
  - OFDM: orthogonal frequency divison multiplexing (5 GHz)
  - HR/DSSS: higher rate direct sequence spread spectrum
  - ERP: Extended Rate Physical (ERP-DSSS und ERP-OFDM)

- Netzwerkstruktur
  - Access Points
  - zellulärer Charakter
- MAC-Layer 802.11
  - Kanal unzuverlässig: Frames bestätigt übertragen
  - atomic operation: Frame+ACK
  - DCF - Distributed Coordination Function: CSMA/CA
  - Point CF, Hybrid CF
- Virtual Carrier Sense: Network Allocation Vector
  - physical CS teuer und hidden node Problem
  - NAV: Feld *duration* im Frame [μsec]
  - Stationen beobachten duration: NAV-counter

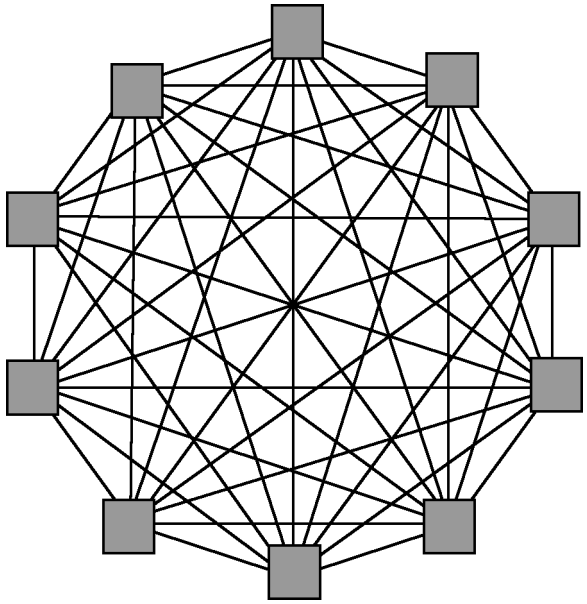




  - short interframe space, DCF interframe space

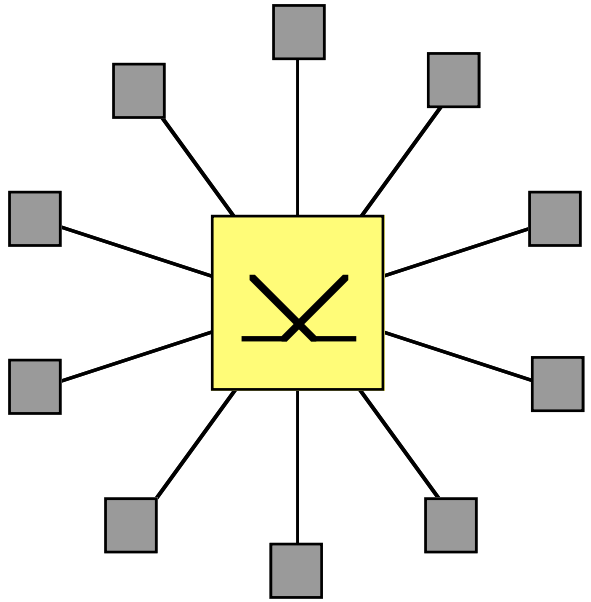
3. Vermittlungsdienste, insbesondere Paketvermittlung

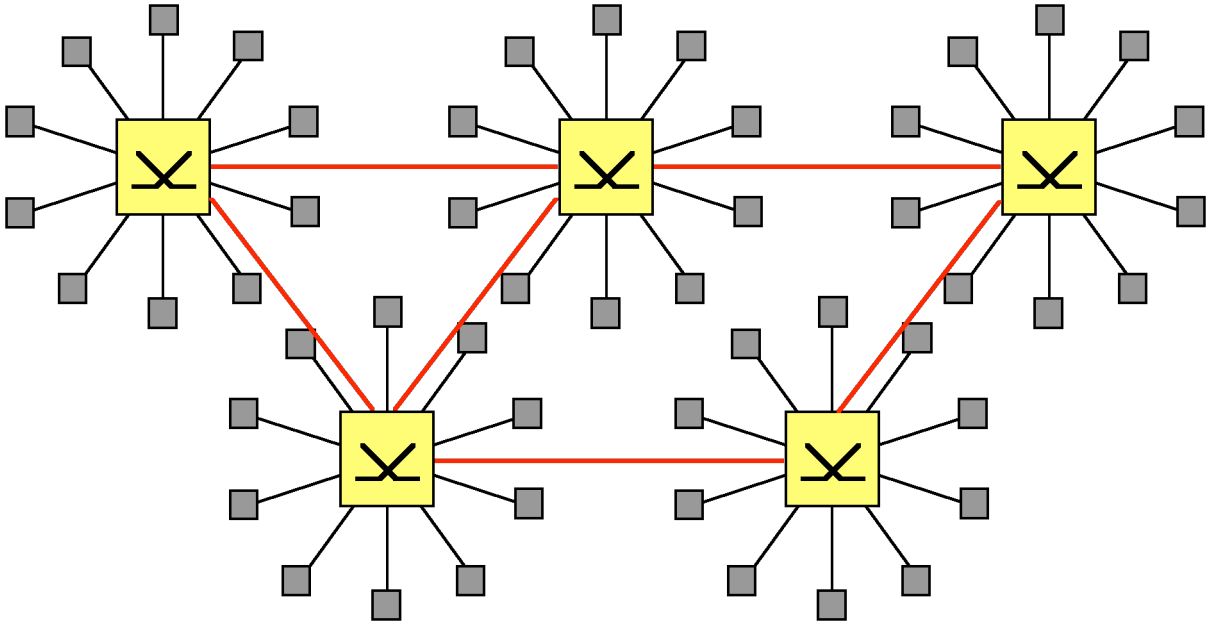
- Welches Problem wird gelöst?
  - N Teilnehmer
  - paarweise temporäre Kommunikationsbeziehungen
  - 'Verbindungen' = Kontext der Kommunikation



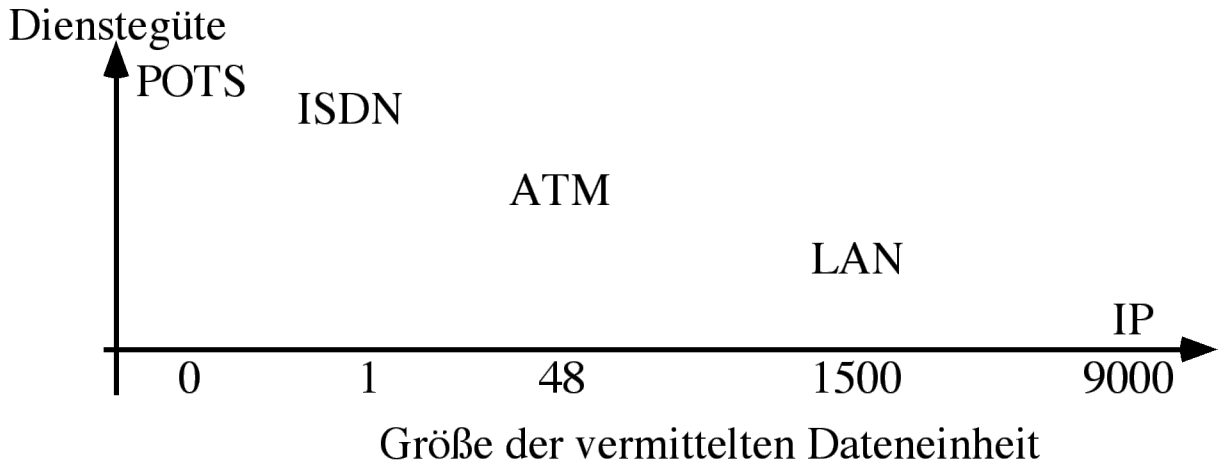
  - voll verbundenes Netz: N*(N-1) Leitungen
- Historische Lösungen
  - Kurier, Post, Versammlung, Kino,
  - Telefon, TV

- Lösungsidee: Sternstruktur
  - N Leitungen
  - 1 Vermittlungsknoten
  - evtl. Multiplex
- Skalierung: Vermaschtes Netz von Sternen
  - M Vermittlungen, M<<N
  - Leitungen zwischen Vermittlungen ≤ M*(M-1)
  - N << N*(n-1) Anschlußleitungen

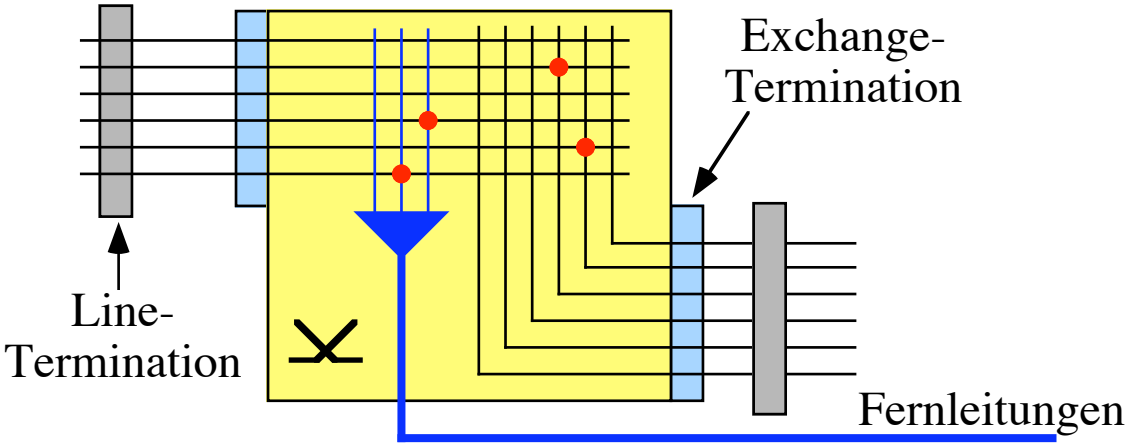




- Was wird transportiert?
  - Medienströme (Telefon, TV, …)
  - Medienpakete (Brief, …)
- Stromorientierte Vermittlung
  - Leitungsvermittlung
  - konzeptuell synchron
  - Dienstegüte eingebaut
- Paketvermittlung
  - Store-and-Forward
  - asynchron, schwer vorhersehbar
  - inhärent 'best-effort'



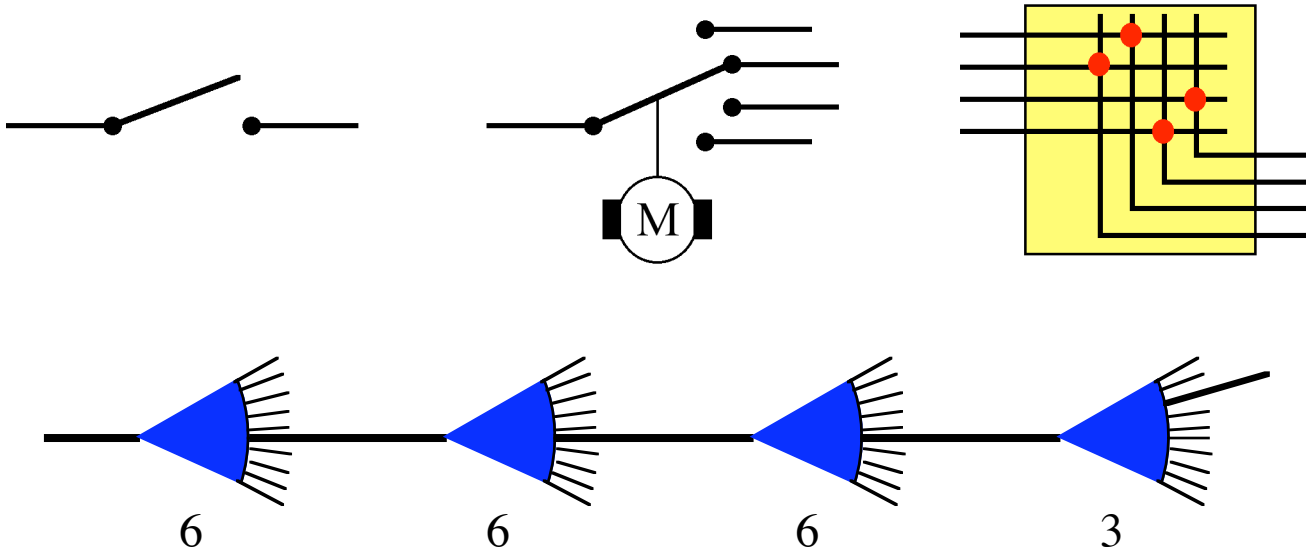
- Aufbau
  - Teilnehmeranschlußleitungen
  - LT, ET
  - Koppelfeld
  - Fernleitung



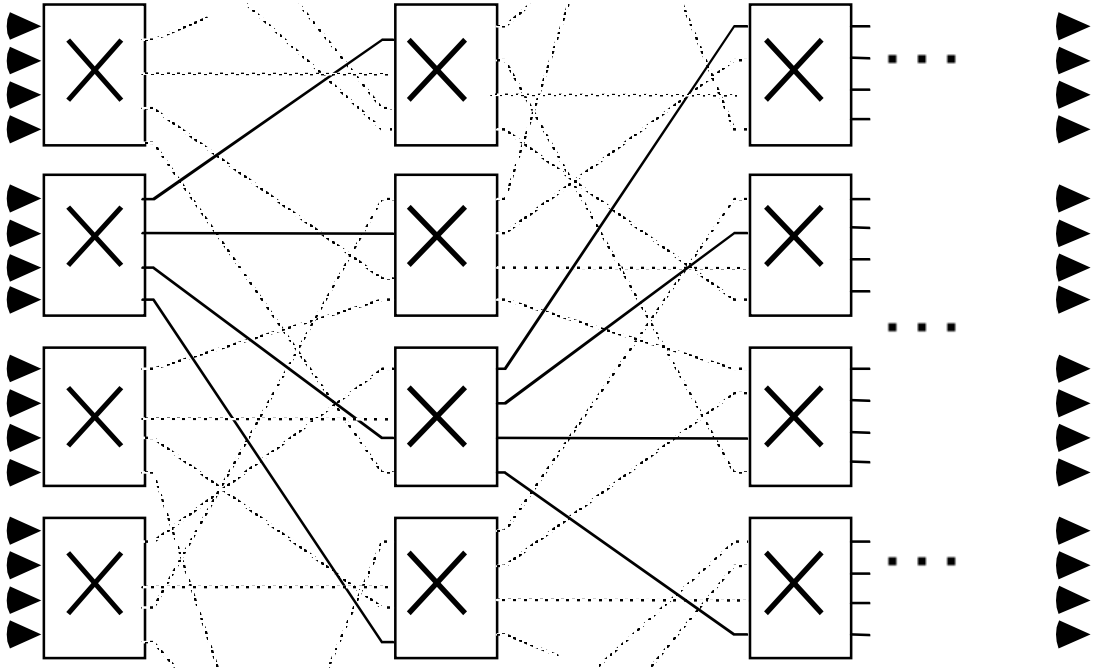
- Koppelfeldtechnik
  - elektrische Durchschaltung: Raummultiplex
  - Bits/Bytes vermitteln: Zeitmultiplex, Speichervermittlung
  - Zellvermittlung
  - Paketvermittlung

3.1 elektrische Durchschaltung

- Einheit der Vermittlung: Elektronen
  - Raummultiplex
  - Manuelle Vermittlung
  - Heb-Dreh-Wähler
  - Matrixschalter



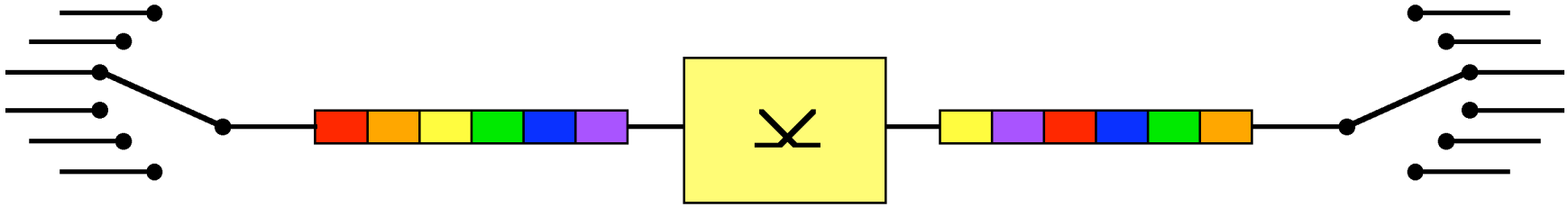
- Elektronische Schalter



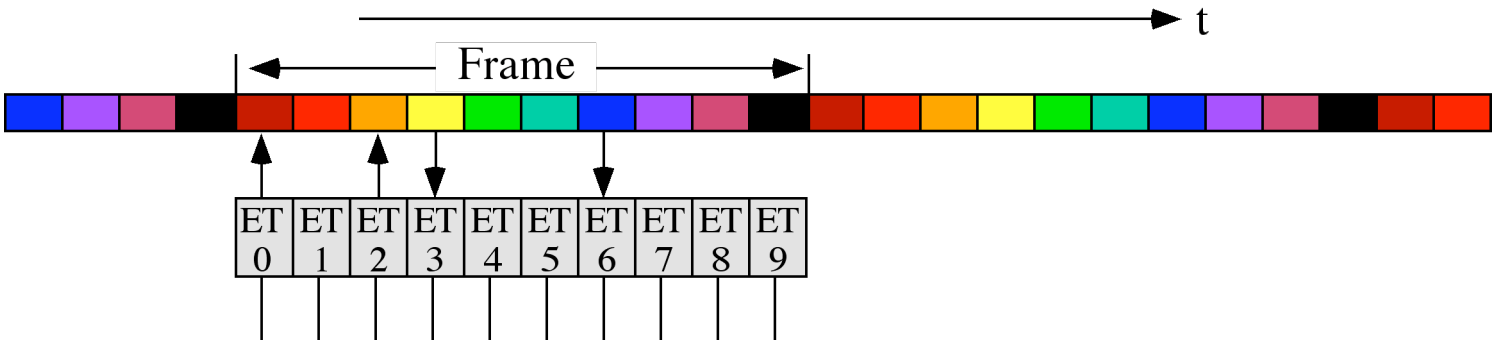
- Aktuelle Variante: Photonic Networks
  - Einheit der Vermittlung: Photonen
  - Licht wird end-to-end durchgeschaltet
  - Wavelength-Division-Multiplex
  - Wavelength-Router; Idee: Prisma

3.2 Bit- und Bytevermittlung

- Zeitlagewechsel

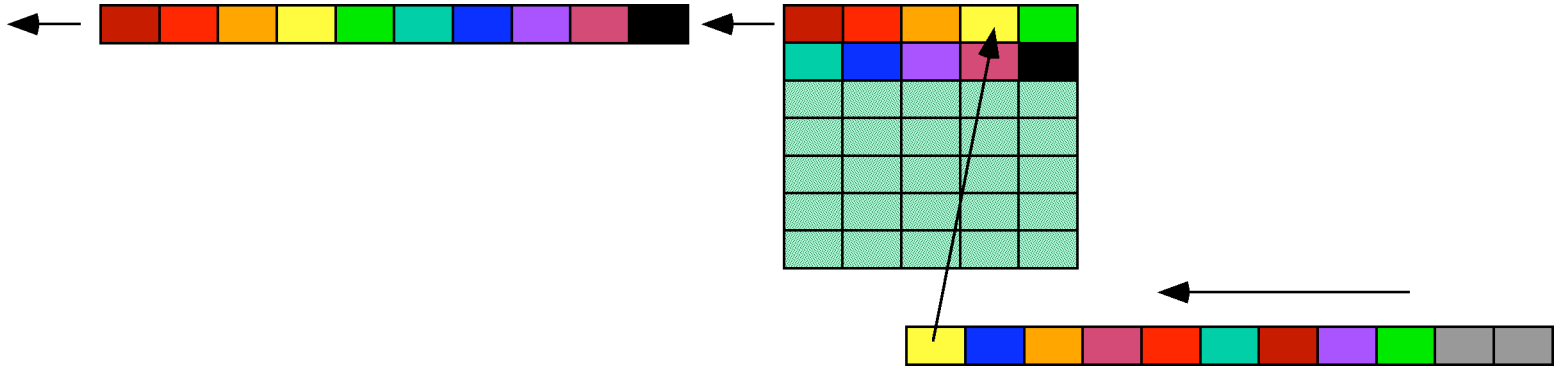


- Zeitkoppelfeld
  - Bus mit hoher Übertragungsrate
  - Zeitschlitztechnik



- Vermittlung einfach
  - Frame[j]:=ET[q];ET[p] := Frame[j];
  - Verbindungssteuerung konfiguriert ETs
- Multicast einfach: m Einheiten empfangen

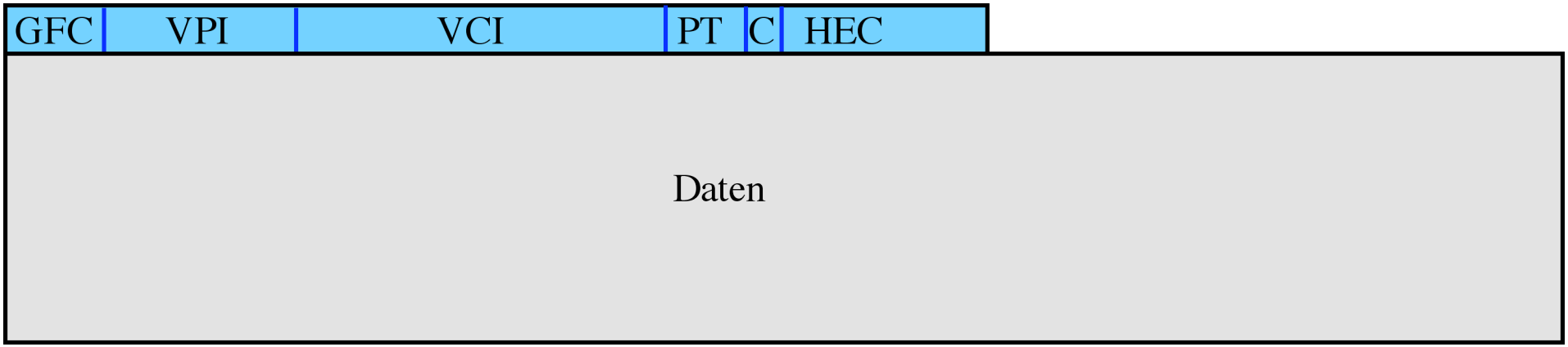
- Sonderfall Ethernet: Pakete statt Bytes
- Speichervermittlung: Memory Time Switch
  - Bytes kommen im Multiplexstrom an
  - Eingabe: Bytes werden einzeln adressiert und in RAM geschrieben
  - Ausgabe: Bytestrom wird sequentiell aus RAM ausgelesen
  - Ähnlich Video-RAM (VRAM)



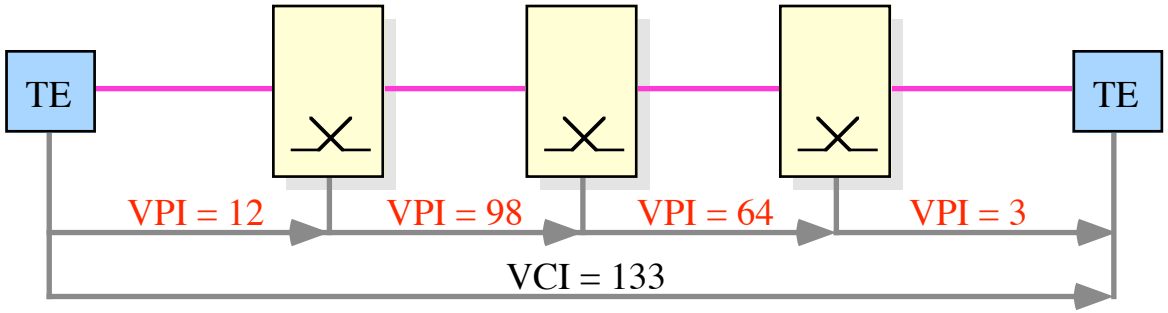
- Verbindungsteuerung
  - Adresstabelle verwalten
  - Tupel (Zeitschlitznummer, Speicheradresse)
- Multicast = ein Byte in mehrere Speicherzellen schreiben

3.3 Zellvermittlung

- 53 Bytes = 48 Byte Nutzlast + 5 Byte Verwaltung

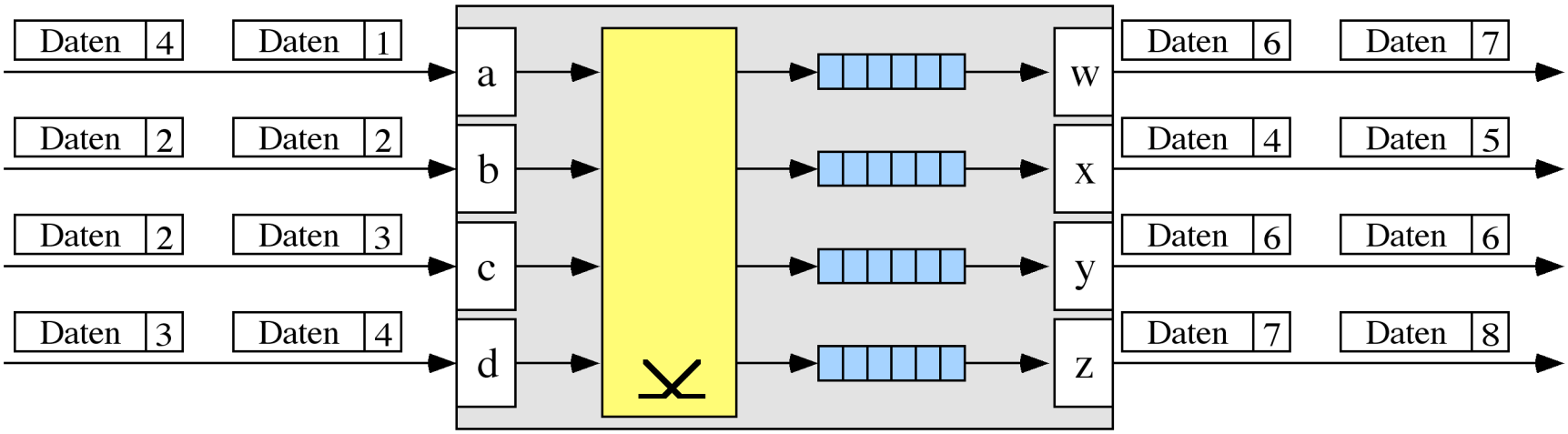


- Header nur auf dem Link (SONET, SDH) konstant
  - in jeder Vermittlung geändert (evtl. auch VCI!)
  - Tabelle mit (inLink, inHeader, outLink, outHeader)



- Vergabe der Header beim Verbindungsaufbau
  - Protokoll Q.2931 ähnlich ISDN

- Adressbasierte Vermittlung
  - (Eingang, Kennzeichen$_{n-1}$) -> (Ausgang, Kennzeichen$_n$)
  - Dynamische Steuerung der Durchschaltung
  - Eingangspuffer und Ausgangspuffer



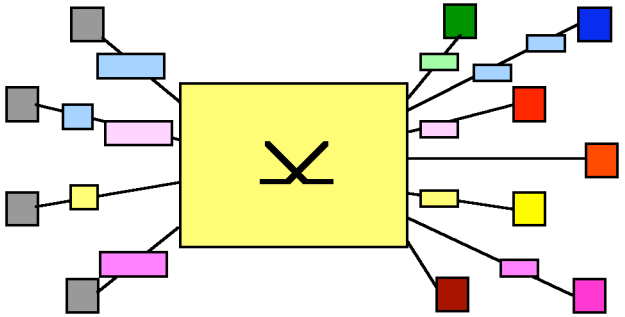
- Adressaustausch

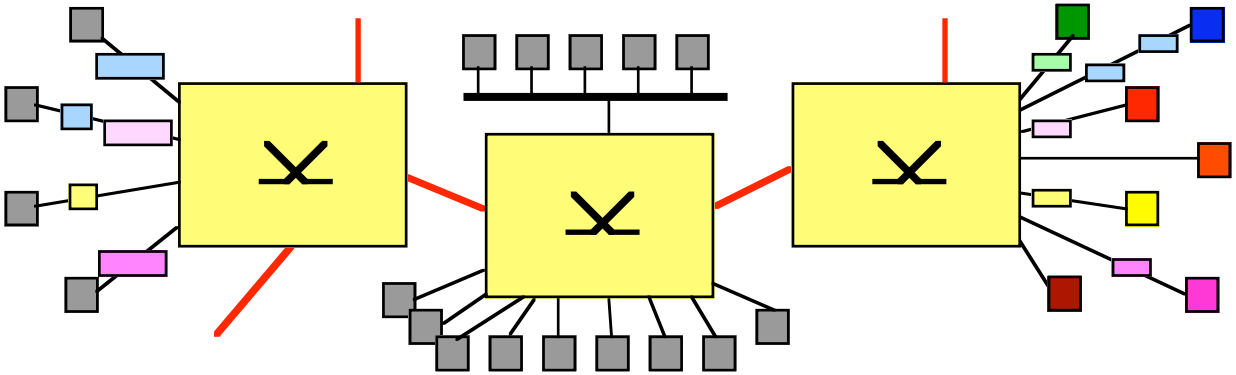
Port in	Header	Port out	Header
a	1	x	5
a	4	w	6
b	2	y	6
b	4	y	7
c	3	z	8
c	2	x	4
d	4	w	7
d	3	z	7

3.4 Paketvermittlung

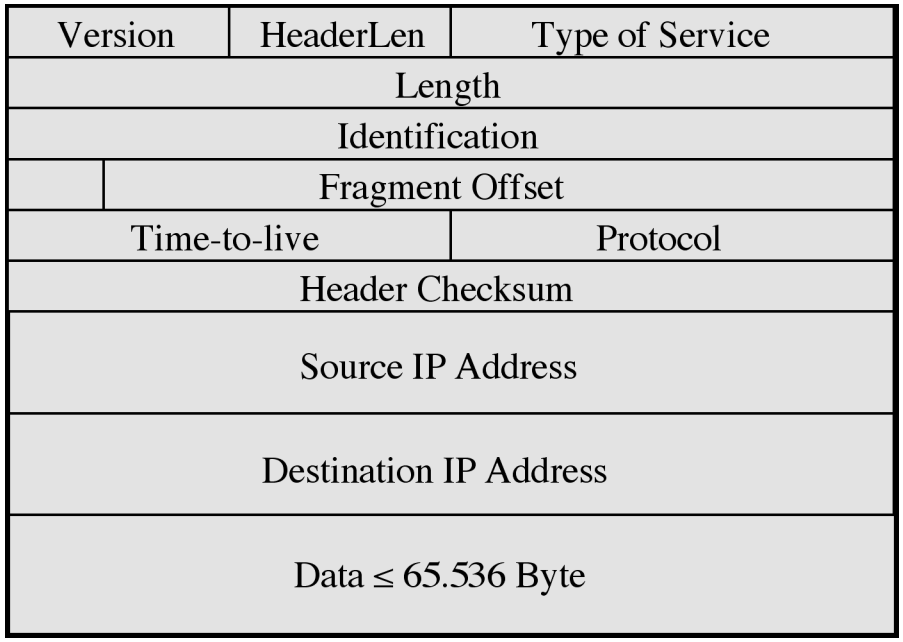
- Paket als Einheit der Vermittlung
  - variable Länge
  - Verbindungskennzeichen im Paket
- Vermittlung
  - m Eingangsleitungen
  - n Ausgangsleitungen
- Verbindungskennzeichen
  - bestimmt Ausgangsleitung

  => Routing, Weglenkung
- 'Verbindungslose' Paketvermittlung
  - Adressabbildung bei jedem Paket



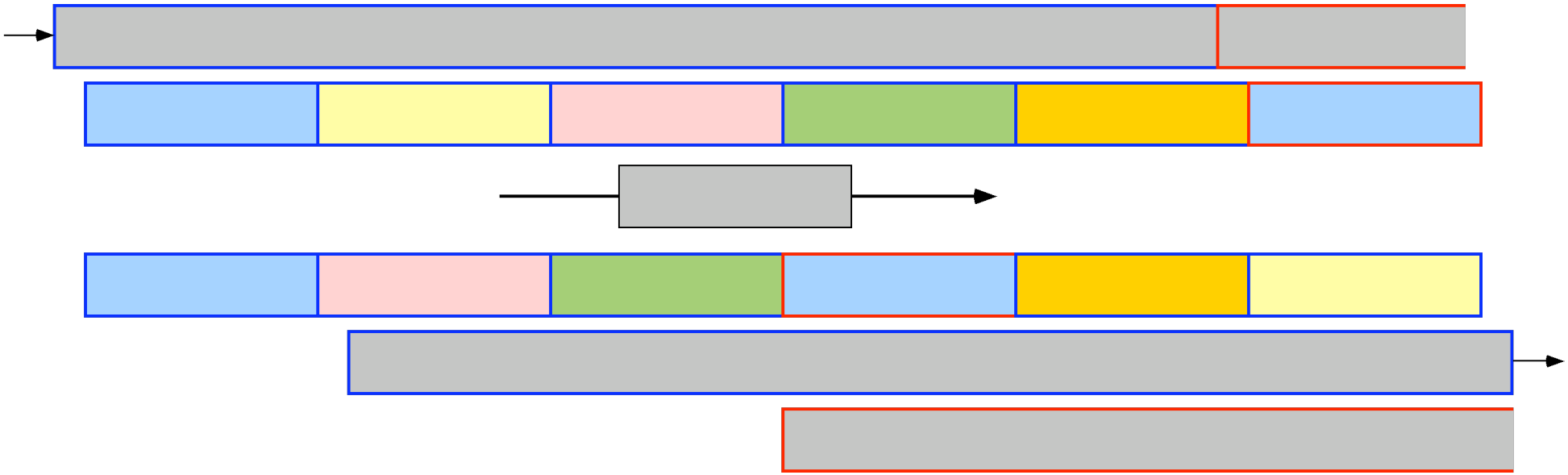


- Internet Protocol (IP)
  - Header mit vielen Feldern
  - Adressfelder siehe 1.7



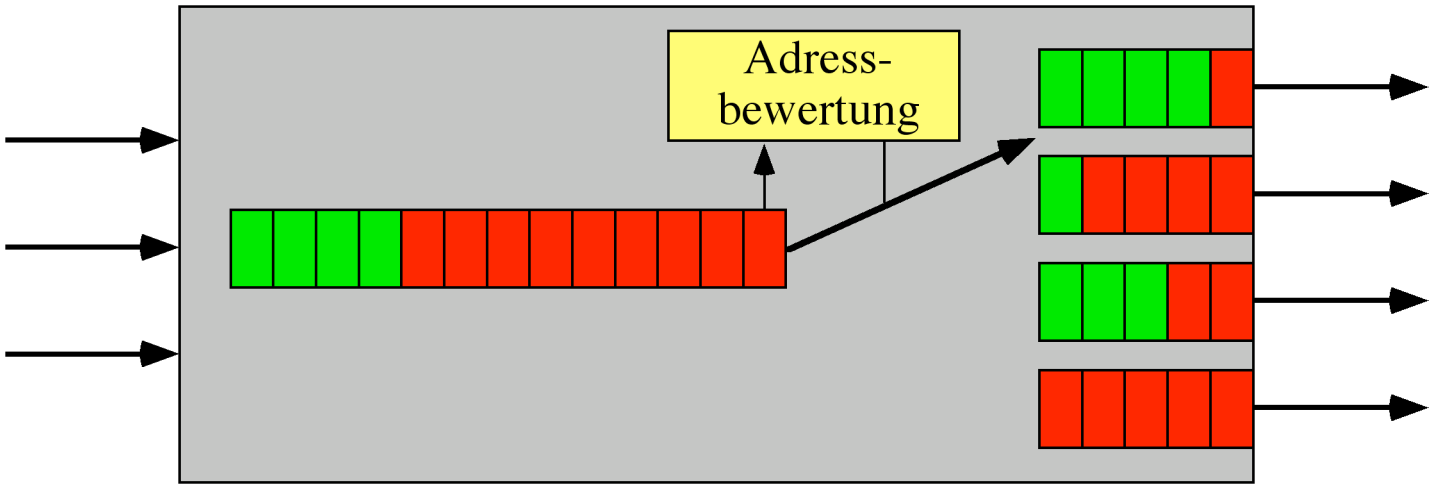
- Protokollfunktionen
  - Fragmentierung
  - Assemblieren (Reihenfolge!)
  - Kreisen der Pakete verhindern (TTL)

- Fragmentierung
  - Fragmentierung (Tokenring <4500, Ethernet <1500, …)
  - 1 PDU => n MTUs
  - Größe kann sich auf dem Weg ändern
  - Assemblieren (Reihenfolge!)



3.4.1 Routertechnik

- Store&Forward
  - Eingangsschlange sammelt Pakete
  - Ausgangsschlangen vor jeder ausgehenden Leitung
  - Adressauswertung ordnet Pakete Queues zu

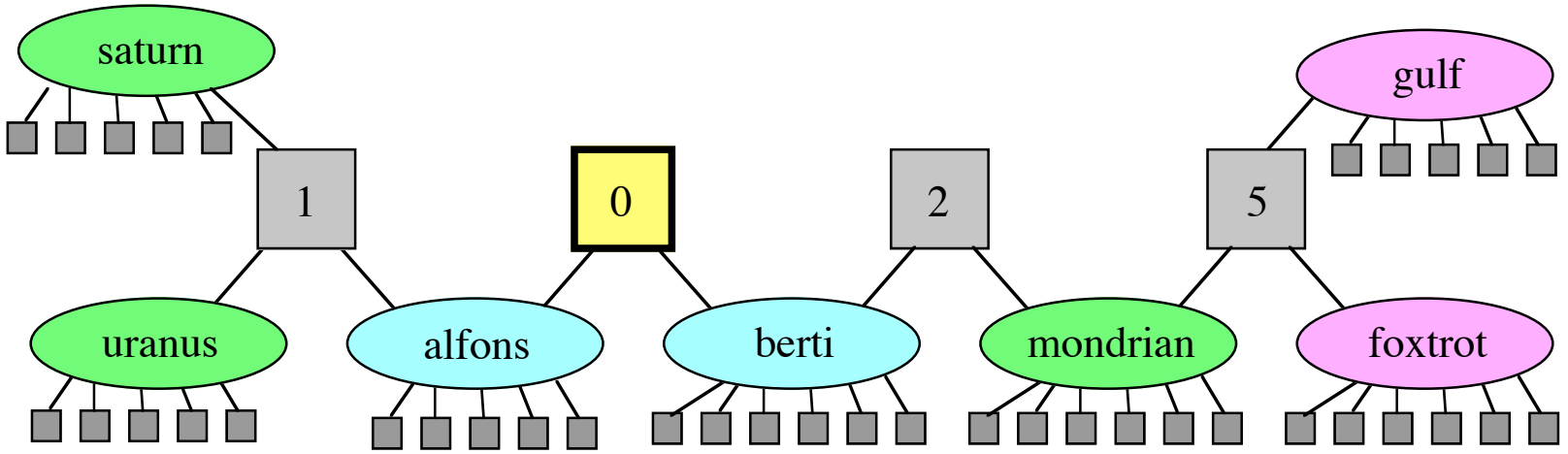


- Header Processing + DMA
  - Input-Queue enthält Pakete
  - Output-Queues enthalten Zeiger auf Pakete
  - Adressbewertung erzeugt Output-Queue-Elemente
  - Direct-Memory-Access von den I/O-Interfaces

- Beispiel Internet Weglenkung
  - von ara.informatik.tu-freiberg.de nach www.berkeley.edu
  - Programm traceroute

```
traceroute to arachne-lb.berkeley.edu (169.229.131.92), 64 hops max, 40 byte packets
 1  139.20.16.254 (139.20.16.254)  14.654 ms  20.434 ms  0.627 ms
 2  139.20.201.240 (139.20.201.240)  1.168 ms  0.576 ms  0.610 ms
 3  xr-dre1-ge8-1.x-win.dfn.de (188.1.35.165)  4.884 ms  4.766 ms  5.349 ms
 4  xr-pot1-te2-3.x-win.dfn.de (188.1.144.213)  22.298 ms  21.984 ms  22.489 ms
 5  xr-fra1-te2-4.x-win.dfn.de (188.1.145.137)  22.454 ms  22.971 ms  22.639 ms
 6  dfn.rt1.fra.de.geant2.net (62.40.124.33)  22.118 ms  22.275 ms  26.124 ms
 7  abilene-wash-gw.rt1.fra.de.geant2.net (62.40.125.18)  114.030 ms *  114.277 ms
 8  atlang-washng.abilene.ucaid.edu (198.32.8.65)  130.817 ms  130.407 ms  134.594 ms
 9  hstnng-atlang.abilene.ucaid.edu (198.32.8.33)  151.466 ms  150.996 ms  150.871 ms
10  losang-hstnng.abilene.ucaid.edu (198.32.8.21)  195.027 ms  189.094 ms  189.005 ms
11  hpr-lax-gsr1--abilene-la-10ge.cenic.net (137.164.25.2)  189.082 ms  189.076 ms  188.951 ms
12  svl-hpr--lax-hpr-10ge.cenic.net (137.164.25.13)  196.538 ms  196.466 ms  196.180 ms
13  hpr-ucb-ge--svl-hpr.cenic.net (137.164.27.134)  197.780 ms  198.091 ms  197.761 ms
14  g3-17.inr-202-reccev.berkeley.edu (128.32.0.35)  197.615 ms  197.587 ms  197.914 ms
15  t1-1.inr-211-srb.berkeley.edu (128.32.255.43)  197.499 ms  197.732 ms  197.749 ms
16  arachne-lb.berkeley.edu (169.229.131.92)  198.253 ms  198.019 ms  198.385 ms
```

- Adressabbildung



- Tabelle (Netz, Kosten, Gateway)
- Tabelle beginnt mit lokaler Information

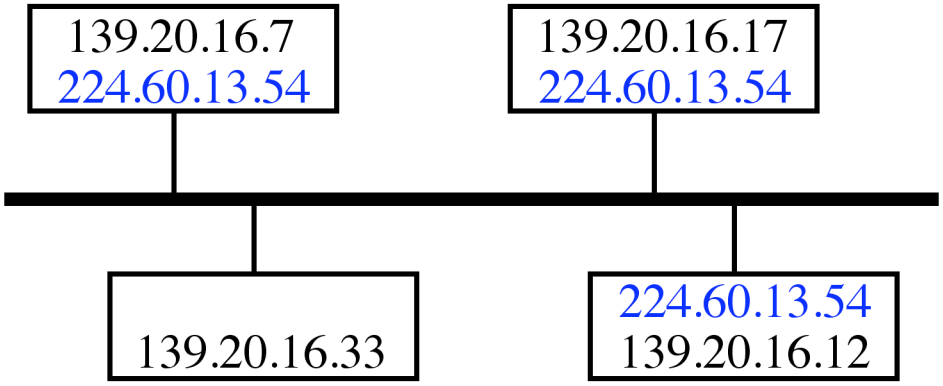
Netz	Kosten	Gateway
alfons	0	0
berti	0	0
saturn	1	1
uranus	1	1
mondrian	1	2
foxtrot	2	2
gulf	2	2

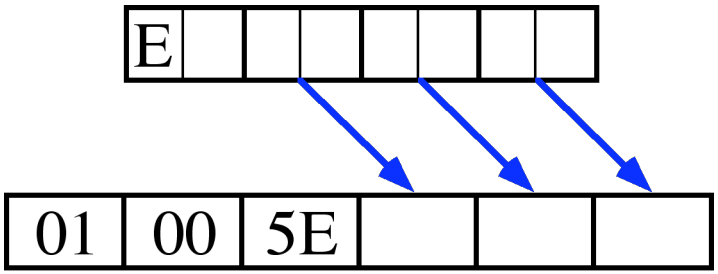
- Mitteilungen von anderen verbessern Tabelle (DVA)
- im Prinzip vollständiges Wissen nötig

- Strategie des Routing
  - Statische Verfahren mit vorher bestimmten Tabellen
  - Dynamische, verteilte Verfahren
  - DVRP: Distance Vector Routing Protocol
  - Metrik: Hops, Verzögerung, Auslastung, …
  - Link State OSPF: Link State Open Shortest Path First
- Cut-Through-Routing
  - Adressauswertung sehr schnell
  - Paketelemente (Bytes) sofort in den Ausgang bewegen
  - Problemfälle konventionell verarbeiten
- Multicastrouting
  - Pakete im Router replizieren
  - wohin schicken?
  - überall ist einfach
  - selektiv: Multicastbaum
  - evtl. unbekannt

MBONE: Multicast Backbone

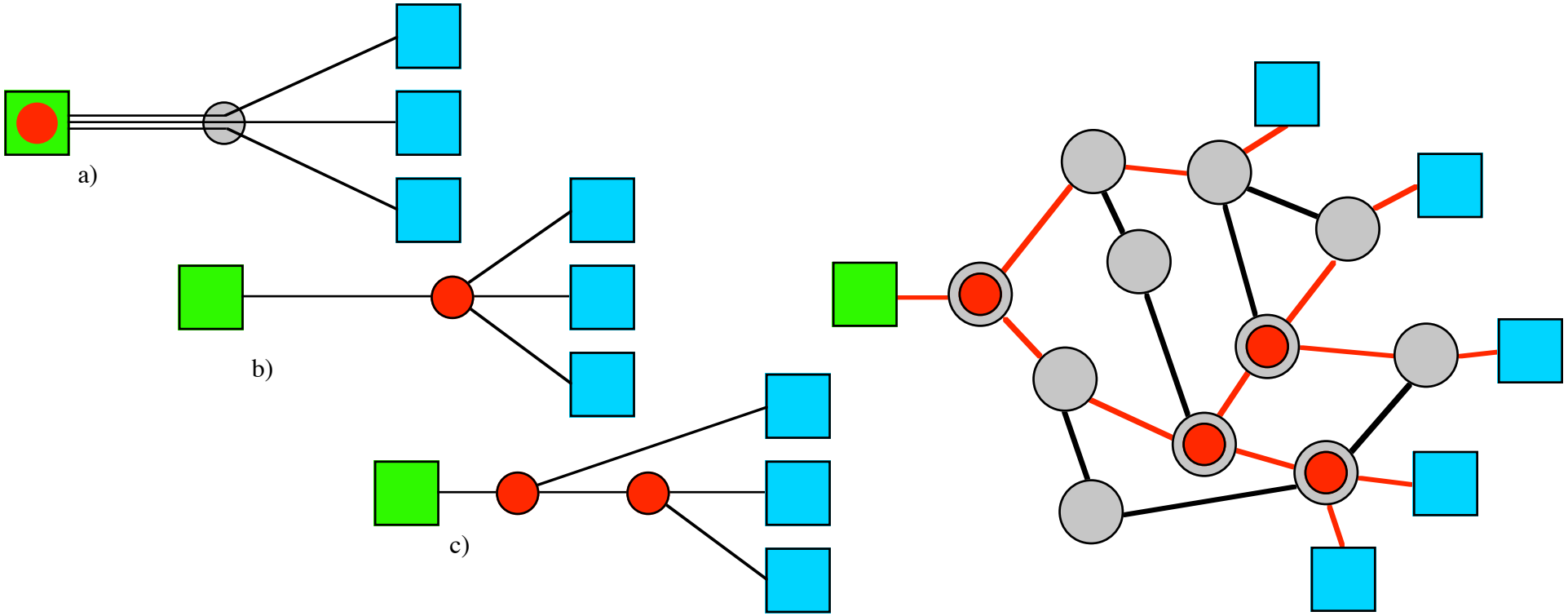
- IP-Multicast
  - Steve Deering
  - RFC 1112
- spezieller Bereich von IP-Nummern
  - Class-D Adressen: En.nn.nn.nn
  - 224.0.0.0 to 239.255.255.255
  - SendIP normal
  - JoinHostGroup (group-address, interface)
  - LeaveHostGroup (group-address, interface)
  - Besondere Adressen zum Management
- z.B. Ethernet Multicast



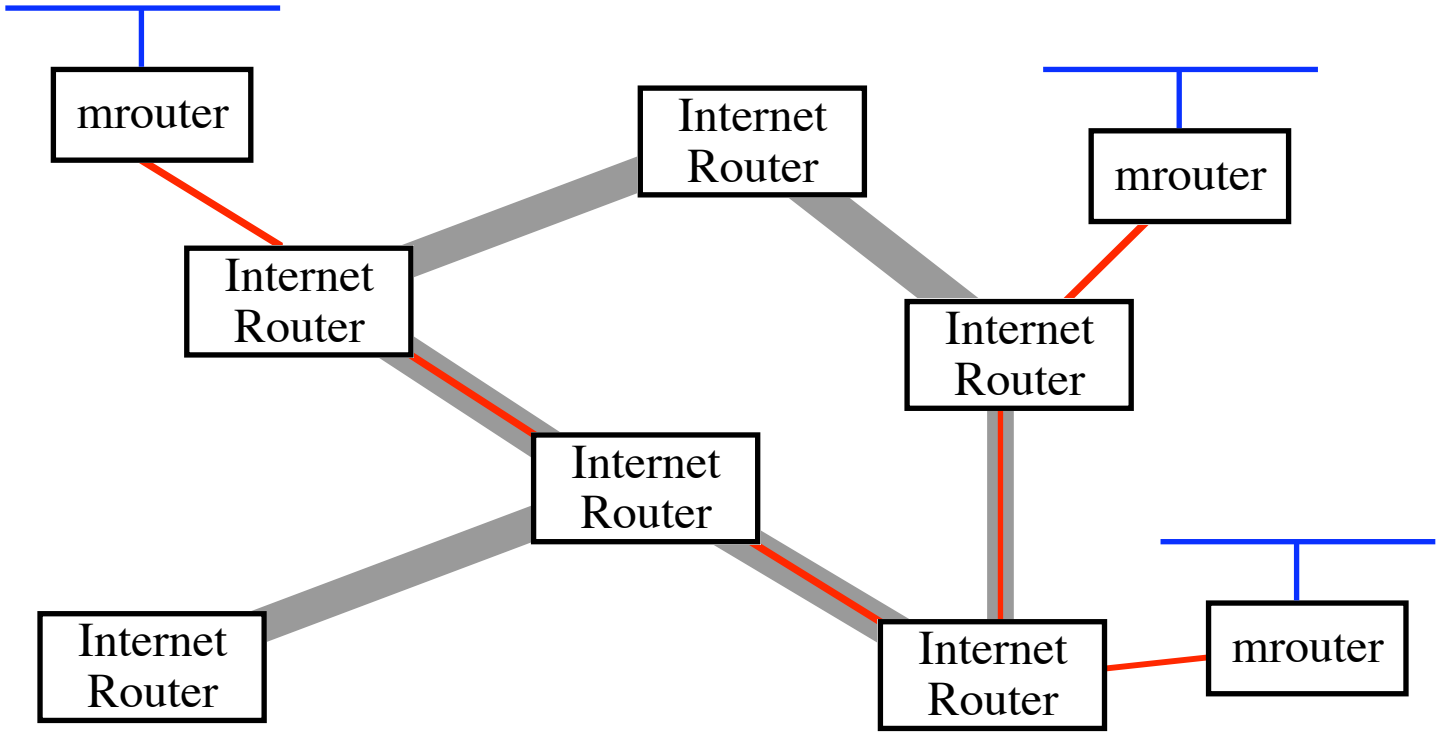


  - Ethernet Interface entsprechend konfigurieren

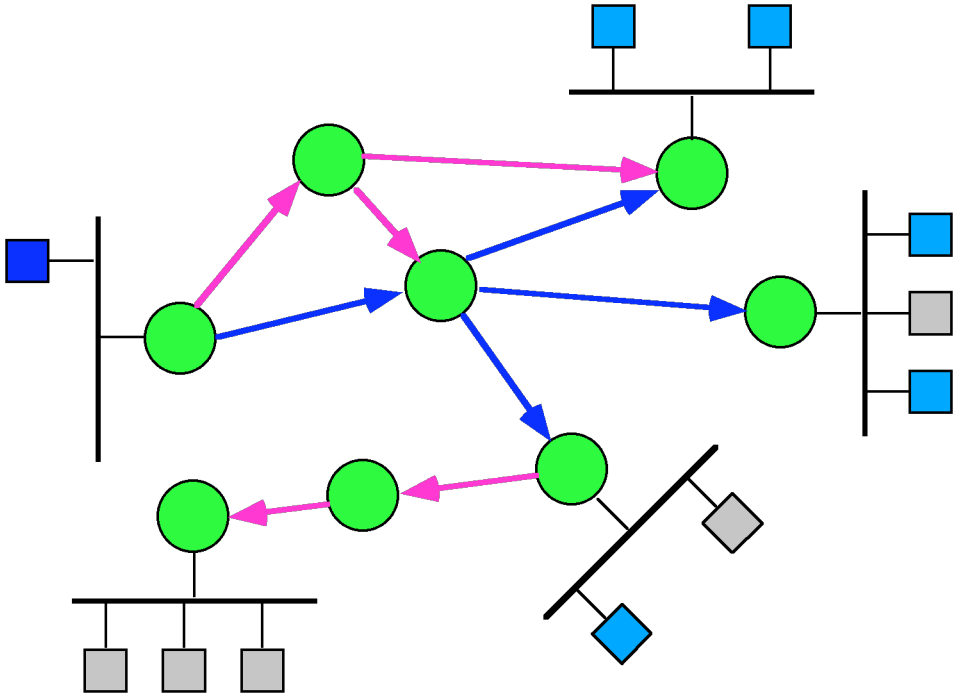
- Multicast-Router
  - Nachrichten-Replikation



- MBone
  - mrouted in einer Workstations im LAN
  - Tunnel zum Transport durch das normale Internet
  - zum nächsten mrouted
  - Multicast-Pakete in normale IP-Pakete kapseln
  - bilden Overlay Netz (virtuelles Netz)

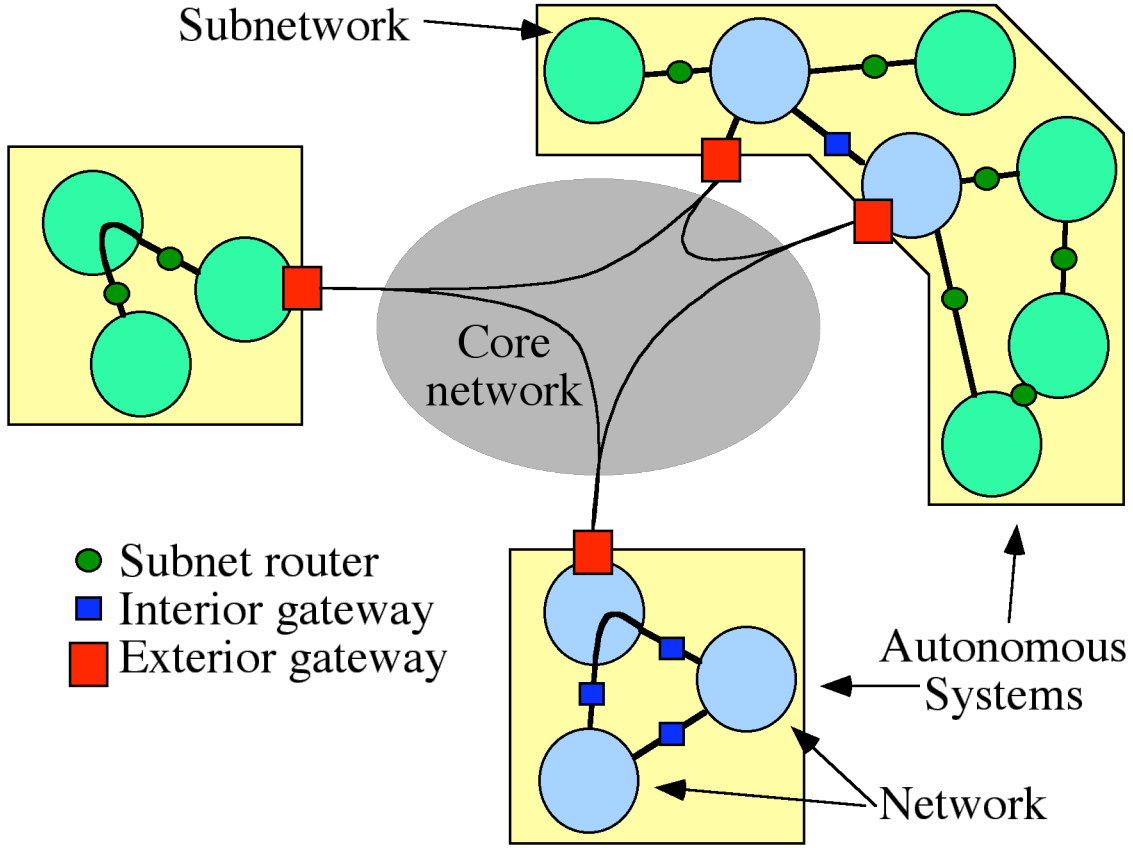


- DVMRP - Distance Vector Multicast Routing Protocol ('dense mode')
  - 'flood and prune'
  - 'grow back' nach einer gewissen Zeit

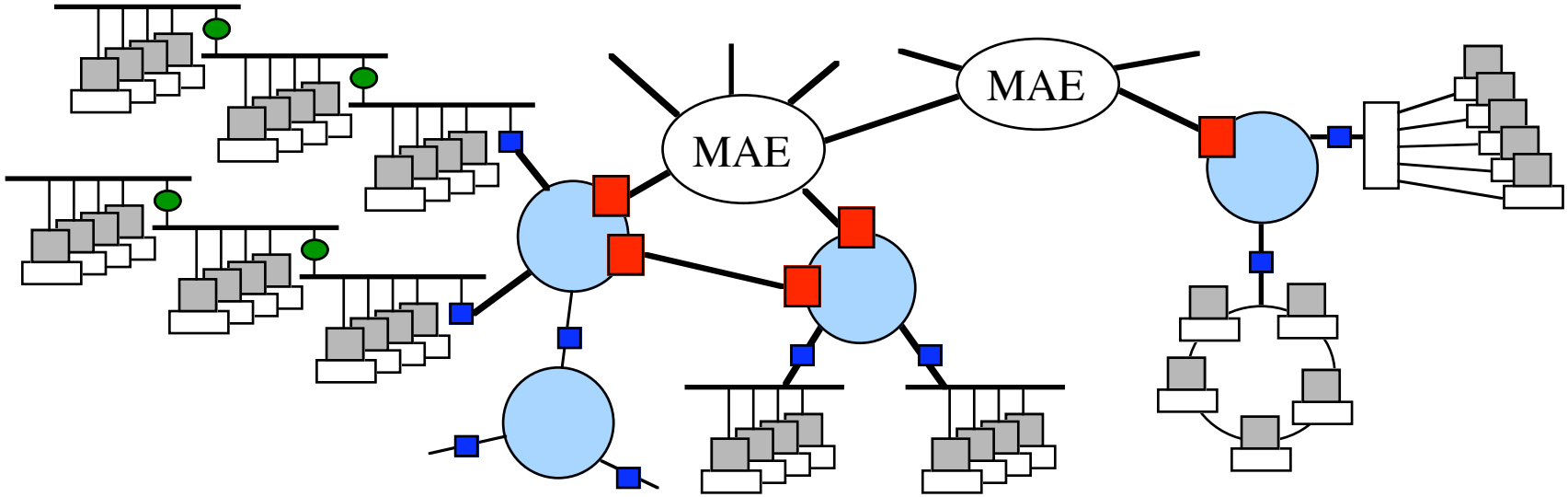


- Sparse Mode
  - Mehr Wissen über Multicast-Baum
- PIM Protocol Independent Multicast ('sparse mode')
  - Beitrittsnachricht an bekannte Rendezvous-Punkte
- MBONE tools (vat, nevot, ivs, nv, vic, sd, wb, rat, …)

- Internet mit mehreren Netzwerken
  - verschiedene Routing-Algorithmen in Teilnetzen
  - Subnetwork (subnetid, z.B. 139.20.16.80)
  - Network (netid, z.B. 139.20.77.80)
  - Interior Gateways
  - Exterior Gateways



- Das Internet
  - heterogene Netzwerke
  - Router
  - homogenes Internet-Protocol (IP)

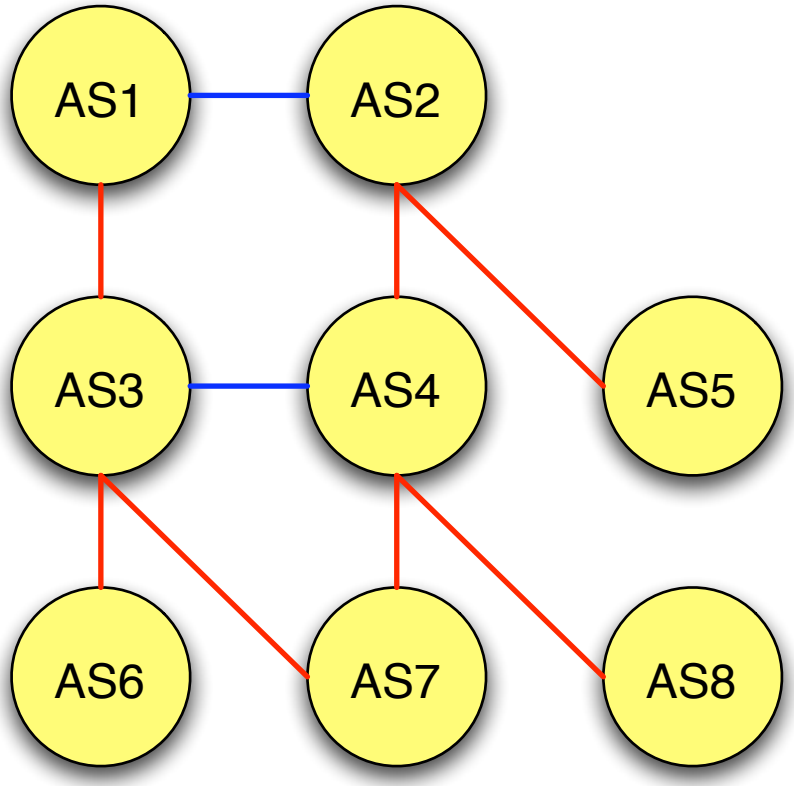


- Wer bezahlt?
  - 'Peering' Abkommen zwischen Netzwerken: Volumenverhältnis
  - Durchleitung zu Subnetzwerken: Pakete, Volumen
  - Abrechnung mit Endkunden: pauschal, Zeit, Volumen, Pakete
  - Peers gleichen Kosten aus
  - Abhängige: decken Kosten und erzeugen Ergebnis

3.4.2 Routingalgorithmen und Protokolle

- Distance-Vector-Algorithm [Bellman-Ford]
  - Enfernung als Metrik: Delay, Anzahl Hops, …
- Ablauf
  0. jeder Router kennt nur seine Netzwerke + Kosten dorthin
  1. jeder versendet Tabelle an die Nachbar-Router
  2. Tabellen-update: for (alle nets in den angekommenen Tabellen)
  2a. D_net:=D(zum Nachbarn)+D(vom Nachbarn);
  2b. if RTable[net] leer: RTable[net].dist:=D_net;
  2c. else RTable[net].dist:=min(RTable[net].dist, D_net);
  3. weiter mit 1.
  4. Tabelle hat vollständiges Wissen über Netzwerk
- Routing Information Protocol: RIP
  - Interior gateway protocol (IGP)
  - hello-messages alle 30 Sekunden
  - timer sorgen für Ausfallsicherheit
  - count-to-infinity-Problem

- Border Gateway Protocol zwischen autonomen Systemen
  - Border Gateway hat Routing-Tabelle für eigenes Netz
  - Border Gateway kennt all Router im Core-Network
  - Transit-System oder Stub-System
  - Versenden Listen mit Präfixen (= IP-Bereichen)
  - Austausch per TCP-Verbindung
  - Modell ähnlich DVA
  - Routing-policy!
  - OK: 6-3-1-2-5
  - schlecht: 6-3-4-2-5
  - gut: 6-3-4-8



Bsp. nach I.v.Beijnum, Ars Technica, 2010

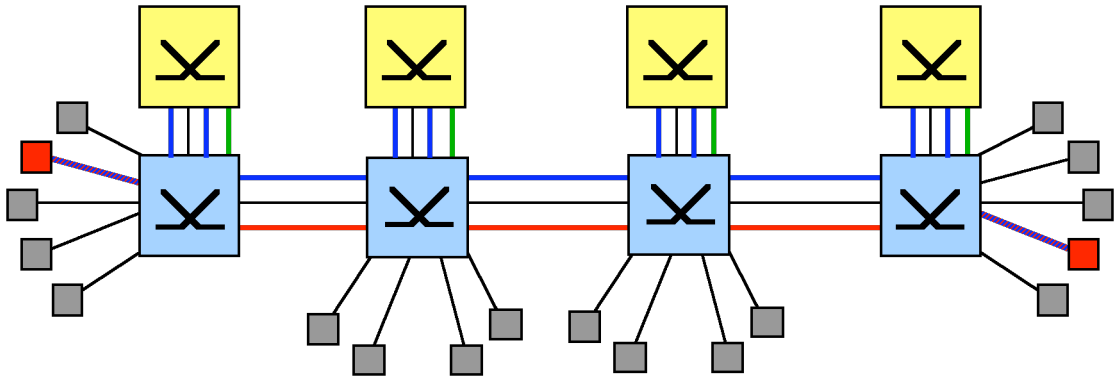
- RIP bis 1979 im Internet
  - Queue-Länge als Metrik
  - Schlechte Nachrichten verbreiten sich langsam
  - Tabellennachrichten verkehrsintensiv
- Link State Verfahren
  - ab 1979

  1. Nachbarn und deren Netzadressen entdecken (HELLO-Paket)
  2. Metrik-Parameter zu Nachbarn messen (ECHO-Paket)
  3. Link-State-Paket zusammenstellen (auch LSA)
  3. Link-State-Paket an alle Router schicken (Flooding)
  4. kürzesten Pfad mit besonderem Algorithmus berechnen

- Dijkstra-Algorithmus
  - berechnet günstigsten Pfad
  - basiert auf Kosteninformation über Leitungen
  - Graphenalgorithmus
- Link State Open Shortest Path First (OSPF)
  - Standard im Internet als Interior Gateway Protocol (IGP)
  - berücksichtigt verschiedene Metriken
  - hierarchische Systeme
  - Lastausgleich
  - Link-State-Update nur an logische Nachbarn
- Internet Control Message Protocol ICMP
  - Wartungsprotokoll
  - Desination unreachable, time exceeded, …
  - Echo Request und Reply

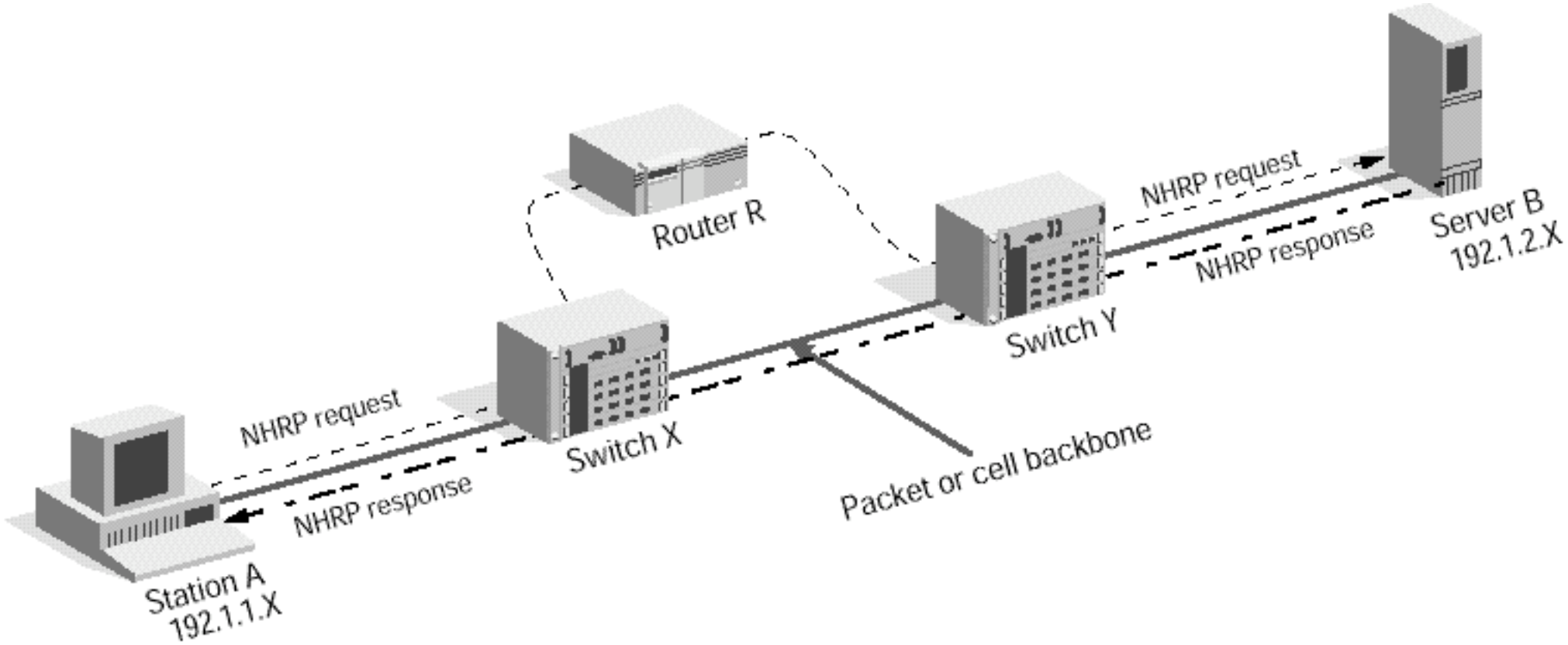
3.4.3 Layer-3 Switching

- Routing ist relativ teuer
  - IP-Adresse => physikalische Adresse
  - zeitkritische Datenströme: viele Pakete mit gleicher Adresse
- Ethernet-Vermittlungen
  - EN-Zieladresse als Vermittlungskriterium
  - Hardware zur Adresserkennung



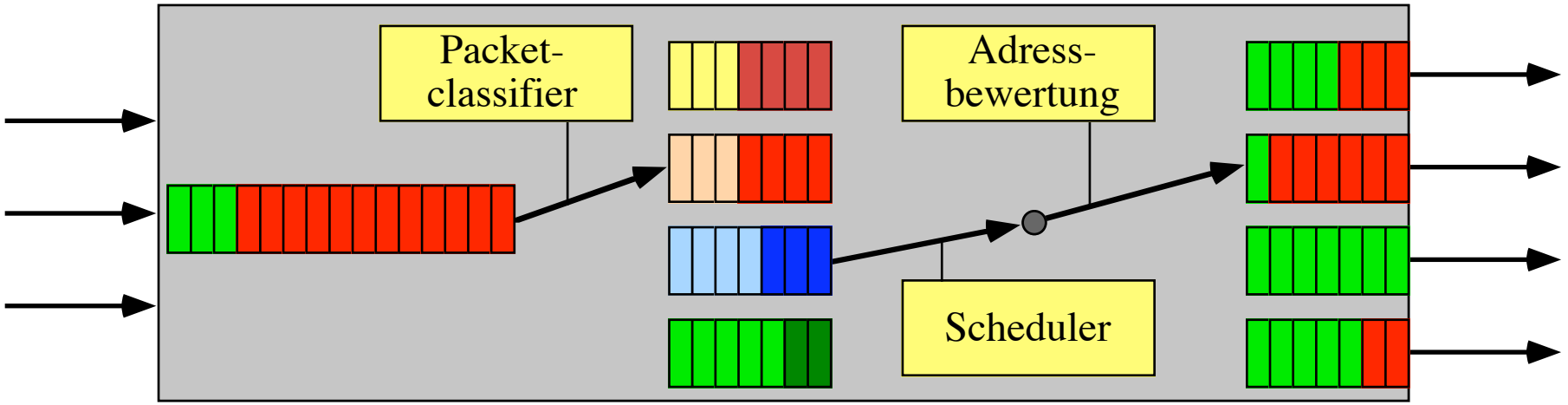
- Route first - then switch
  - Routing in Software (Tabellengrösse!)
  - Router finden Ethernetadresse des Zieles heraus
  - Ethernet-Switches konfigurieren
  - im ersten EN-Switch Zieladresse von Gateway auf Ziel ändern

- Fast IP (3COM)
  - Zunächst ganz normal Pakete schicken
  - Verbindungsaufbau mit NHRP-Request (Next Hop Routing P)
  - Router setzt Verbindung auf falls Switched-IP existiert
  - Endgerät erhält Ziel-Ethernet-Adresse



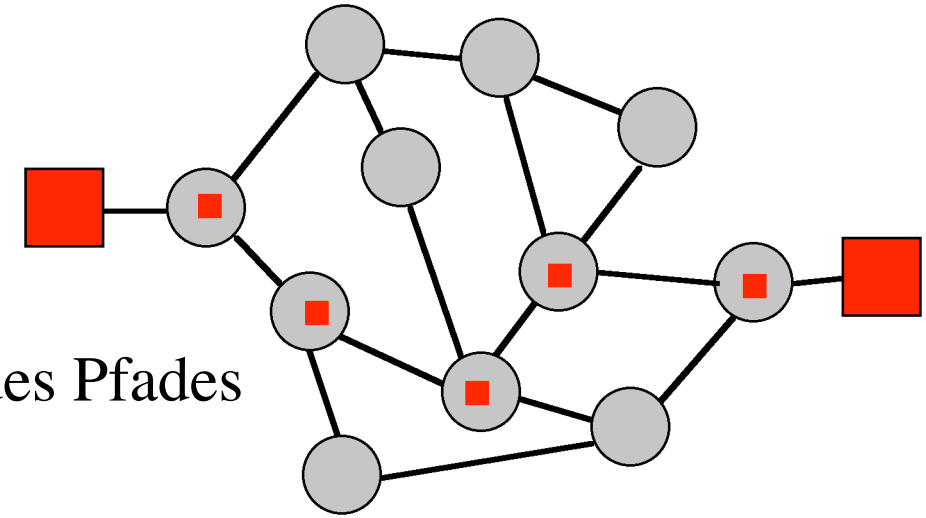
3.4.4 QoS im Internet

- Einfacher Ansatz: DiffServ (differentiated services)
  - mehrere Eingangsqueues
  - verschiedene Paketklassen
  - Klassifizierung mit Type_of_Service-Feld
- Priority-Queues
  - pro Klasse eine Queue

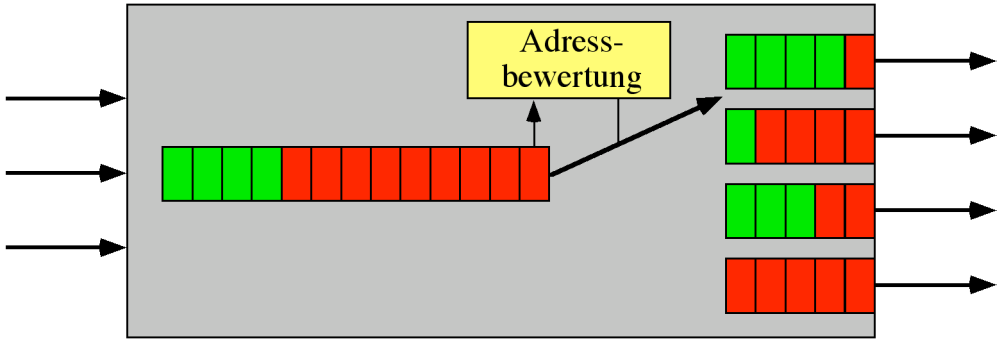


- Deja Vu
  - Prioritäten gab es schon oft
  - Welche Applikation benutzt niedrige Priorität freiwillig?
  - Hohe Priorität = hohe Kosten?

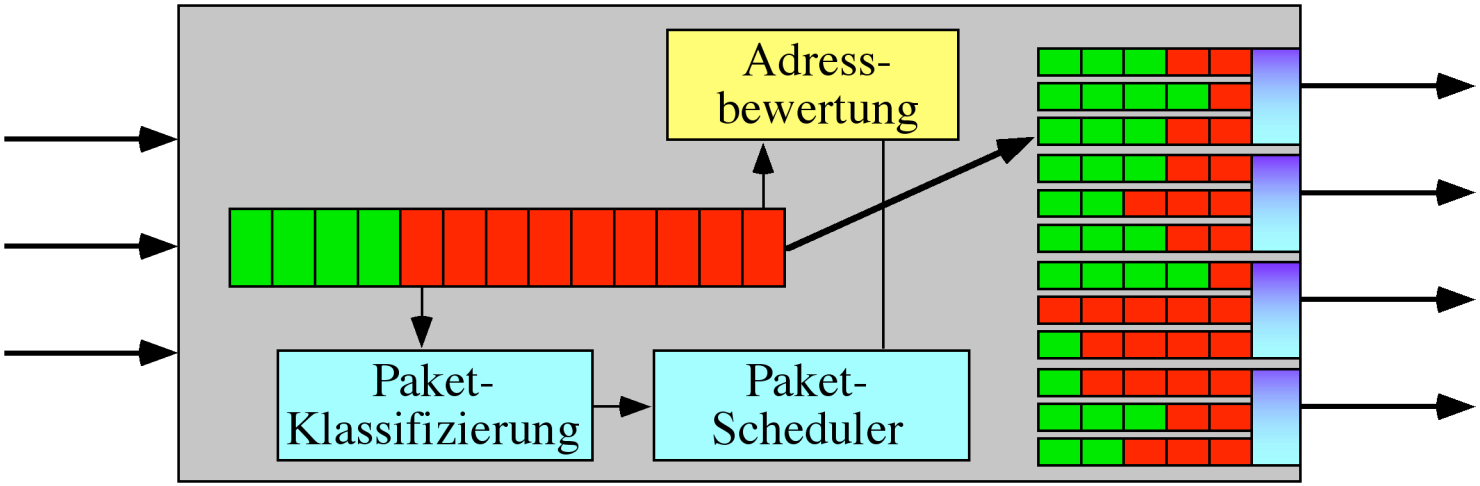
- Internet Integrated Services Architecture
  - IISA, IntServ
  - alle Einheiten im Pfad müssen QoS unterstützen
  - Reservierungsprotokoll
  - Zulassung zum Dienst
  - Überwachung
- QoS im Internet
  - erfordert Verbindungsparadigma
  - Reservierung von Ressourcen entlang des Pfades
  - eindeutiger Pfad?
  - Zulassungsbeschränkungen



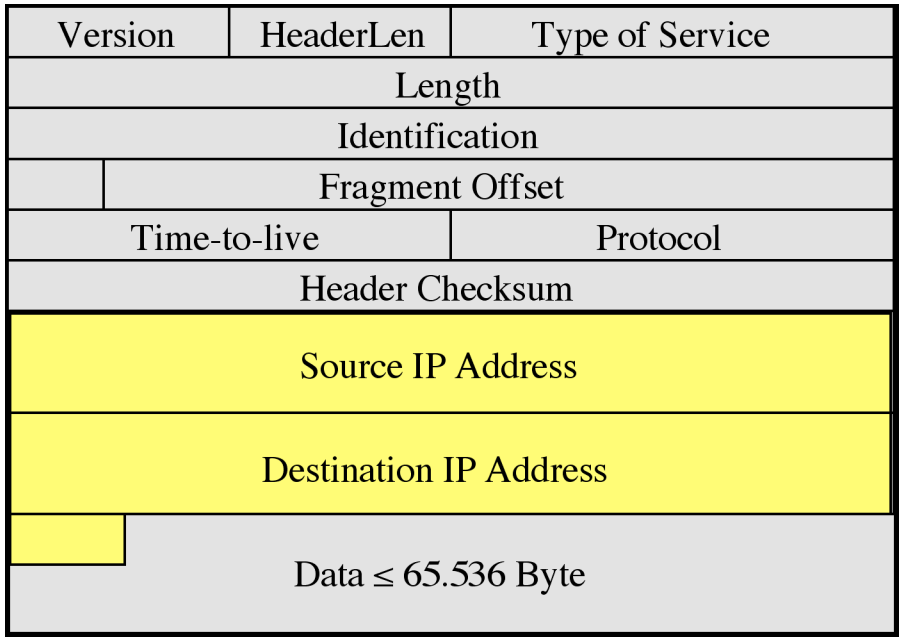
- Router: Store&Forward
  - Eingangsschlange sammelt Pakete
  - Ausgangsschlangen vor jeder ausgehenden Leitung
  - Adressauswertung ordnet Pakete Queues zu



- Paket-Scheduling



- Paketklassifizierung?
- Definition: Flow (~ Verbindung)
  - Identifikation von IP-Paketen
  - IPv6: flowlabel
  - IPv4?

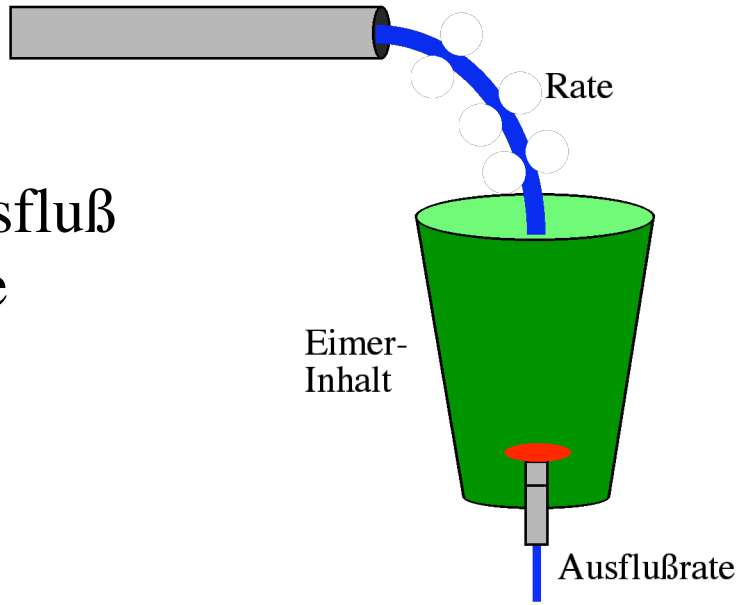


- Session + Sender_Template
  - Bezeichnung des Stromes durch den Sender
  - z.B. IP-Nummern, Port

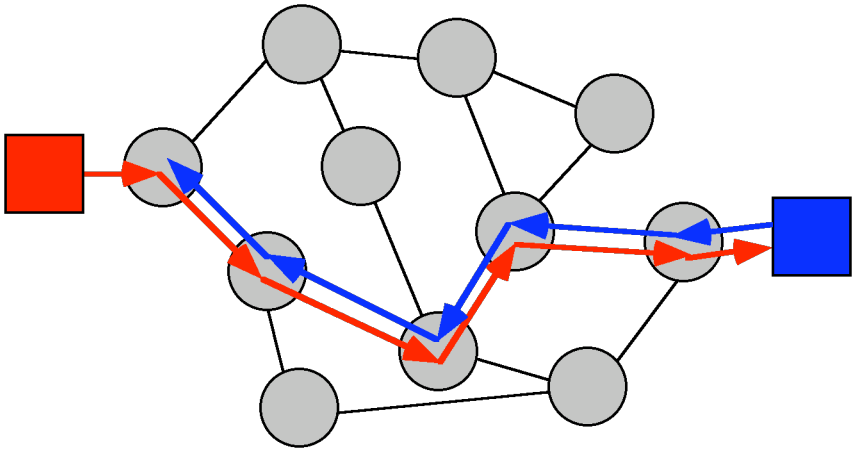
- referenziert beim Reservieren

- Filterspec
  - identifizieren Untermengen im Paketstrom
  - Verwerfen von Paketklassen
  - getrennt von der Reservierung
- Sender_TSpec
  - Obergrenze des Verkehrs vom Sender
- AdSpec
  - QoS-Schätzungen vom Netzwerk
  - Verzögerung, Durchsatz
  - wird auf dem Weg durch das Netzwerk aufgesammelt
- FlowSpec, Flow-Specification
  - rudimentäre QoS-Spezifikation
  - controlled delay
  - predictive QoS, guaranteed QoS
  - TSpec: Traffic-Specification: Peak Rate, Packet Size, Token-Bucket rate und size
  - RSpec: Reveiver-Specification: Rate

- Token Bucket Modell
  - Verkehr modellieren
  - Verkehr 'formen' (traffic shaping)
  - Zufluss von Daten in Bursts
  - Zufluss von Token mit konstanter Rate
  - Datensenden konsumiert Token
  - Bucket-Parameter beschreiben den Verkehrsfluß
  - Eimerinhalt entspricht Warteschlangenlänge
- Leaky Bucket Modell
  - Leckgrösse bestimmt konstanten Datenfluss
  - evtl. mehrere Löcher für mehrere Ströme
  - kein Ausgleich zwischen Strömen

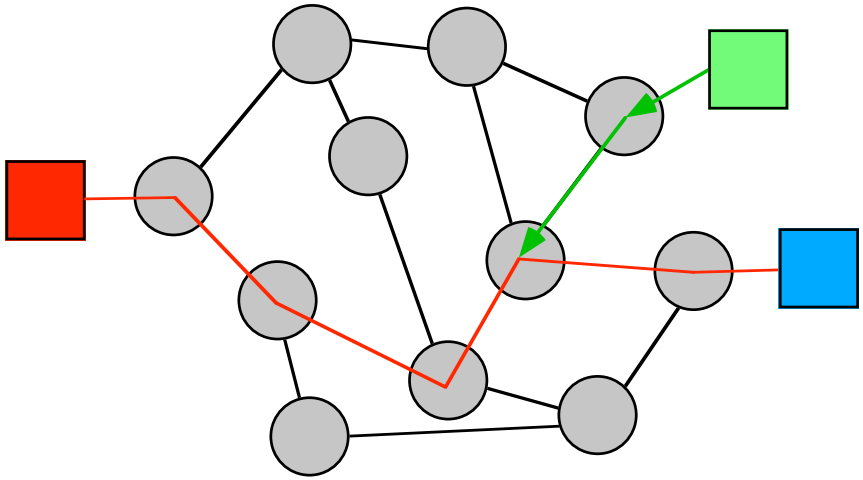


RSVP: ReSerVation Protocol



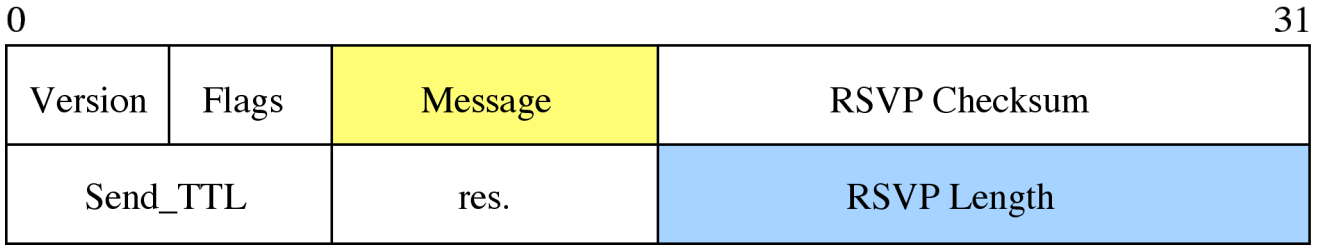
- Pfad-Nachrichten vom Sender
  - enthalten Flowspec
  - erzeugen 'Pfad' im Netz (path state Einträge in Routern)
  - enthalten Sender_Template, Session, Sender_TSpec
  - werden von Routern periodisch erneut gesendet (flooding)
- Reservation-Request
  - von den Blättern zur Wurzel
  - rückwärts entlang des Pfades
    1. Admission control
    2. Reservation: flowspec, filterspec
    3. Zusammenfassung von Res. Requests eines Multicast-Baums
  - eventuelles Scheitern upstream -> Freigabe

- IP-Multicast und MBone
  - Multicast-Baum
  - flood-and-prune
  - MBone: Multicast-Router und Tunnel
- 'Einklinken' weiterer Empfänger

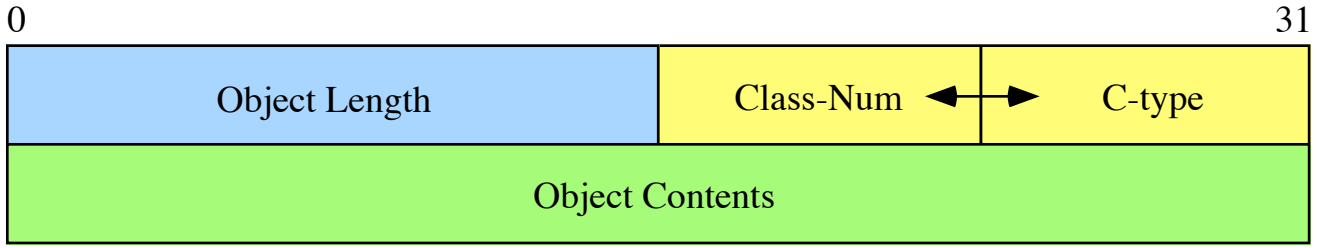


- Reservierung freigeben
  - upstream: ResvTear(session, style, filterspec)
  - downstream: PathTear(session, sender-template, …)
  - von Sender oder Empfänger
  - von Routern nach path-state timeout

- RSVP-Pakete
  - Path, Resv, Resv_Confirm
  - PathTear, ResvTear
  - PathErr, ResvErr

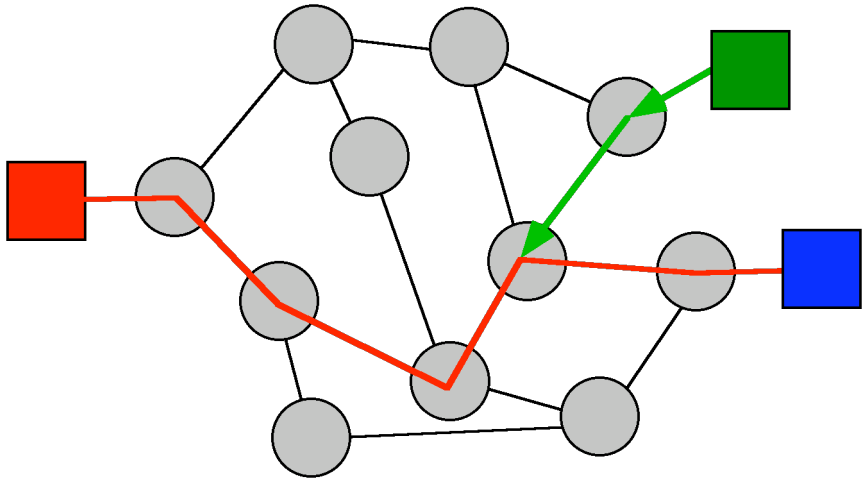


- RSVP-Objekte
  - session, sender_template
  - style, flowspec, filterspec
  - sender_tspec



  - Objekt-Klasse und Objekte

- Soft-State
  - Gruppen groß und sehr dynamisch (TV-Modell)
  - Fehlersituationen in Routern managen?
  - Reservierungen verfallen nach kurzer Zeit (path state timeout)
  - Quellen senden periodisch 'Path'-Nachrichten
  - Empfänger senden periodisch 'Reservation'-Nachrichten
  - Paketrate-Kontrolle
- Charging fehlt



4. Protokolle

- Ende-zu-Ende Datenübertragung
  - Abstraktion benutzter Dienste
  - Standardisierung von Operationen

4.1 Aufgaben und Mechanismen

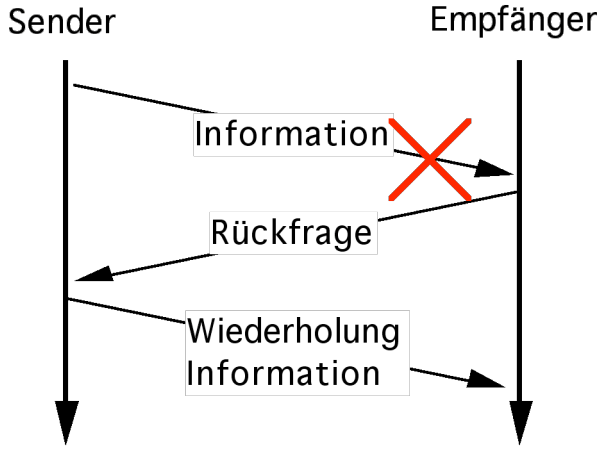
- Ende-zu-Ende Verbindungen
- Partner-Prozess finden
  - Portkonzept (siehe 1.7)
  - Object Request Broker (CORBA, siehe Vorlesung Verteilte Systeme)
- Verbindungsaufbau
  - Ressourcen anlegen, Dienstegüte verhandeln
- Datentransport
  - Pipeline-Charakter
  - Aufteilen und Verpacken, Fehlerkontrolle
- Verbindungsabbau (Aufräumen)
  - Ressourcen im Netz freigeben
  - Kontexte in Endgeräten löschen

4.1.1 Fehler kontrollieren

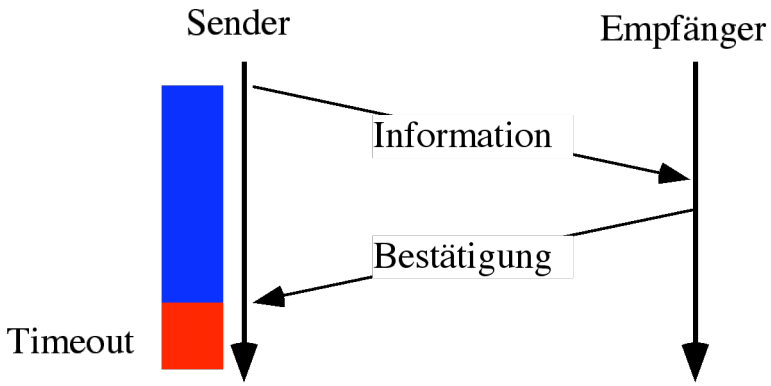
- Fehlerarten
  - Verlust der Meldung (Router, Puffer im Empfänger, …)
  - Verfälschung der Meldung (Bitfehler, Zelle verloren)
  - Reihenfolge, Duplikate
- Fehler erkennen
  - Sequenznummer
  - Prüfinformation (CRC - Cyclic Redundancy Check)
- Nachrichten bestätigen
  - Bestätigungen (ACK)
  - Rückfragen (NACK)
  - separate Pakete oder piggyback
- Fehlerkontrolle und Medien
  - Audio und Video brauchen keine Wiederholungen
  - Text und Bild tolerant gegen Bitfehler
  - verschiedene Empfindlichkeit auch in den Daten
  - Fehlerkontrolle sollte kontrollierbar sein!

4.1.1.1 a-posteriori Fehlerbehebung

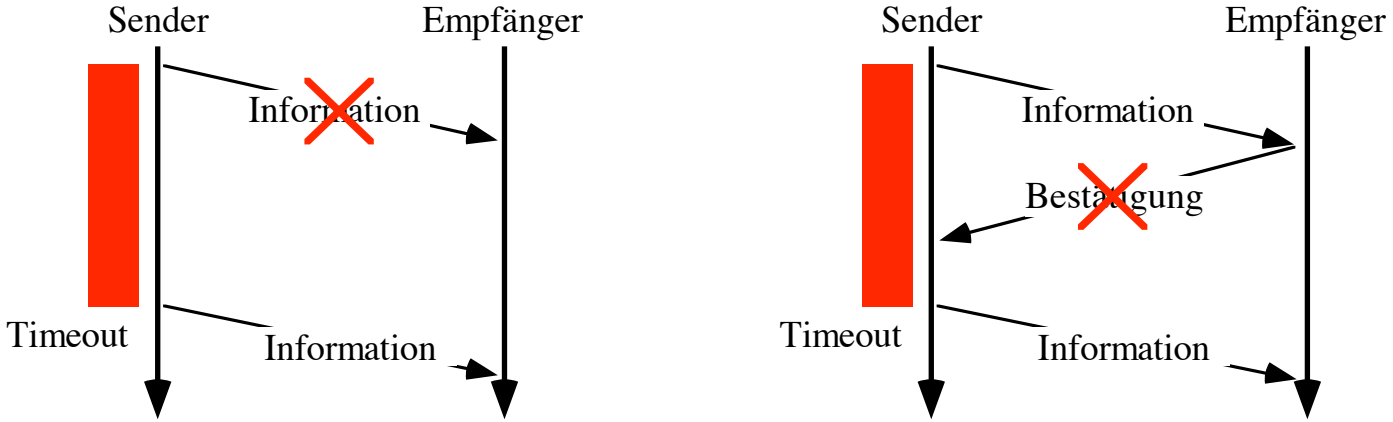
- Erneute Übertragung: Retransmission
  - fehlerhaftes/fehlendes Paket wird nochmal übertragen
  - go-back-n
  - selektive Wiederübertragung
- Aktive Fehlerkontrolle
  - NACK bei fehlerhafter Übertragung
  - Sender wiederholt auf Anforderung
- Probleme
  - wie merkt der Empfänger das ein Paket *fehlt*?
  - was passiert wenn NACK verloren?
  - Endlosschleife bei fehlerhafter Rückfrage



- Passive Fehlerkontrolle
  - Empfänger passiv
  - Zeitüberwachung: Timeout im Sender



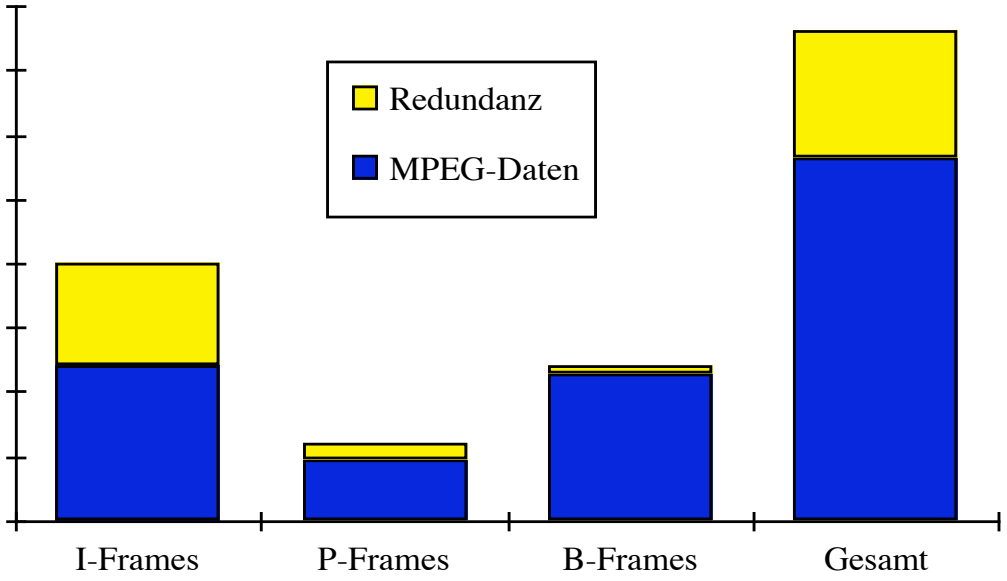
- Nachricht bei Ausbleiben der Bestätigung wiederholen



- Wiederholt auch verlorene Meldungen
- Round-trip-delay: Information-Bestätigung ...

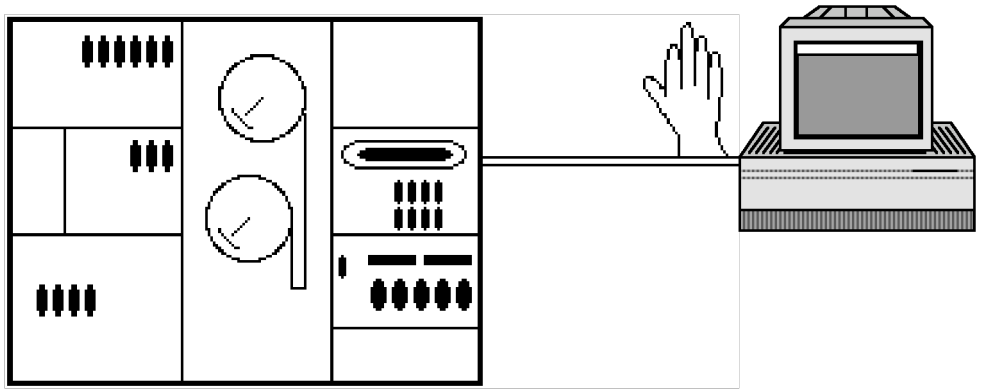
4.1.1.2 a-priori Fehlerbehebung

- Forward Error Correction
  - Nachrichten mehrmals senden
  - Nachrichten in verschiedenen Kodierungen senden
- Fehlerkorrekturinformation
  - Reed-Solomon-Codes, Faltungscodes (GSM)
- PET: Priority Encoded Transmission [Albanese, et.al]
  - selektive Redundanz für wichtige Komponenten



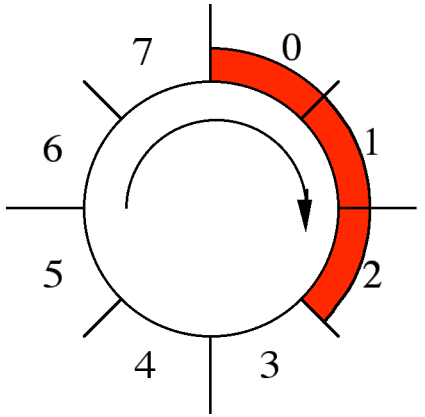
4.1.2 Kapazitäten managen

- Flußkontrolle
  - Anhalten des Datenflußes infolge Überlastung im Empfanger
  - keine Puffer frei
  - Anwenderprogramm verarbeitet zu langsam
  - Empfänger kann nicht schneller schreiben
  - Papier, Bildspeicher, Platte, zum nächsten Knoten …



- Explizite Flußkontrollbefehle (RR, RNR, X-ON, X-OFF, ...).
- Implizite Flußkontrolle
  - Anzahl ausstehender Bestätigungen begrenzt
  - Empfänger hält Bestätigung zurück
  - Bestätigung nicht zu lange zurückhalten (time-out)

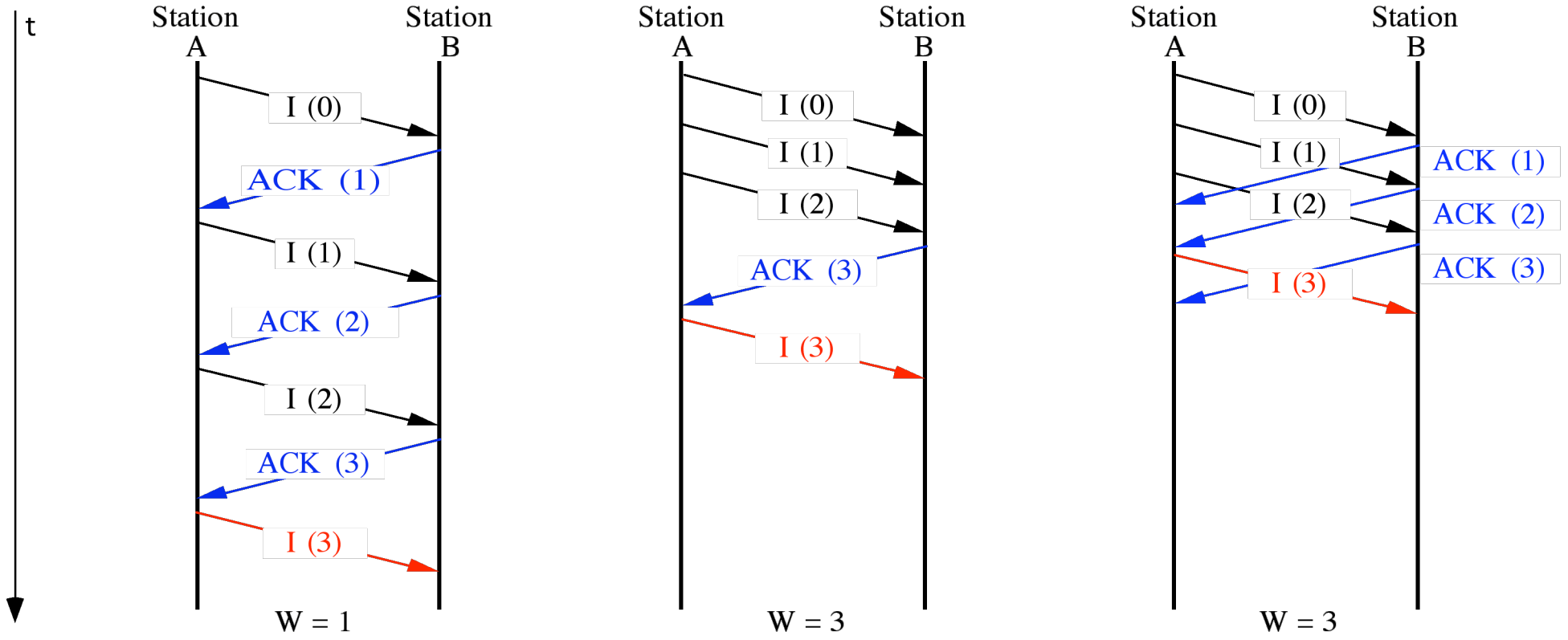
- Fenstertechnik
  - Übertragungsfenster = Anzahl unbestätigter Pakete
  - Fenster ausgeschöpft => nicht senden



Sequenznummern mod 8
Fenstergröße 3

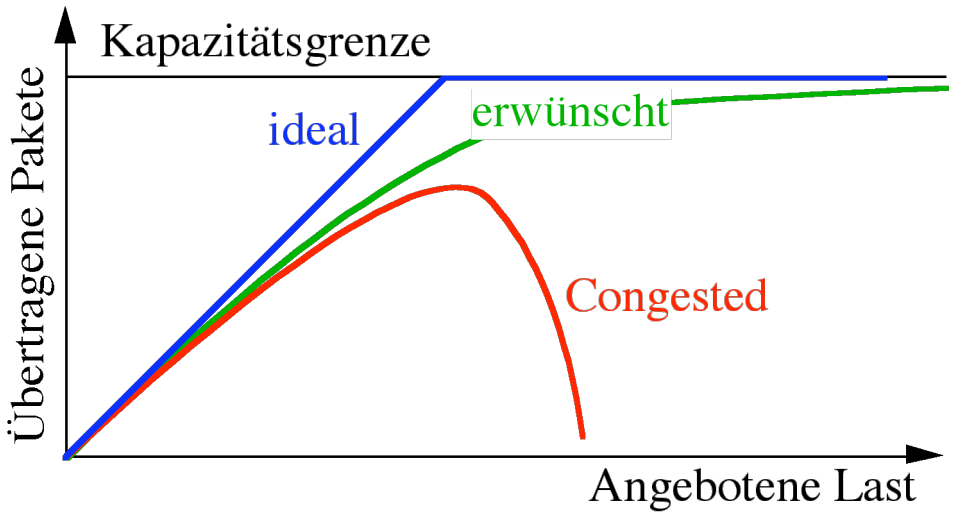
```
IF packetIn THEN
CASE inPacket.type OF
    Ack :  window:=window+1;
    NAck:  Retransmit(inPacket.expectedNumber);
    …
END {CASE};
IF packet[bufidx].ready AND (window>0) THEN BEGIN
    window:=window-1;
    SendPacket(packet[bufidx]);
    bufidx := (bufidx+1) MOD cXmitBufferSize
END {then};
```

- Beispiele für unterschiedliche Fenstergrößen
  - ACK enthält Sequenznummer des erwarteten Paketes/Bytes



  - weitersenden nach ACK bis Fenster erschöpft
  - oft vermischt mit Fehlerkontrolle
  - sofortiges Senden individueller ACKs verbessert Durchsatz
- W>1 reduziert round-trip-delay

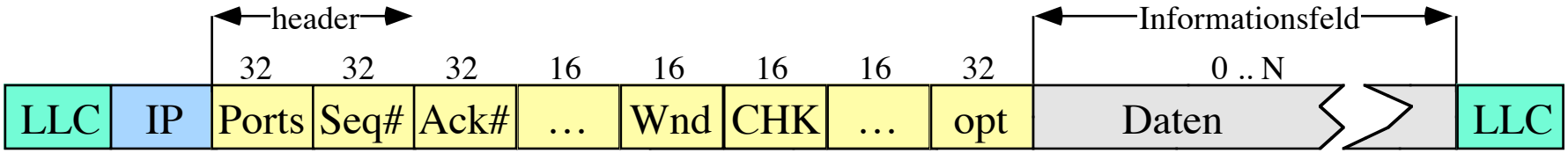
- Verstopfungskontrolle
  - typischer Durchsatzverlauf
  - Puffermangel im Router
  - Leitungsengpässe
- Unterschied zur Flusskontrolle
  - ein Phänomen im Netz
  - keine Verstopfung im Endknoten
  - Kommunikation Netz-Endgerät
- Netz verwirft SDUs
  - Reaktion des Endgerätes?
  - Beispiel IP und TCP



4.2 Beispiele

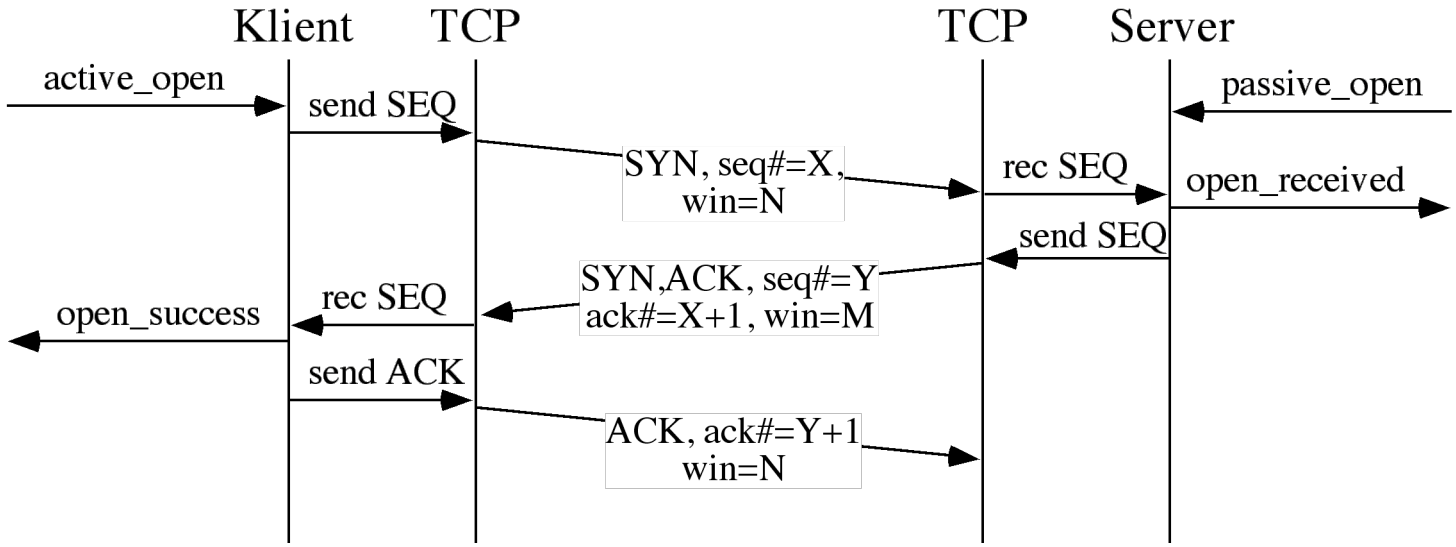
4.2.1 Transmission Control Protocol TCP

- RFC 793 und viele weitere
- Stromorientierter Dienst (Sockets)
  - Bytestrom ohne weitere Struktur
  - vollduplex, gepuffert
- Sockets als Benutzungsmodell (service access point)
  - Server: passive_open, open_received
  - Klienten: active_open -> open_success
  - send(daten) -> deliver
  - close -> closing -> terminate
- Adresse = IP-Nummer + Port
- Paketformat



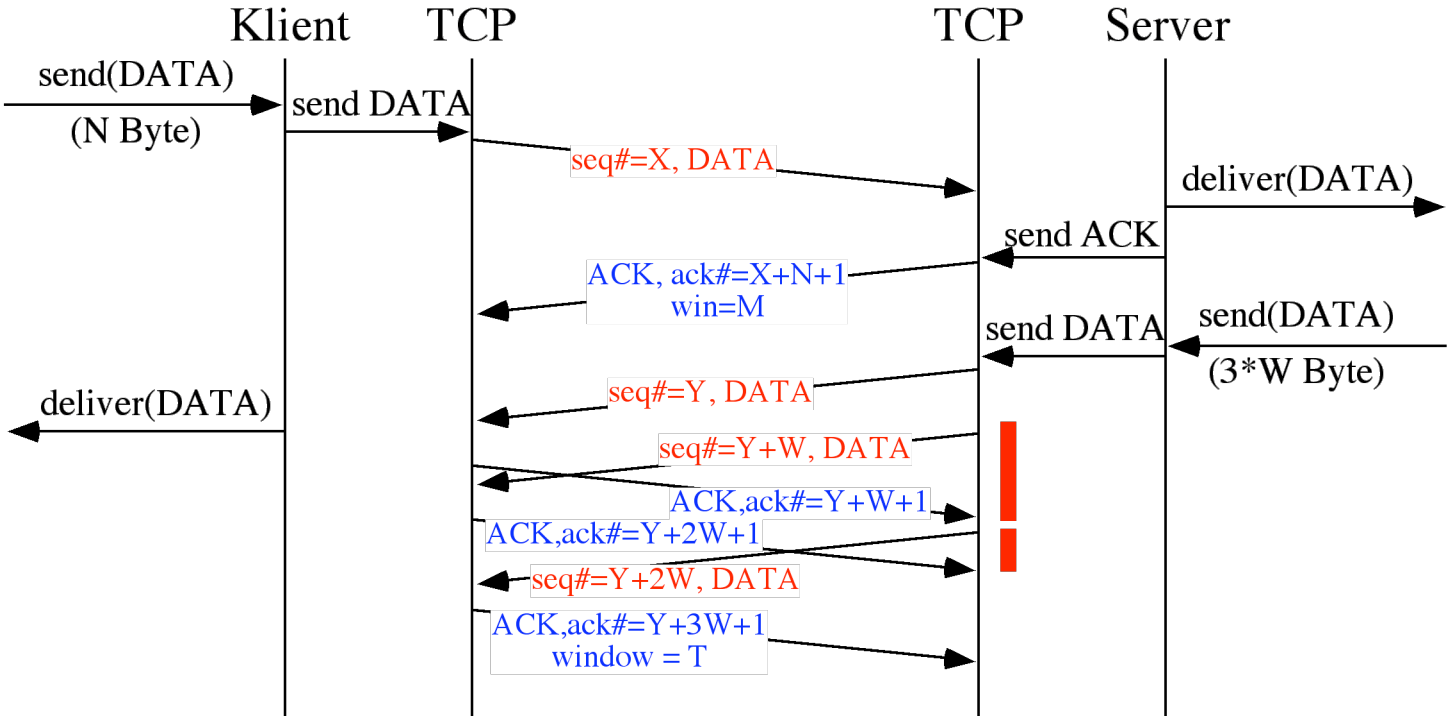
  - …: 6 Bit zur Signalisierung: URG, ACK, PuSH, ReSeT, SYN, FIN

- Verbindungsaufbau
  - Active Open: Verbindungsaufbauwunsch
  - Passive Open: ankommende Verbindung wird akzeptiert
  - three-way handshake
  - SYN, ACK sind Bits im Headerfeld Codebits

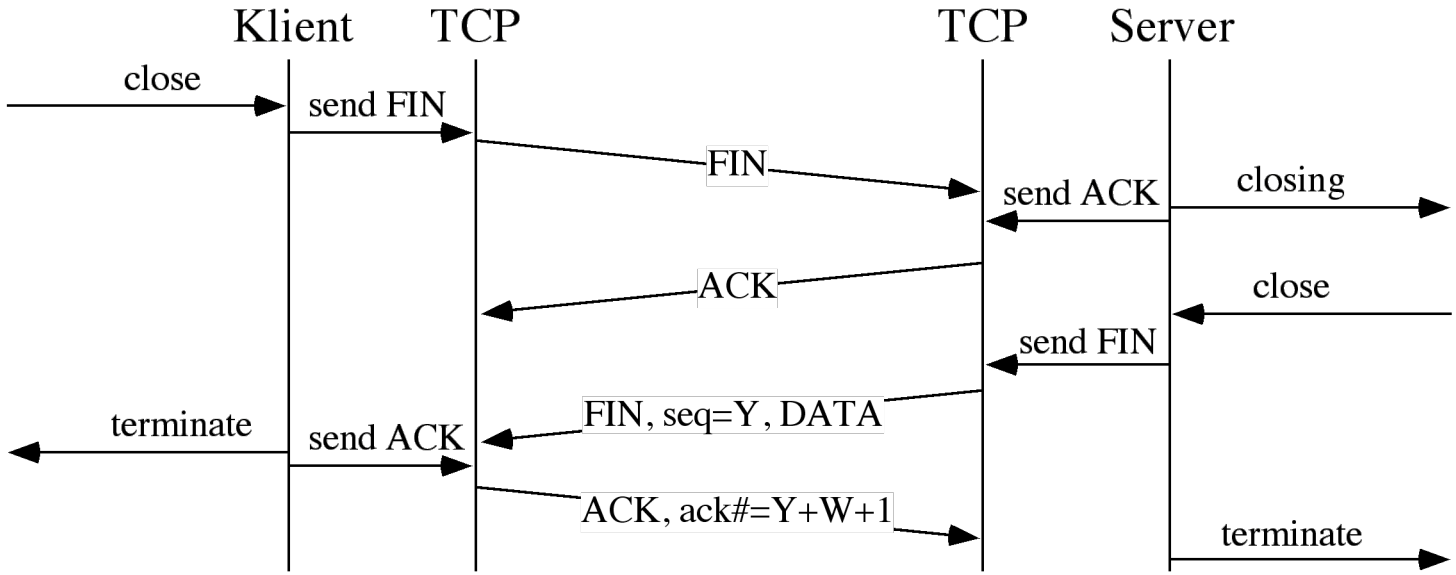


  - Sequenznummern werden synchronisiert
- Auch andere Verbindungsaufnahme möglich
  - active_open auf beiden Seiten
  - auch das erste Paket kann Daten enthalten

- Datenaustausch
  - sliding window, Byte-basiert
  - window advertisement im ACK
  - Sequenznummern 32 bit
  - adaptive Timeouts basierend auf RTT (round trip time)
  - window:= Min(advertisement, congestion_window)



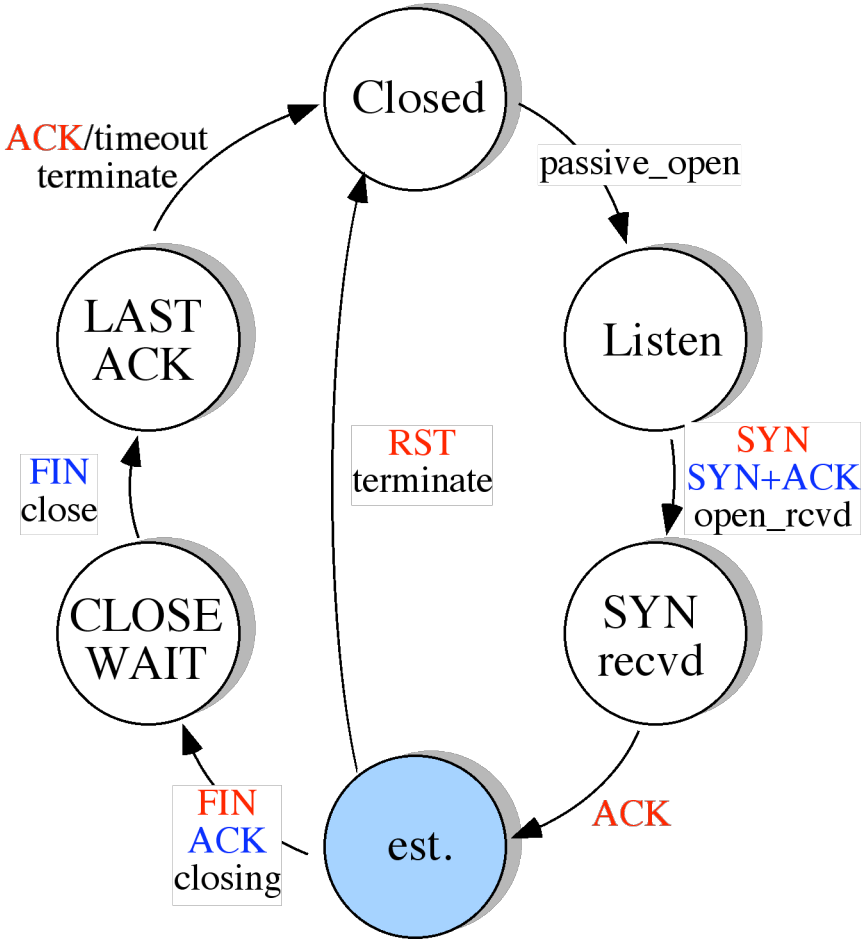
- Verbindungsabbau
  - Handshake zum Abbau
  - Daten können noch gesendet werden
  - warten auf FIN vom anderen Ende



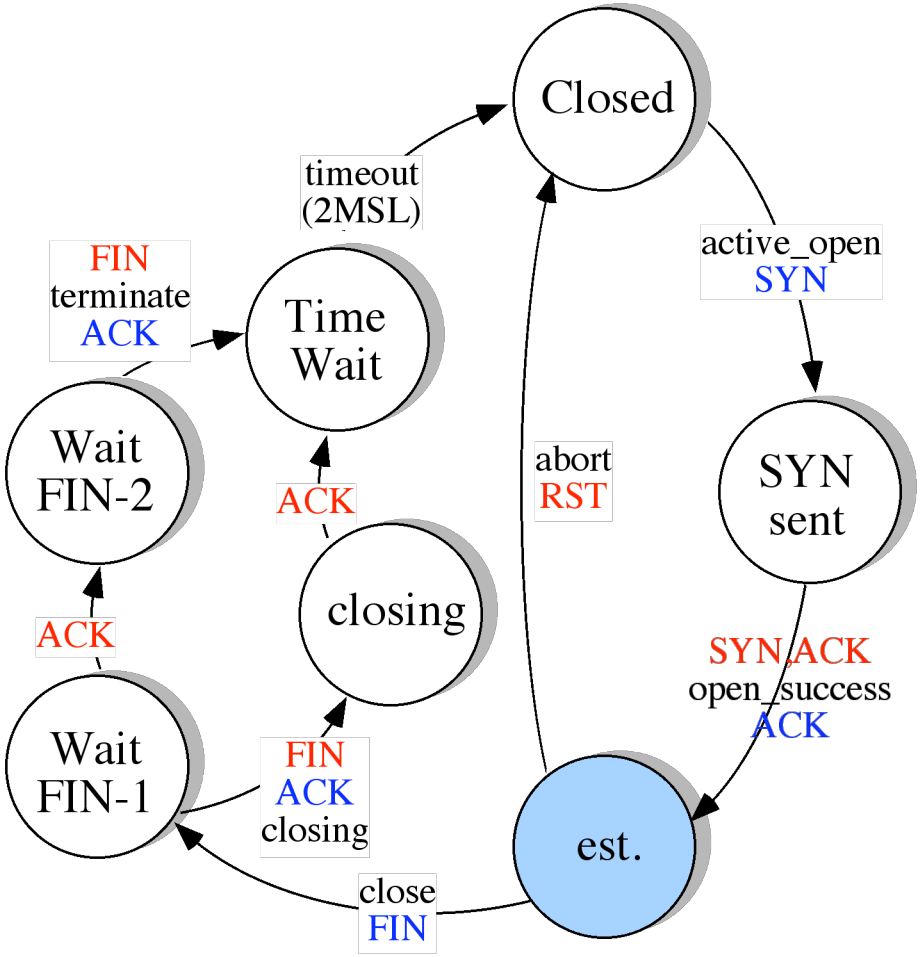
- FIN-Wait
  - Verbindungsende
  - A hat alles gesendet => schickt Paket mit FIN-flag
  - B antwortet mit ACK, hat aber noch Daten
  - B schickt Daten
  - B schickt FIN mit den letzten Daten

- Die TCP-State-Machine (siehe [W.R. Stevens: TCP …])

Server/passive close



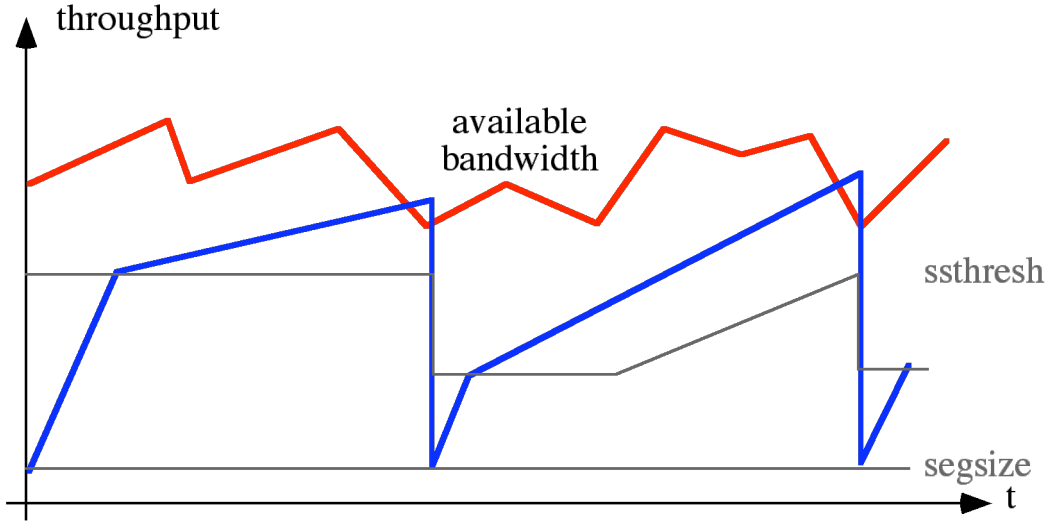
Klient/active close



rot: ankommendes Segment

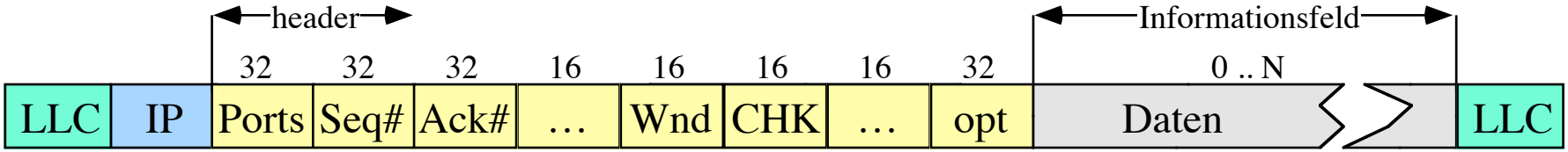
blau: abgehendes Segment

- Lastabwehr
  - Verlust eines IP-Datagrammes => Netzwerk überlastet
  - => Congestion-Window verkleinern
  - 'self-clocking': ACK=> inc(cwin)
  - slow_start [Jacobsen, 1988]
  - Timeout: cwin = seg_size;
    ssthresh = ssthresh/2;
  - Slowstart: Paket erfolreich
    übertragen:
    cwin = cwin+seg_size;
  - bis cwin>ssthresh
    => congestion avoidance (cwin=cwin+seg_size*seg_size/cwin)
  - am Übertragungsanfang zufällige Werte
  - Sonderbehandlung für 2*ACK und 3*ACK



- eine kurze Kritik von TCP
  - einfache Protokollmaschine
  - Prüfsumme am Anfang

  => erst Paket vollständig zusammenbauen
  dann Prüfsumme eintragen
  danach versenden

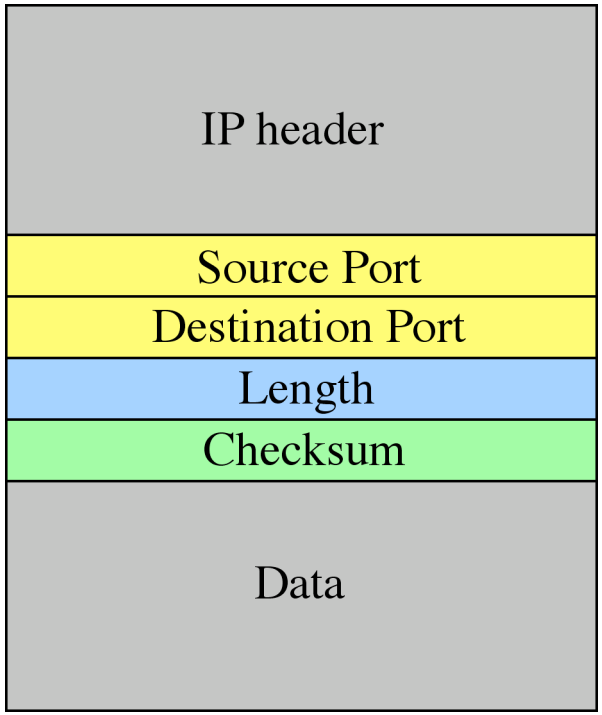


  - Trick: const = trueCHK + fudge
- Fehlerkorrektur nicht abschaltbar
  - Flußkontrolle und Fehlerkontrolle vermischt
  - Go-back-n Fehlerkorrektur
  - Selective acknowledgement (SACK) vorgeschlagen
  - bei jedem Paketempfang ACK(höchste# 'in-order')

- Verbesserung des Fehler- und Verstopfungsverhaltens
  - doppelte Acks
  - Tahoe: fast Retransmit
  - Reno: slow start
  - schnell wieder an ordentliche Fenstergröße herantasten
  - exponentiell bis ssthresh, dann linear
  - Vegas
  - trotzdem: Paketverlust ≠ Verstopfung
- Timed Wait Problem am Verbindungsende
  - A hat alles gesendet => schickt Paket mit FIN-flag
  - B antwortet mit ACK, hat aber noch Daten
  - B schickt Daten
  - A-Socket zu + neuer A-Socket offen => falsche Daten
  - => A muß Socket noch offenlassen
  - Mindestens 2 round-trips
  - Absturz an der anderen Seite?
  - => Server mit vielen, kurzen Transaktionen

4.2.2 User Datagramm Protocol

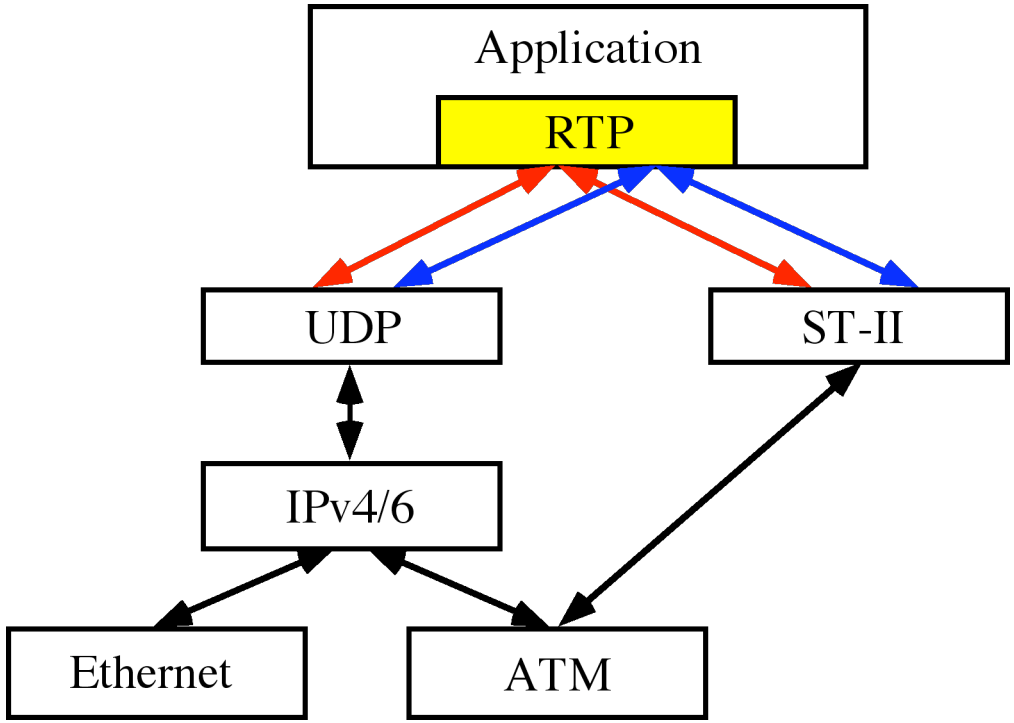
- Datagrammdienst für die Applikationen
  - IP für jeden
  - Ports kommen dazu
  - Prüfsumme (kann auch abgeschaltet werden)
  - Längenfeld
- Einfaches Paketformat



- Keine Fehlerbehandlung

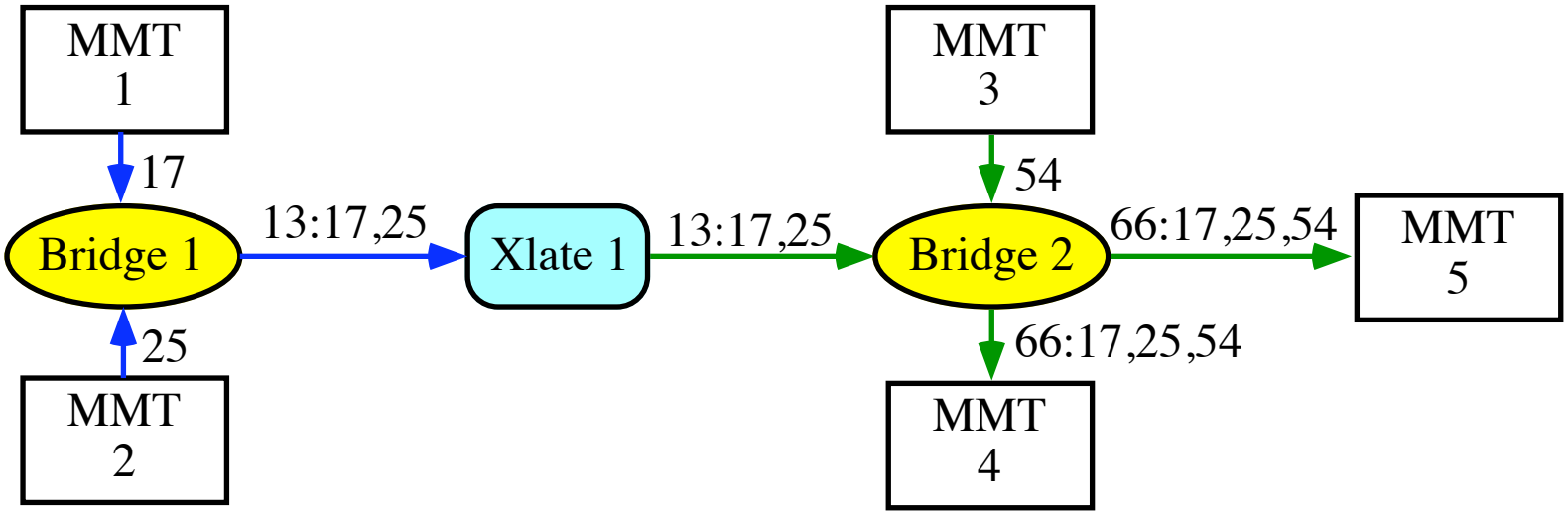
4.2.3 RTP: Real-Time Transport Protocol

- Schulzrinne, Casner, Jacobson
- Multicast-Unterstützung

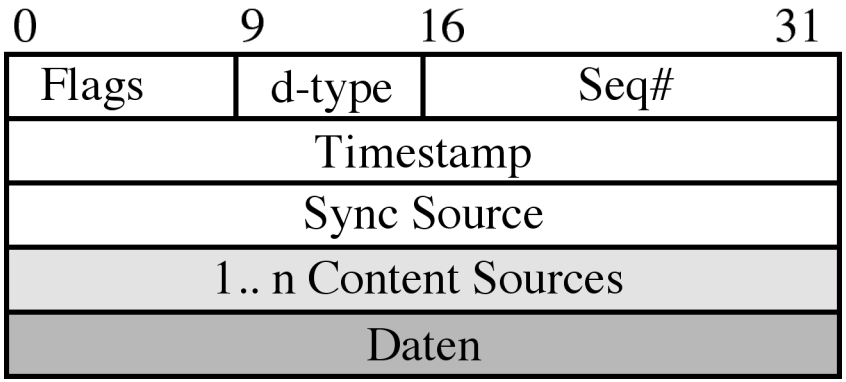


- RTP Control Protocol: RTCP
- RTP Data Transfer Protocol

- Bridge
  - Mixer



  - Translator
  - Synchronization Source, Content Source
- Einfaches Paketformat



- RTP Control Protocol: RTCP
  - Session-Kontrolle
  - Datenverteilung messen
- Problem (N)ACK-Implosion
  - Multicast
  - n Empfänger -> n Receiver-Reports
- Lösung: Reports auf Multicastverbindung
  - Verwaltungsverkehr < 5%
  - Empfänger überwachen Verwaltungsverkehr
  - Bestätigungen auswerten
- Skalierbarkeit der Session (Multicast): Gesamtdurchsatz begrenzen
  - Multicast von RTCP-Paketen
  - Sendezeit 'zufällig' wählen
- Mechanismen
  - Receiver Reports: Paketverluste, Jitter, timestamp
  - Sender Report: Session-Größe, Uhrsynchronisation
  - Source Description
- Receiver Report (RR)

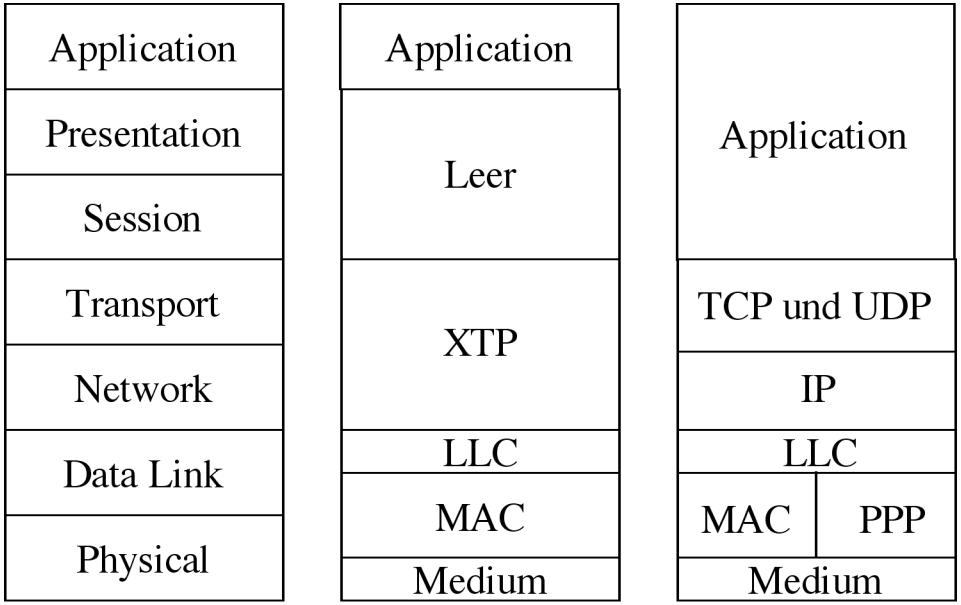
  - Gesamtzahl Pakete empfangen und erwartet
  - Jitter, Letzter SR
- Sender Report (SR)
  - Zeitstempel
  - Gesamtzahl Pakete und Bytes gesendet
- Source Description (SDES)
  - Source-ID
  - Canonical Endpoint Identifier: <User>@<DNS-name>
  - Name, E-Mail Adresse, Ort und Beschreibung (Prosa)
- Payload type mapping (FMT)
  - Source-ID
  - Typ 8 bit
  - Theoretischer Sample-Takt
  - IANA-ID (Internet Assigned Numbers Authority)
- Endpaket: BYE
  - Quelle beendet Sendung

4.2.4 eXpress Transfer Protocol: XTP

(Literatur: T. Strayer, B. Dempsey, A. Weaver: XTP)

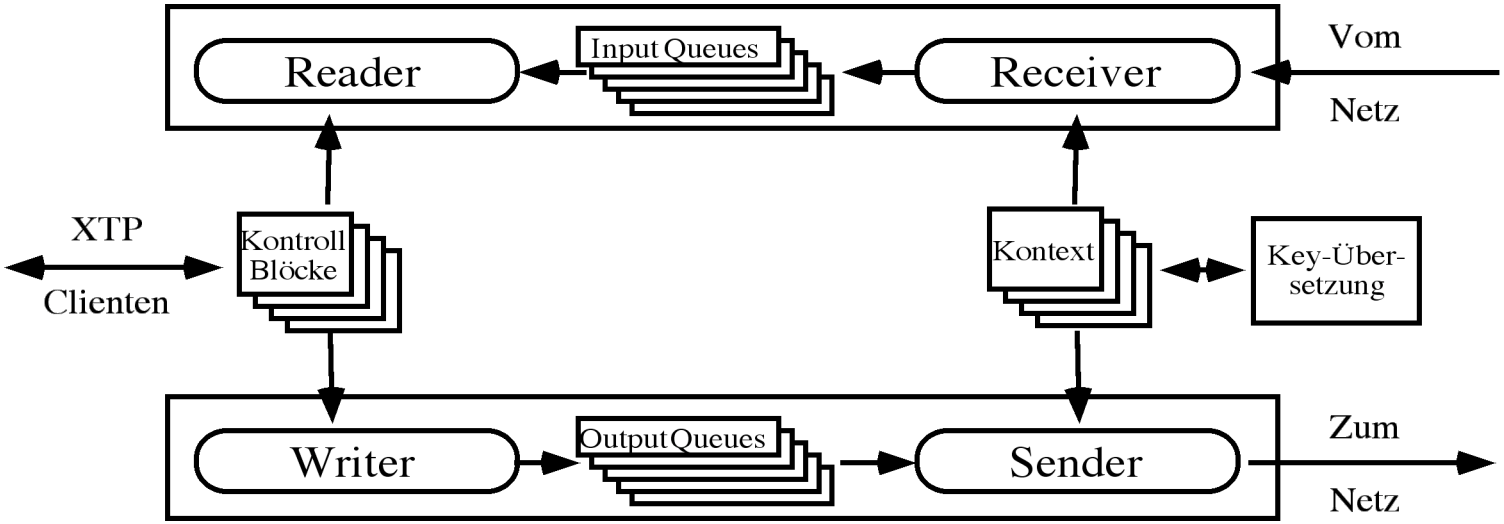
3.2.4.1 Ideen

• Transfer Layer



- Transport-Schicht + Netzwerk-Schicht
- Gründe: Leistungssteigerung
  Integriertes Puffermanagement und Lastabwehr
  Verminderte Duplizierung von Mechanismen (siehe X.25!)

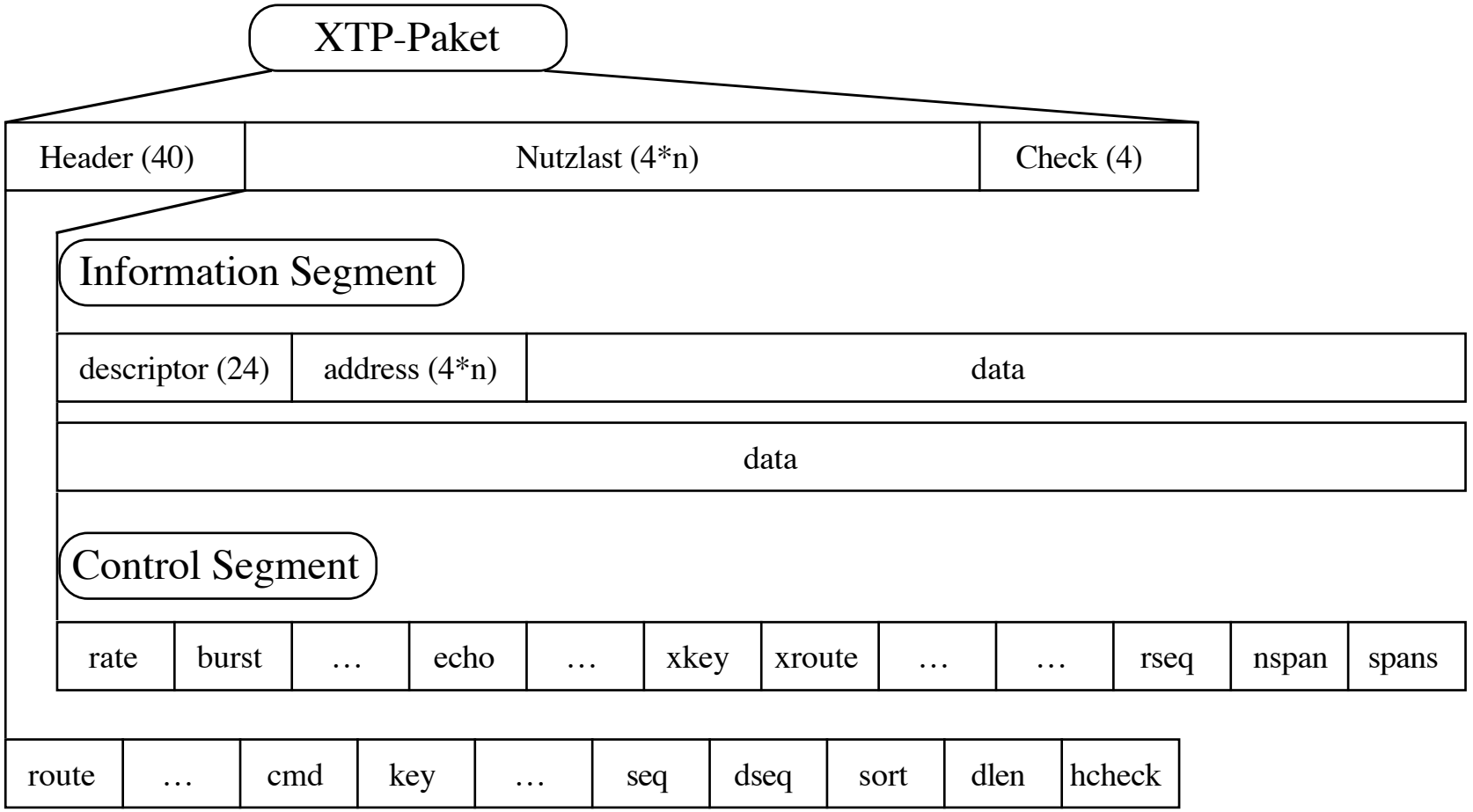
- Mechanisms vs. Policy
  - flexible Prozeduren
  - Verwendung bestimmt Protokoll
- Architektur



- Assoziationen
  - Sammlung von Statusinformationen für aktiven Datenaustausch
  - Im Sender und im Empfänger
  - In Zwischenknoten
  - Kontext-Record
- XTP - Vermittlungen (Router)
- Multicast
  - ungesichert (blast)
  - semi-zuverlässig
    - Synchronizing Handshake mit StatusRequest und -Report
    - Verschiedene Heuristiken zur Fehlerkontrolle
- Hardwareimplementierung als Entwurfsziel
- Prozeduren
  - Assoziations- und Pfadmanagement
  - Datenübertragung
  - Raten- und Flußkontrolle
  - Fehlerkontrolle

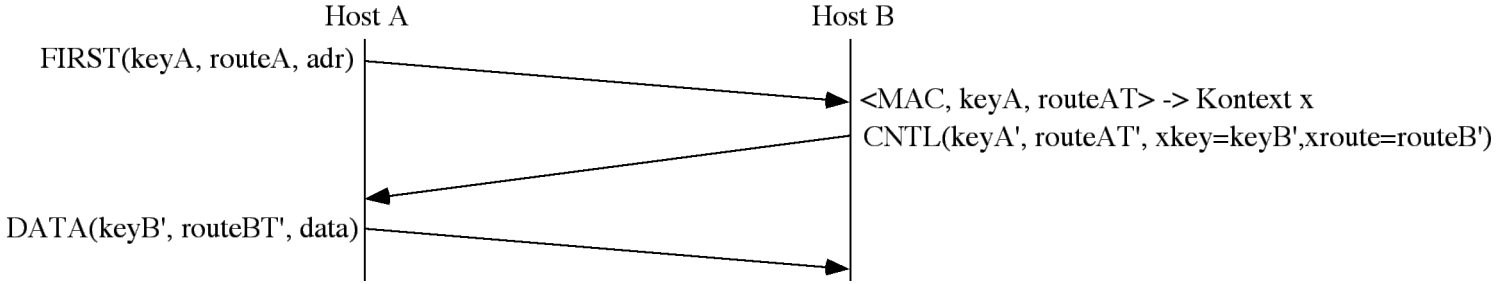
3.2.4.2 Paketformat

- Langwortaligniert, es gibt aber Bitfelder



3.2.4.3 Verbindungsaufbau

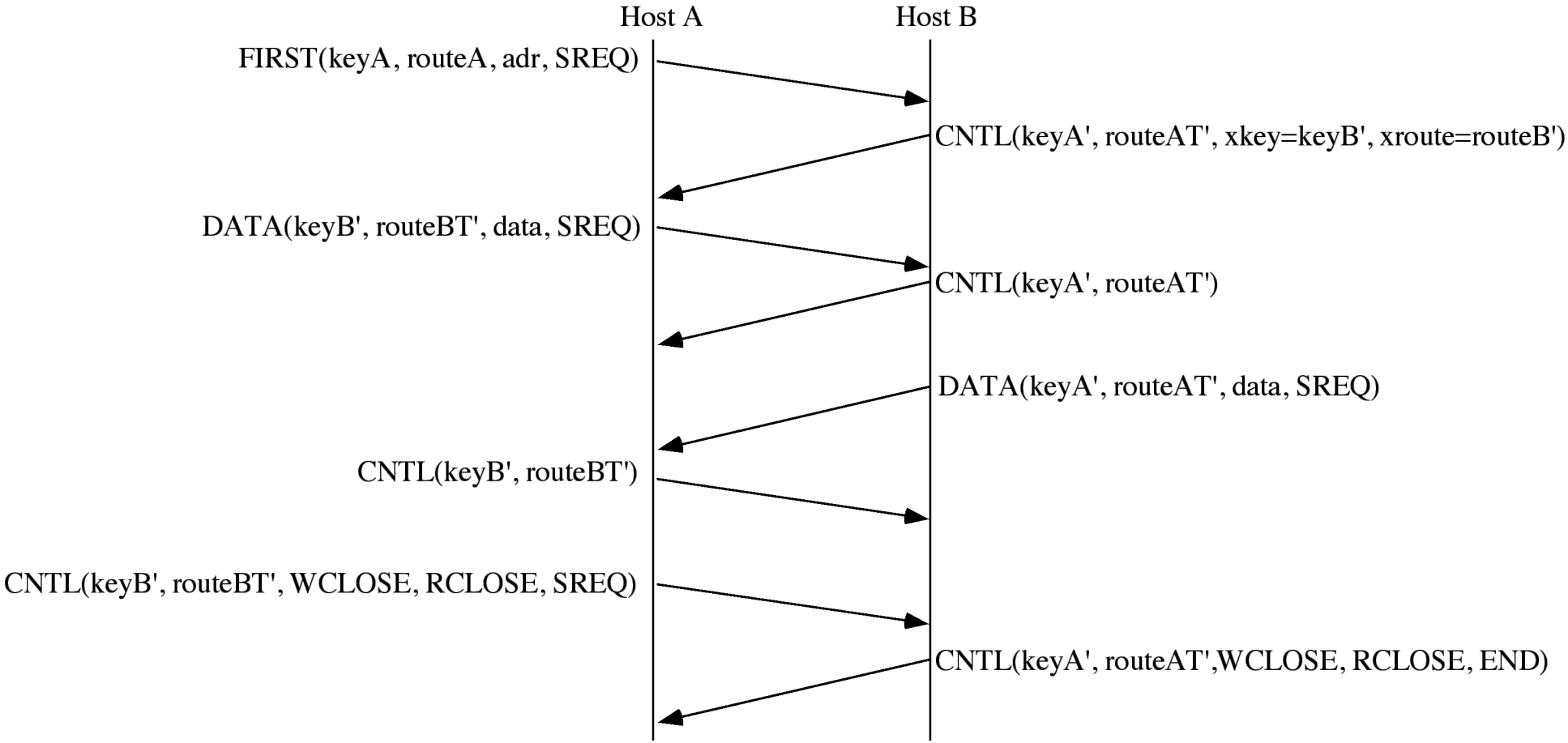
- FIRST-Paket
  - enthält Zieladresse
  - route und key Felder wichtig
  - rate-Feld
- In den Vermittlungsknoten
  - Pfad durch das Netz finden
  - Tabelle mit Tupeln (inroute, Eingangsport, Ausgangsport, outroute)
- Beim Emfänger
  - Muß Kontext im '*listen*' Zustand haben
  - Abbildung des Tupels (key, route, MAC-Adresse) auf Kontext
  - Rückwärtspakete mit route+MSB und key+MSB (= returnkey)
  - key-Austausch zur einfacheren Abbildung auf Kontext



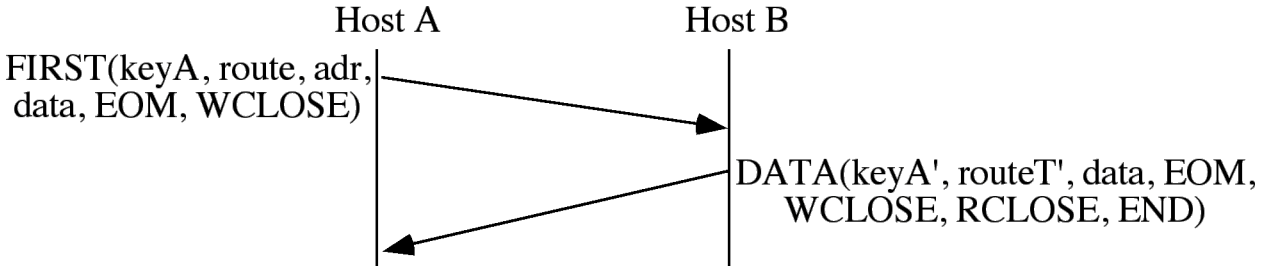
3.2.4.4 Datenübertragung

- In DATA und FIRST-Paketen
- Flußkontrolle mit Krediten
- Fehlerkontrolle
  - Bytesequenznummern
    geben Adresse im Empfangspuffer an
    ein langer Puffer genügt
  - Sender fordert Bestätigung an
    SREQ- oder DREQ-Bit (DREQ nach 'delivery' beim Klienten)
    Empfänger antwortet mit CNTL(rseq) oder CNTL(rseq,span(s))
    rseq allein ermöglicht go-back-n
    span(s) ermöglichen selective retransmission
  - Beispiel (Daten 1..34 bereits gesendet):
    -> DATA(seq=35, len =5, data=(35..39), SREQ)
    <- CNTL(rseq=15, nspan=2, span1=(20,25), span2=(30,35))
    -> DATA(seq=15, len=5, data=(15..19))
    -> DATA(seq=25, len=5, data=(25..29))
    -> DATA(seq=35, len=5, data=(35..39))

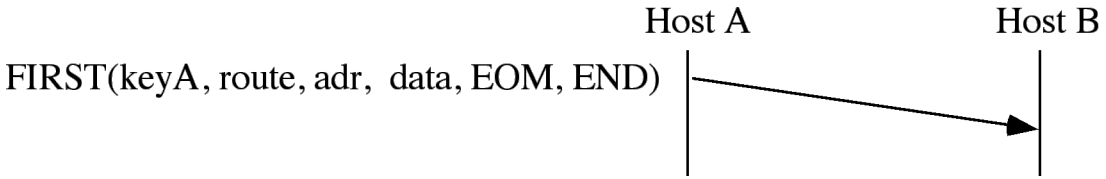
- Verschiedene Verbindungs-Paradigmen auf XTP abbilden
  - verbindungsorientiert



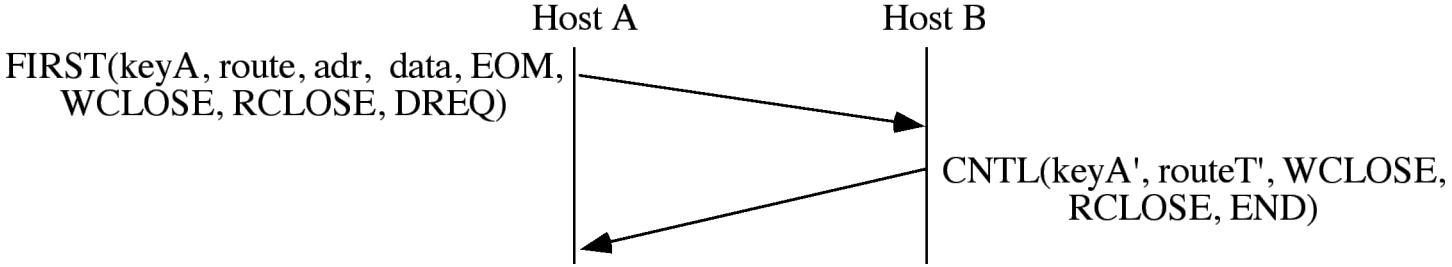
- Transaktion



- Datagramm

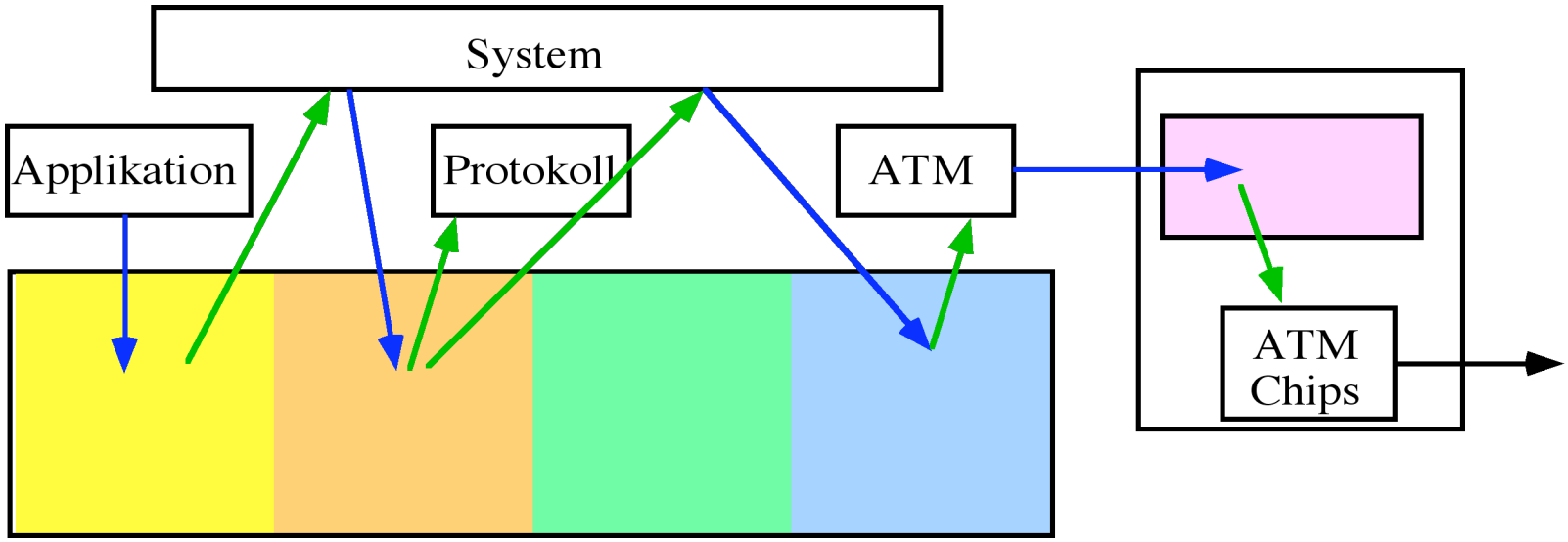


- zuverlässiges Datagramm



4.3 The demise of protocols

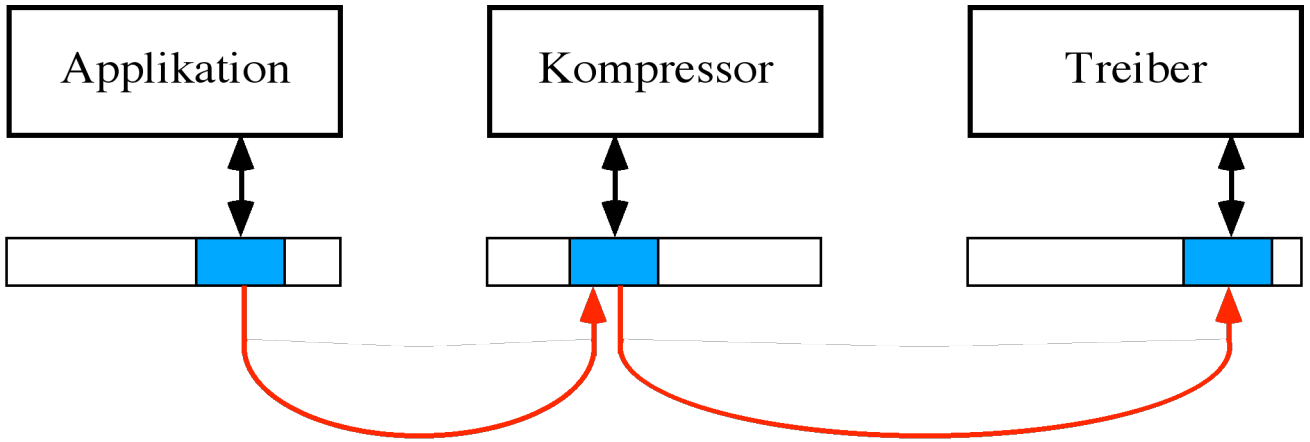
- Schichten bremsen den Datenfluß



- Ausführungsfluß im Schichtenmodell
  - Applikation
  - Systemdienst: Socket-Layer
  - Protokoll
  - Systemdienst: BlockCopy/Gather
  - ATM/AAL (Segmentation)
  - System: Scheduler

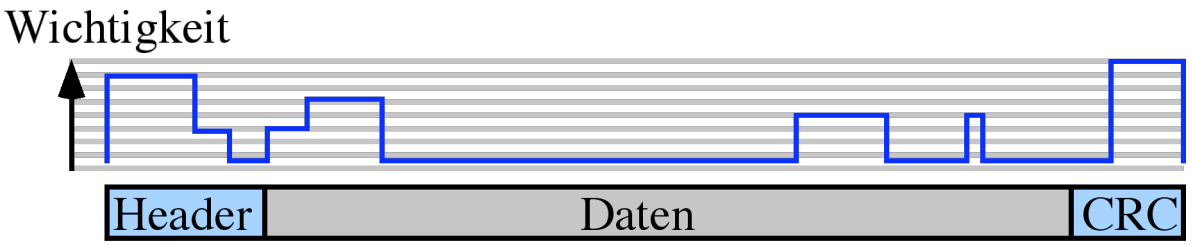
- Warum?
  - Kontextwechsel (Taskswitch) kostet Zeit, Cache ungültig
  - Adressräume müssen umgeschaltet werden

  => Datenkopieren oder shared memory
- Datenkopien vermeiden
  - CPU-Last, Bus-Last
  - Cache-Effizienz



- Beispiel Standard C - Ein/Ausgabe
  - getc, putc jeweils mit einem Zeichen

- Integrated Layer Processing
  - Beschleunigung von Protokoll-Stacks
  - kein Kontextwechsel
  - eventuell Threads zur Abstraktion
  - nur ein Adressrum
  - keine Duplizierung von Funktionen
- Dynamische Dienstegüte
  - Quellekodierer kennt Datenstruktur
  - Beispiel Fehlerkorrektur:

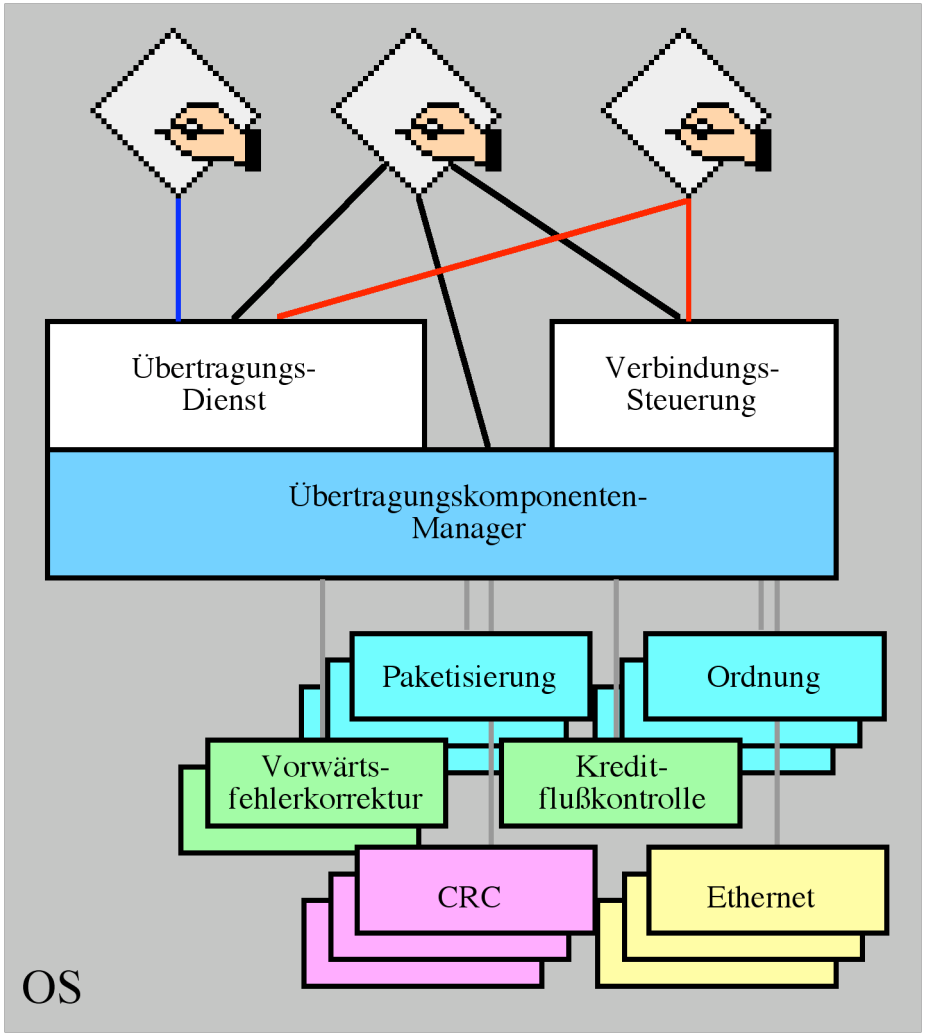


  - Zusammenarbeit Quellkodierer, Protokoll und Kanalkodierer
- Application Layer Framing
  - Abbildung (Anforderungen -> QoS -> Netzwerk) komplex
  - Dienstegüteverhandlung
  - flexible Fehlerkontrolle

- Optical Switching und WDM
  - End-to-End Verbindungen
  - superschnell
  - skalierbar mit schmalen Bändern (präzisen Lasern)
  - hochgradig zuverlässig

  => viele Protokoll-Funktionen werden nicht mehr gebraucht
- Baukastenprinzip
  - mehrere Abstraktionsebenen
  - Mechanismenklassen
- Klassen
  - Protokollfunktionen
  - Fehlerbehandlung
  - Flußkontrolle
  - Lastkontrolle
  - Traffic-Shaping

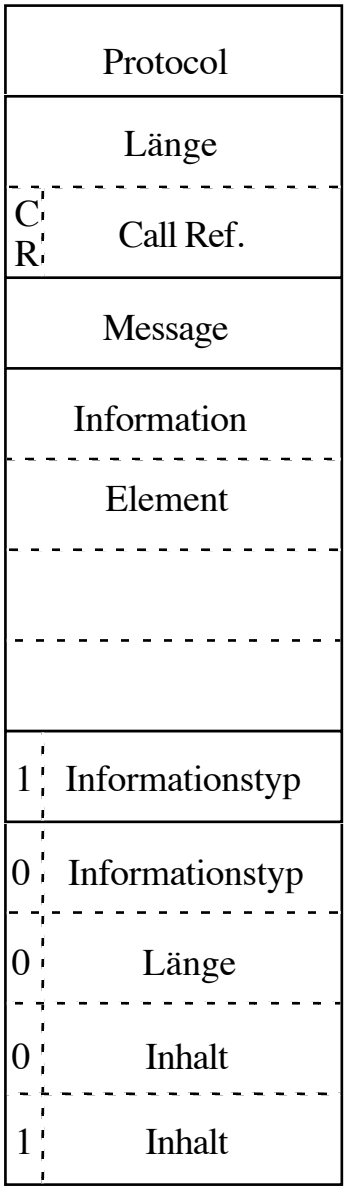
  => konfigurierbare Dienste



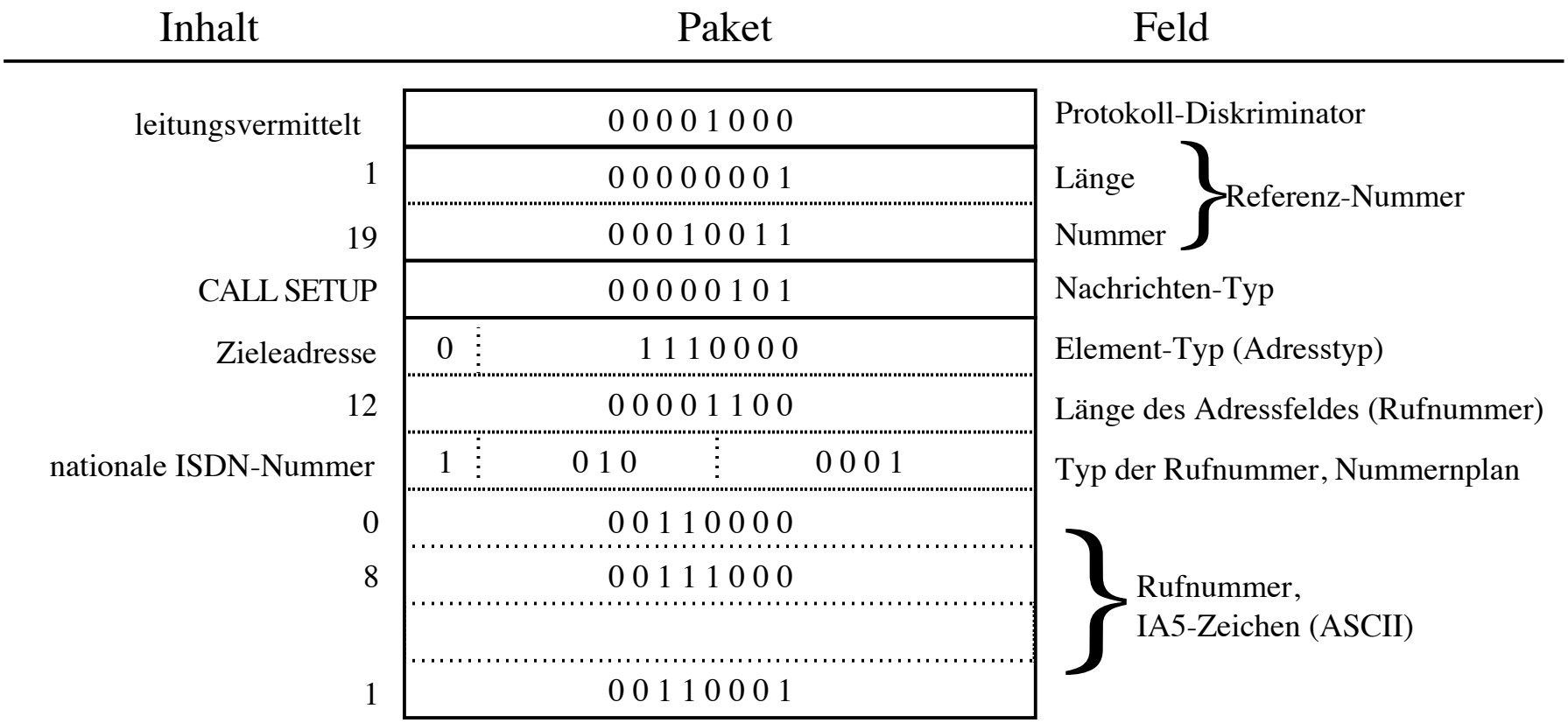
5 Steuerungsdienste

5.1 ISDN-Signalisierung: Q.931

- Paketformat
  - Protocol Discriminator
    Q.931 (5E, NT): 8;
    1TR6: 64, 65; …
  - Connection Reference
  - Nachricht
    Setup, Setup-Ack, Call Sent,
    Alert, Connect,
    Disconnect, Disconn-Ack
  - Informations-Elemente
    Länge 1: Codesatzumschaltung
    1 < Länge < 256:
    Called Number,
    Calling Number, …
    Cause,
    Bearer Capability,
    Service Indicator …

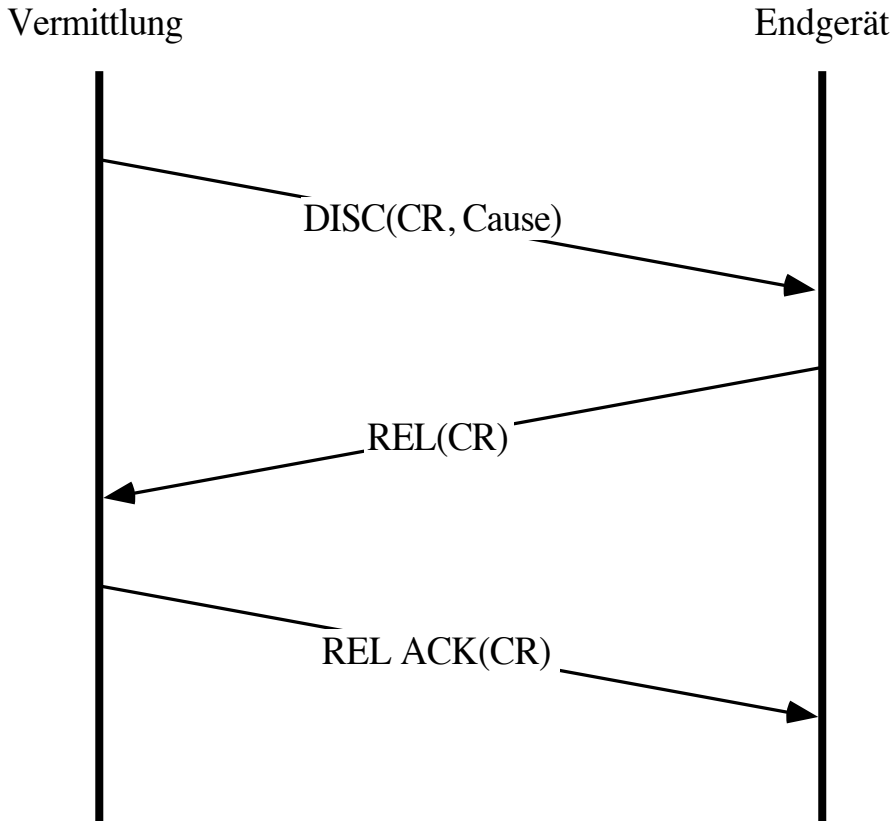


- Beispielpaket
  - Verbindungsaufbau
  - Telefonnummer
  - evtl. weitere Beschreibung: bearer capabilities

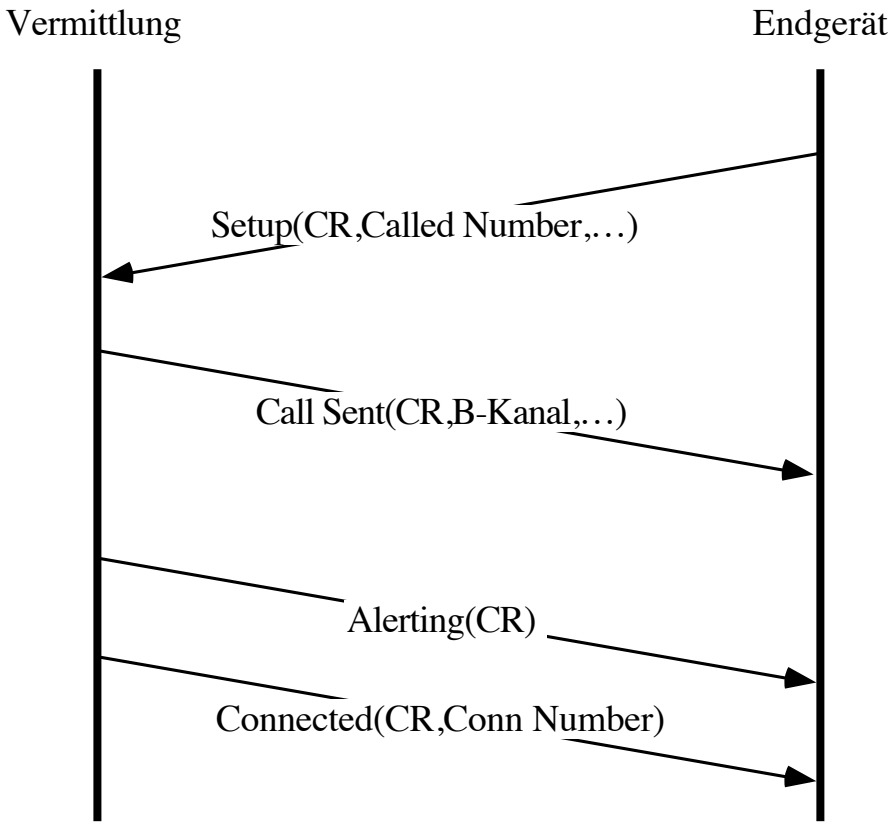


- Prozeduren
  - Command / Response für Ebene 3
  - implizite Festlegung der Anwendungs-Schicht
- Basic Calling
  - Verbindungsaufbau, -abbau
  - Status, Information, Progress
  - Pakete: Setup, Disc, Alert, …
- Verbindungsbezogene Leistungsmerkmale
  - Facility Paket
  - huckepack in anderen Paketen
  - entsprechende Connection reference
- Verbindungslose LM (anschlussbezogen)
  - Facility Register, -Status, -Indication,
  - besondere Connection Reference
- Informationselemente für LM
  - Functional: Facility Information Element
  - Stimulus: Feature Activator/Indicator

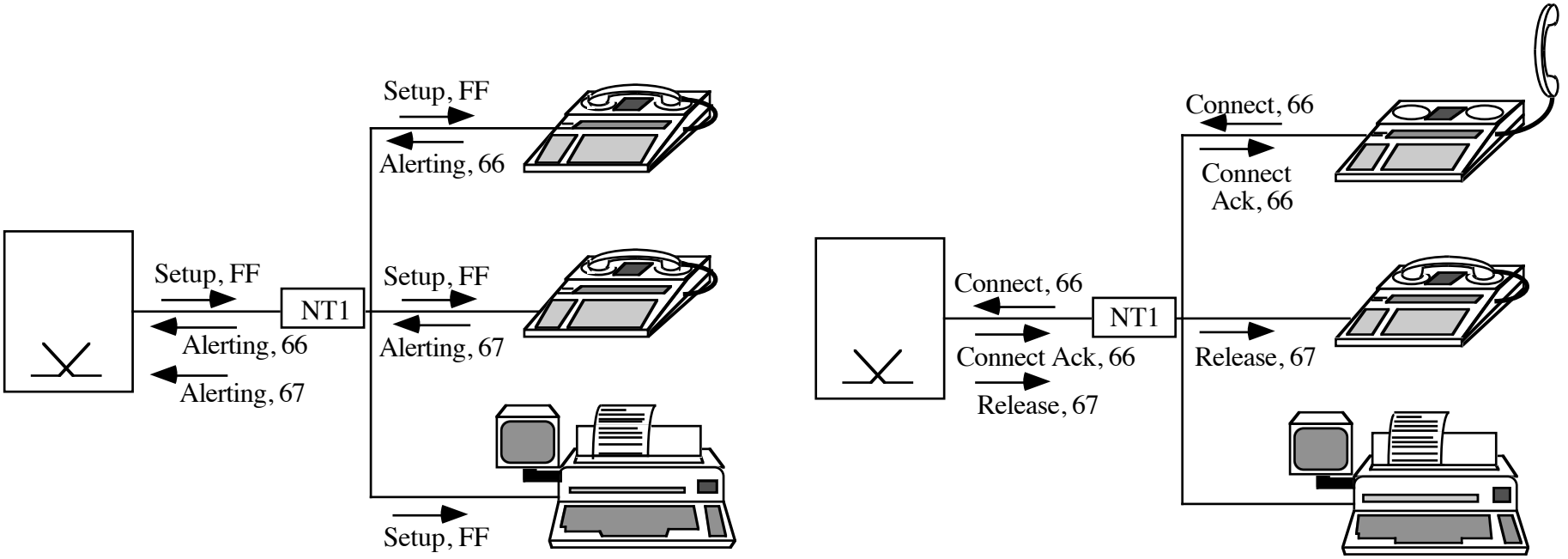
- Funktionale Signalisierung
  - Remote Procedure Call
  - Prozedur (Facility(X)):
    - → Leistungsmerkmal X ausführen
    - ← Leistungsmerkmal X (nicht)ausgeführt
- 1TR6; DKZ-N1; NT BCS 29; 5E6 (; VN2); Euro-ISDN
- Einfaches Beispiel:
  - Verbindungsabbau durch Netz



- Mittleres Beispiel:
  - Verbindungsaufbau durch Endgerät

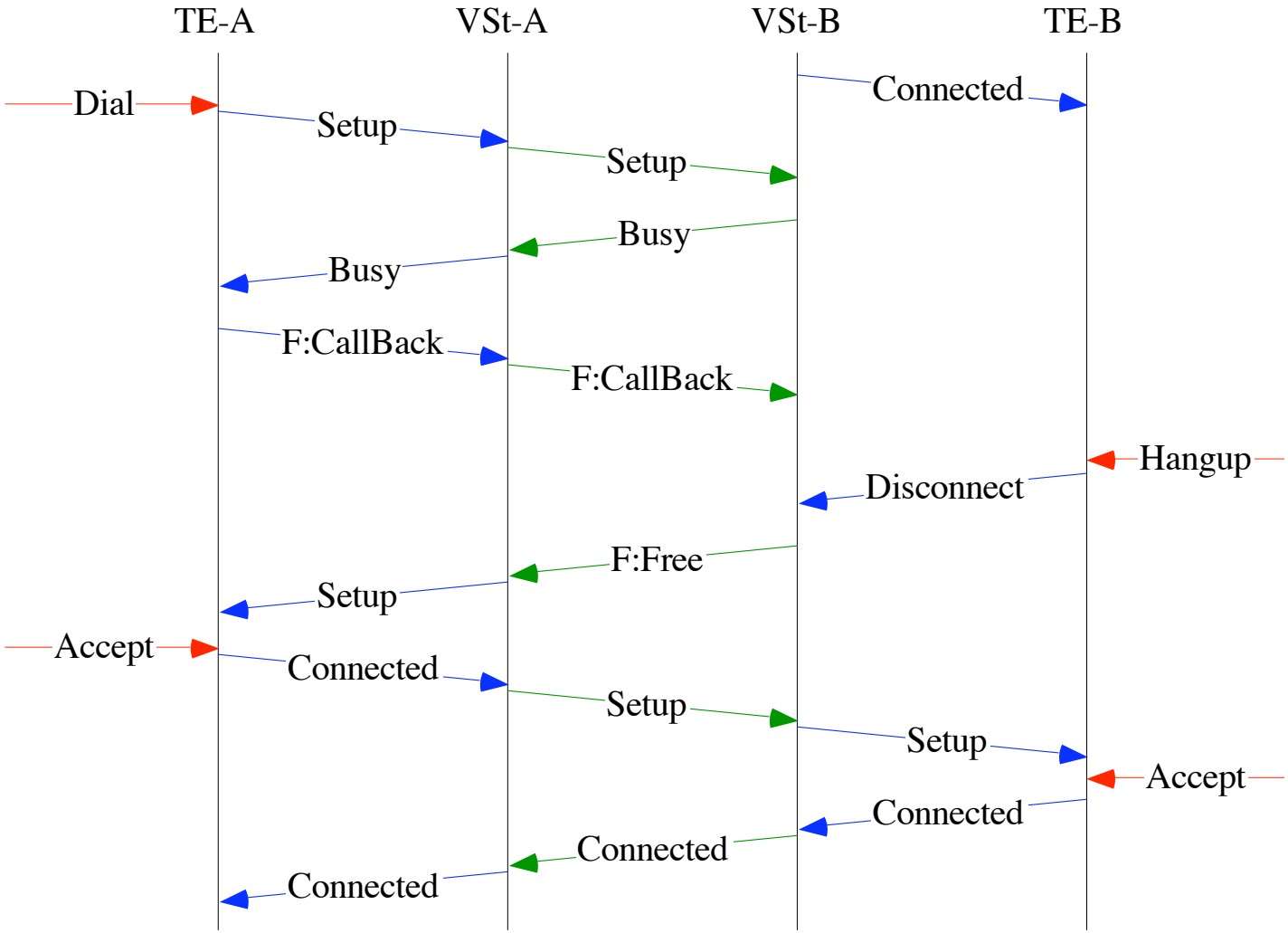


- Komplexes Beispiel: Verbindungsaufbau vom Netz
  - mehrere Geräte am Bus
  - Anbieten der 'kommenden Belegung'
  - Prüfen des Dienstes in den Endgeräten

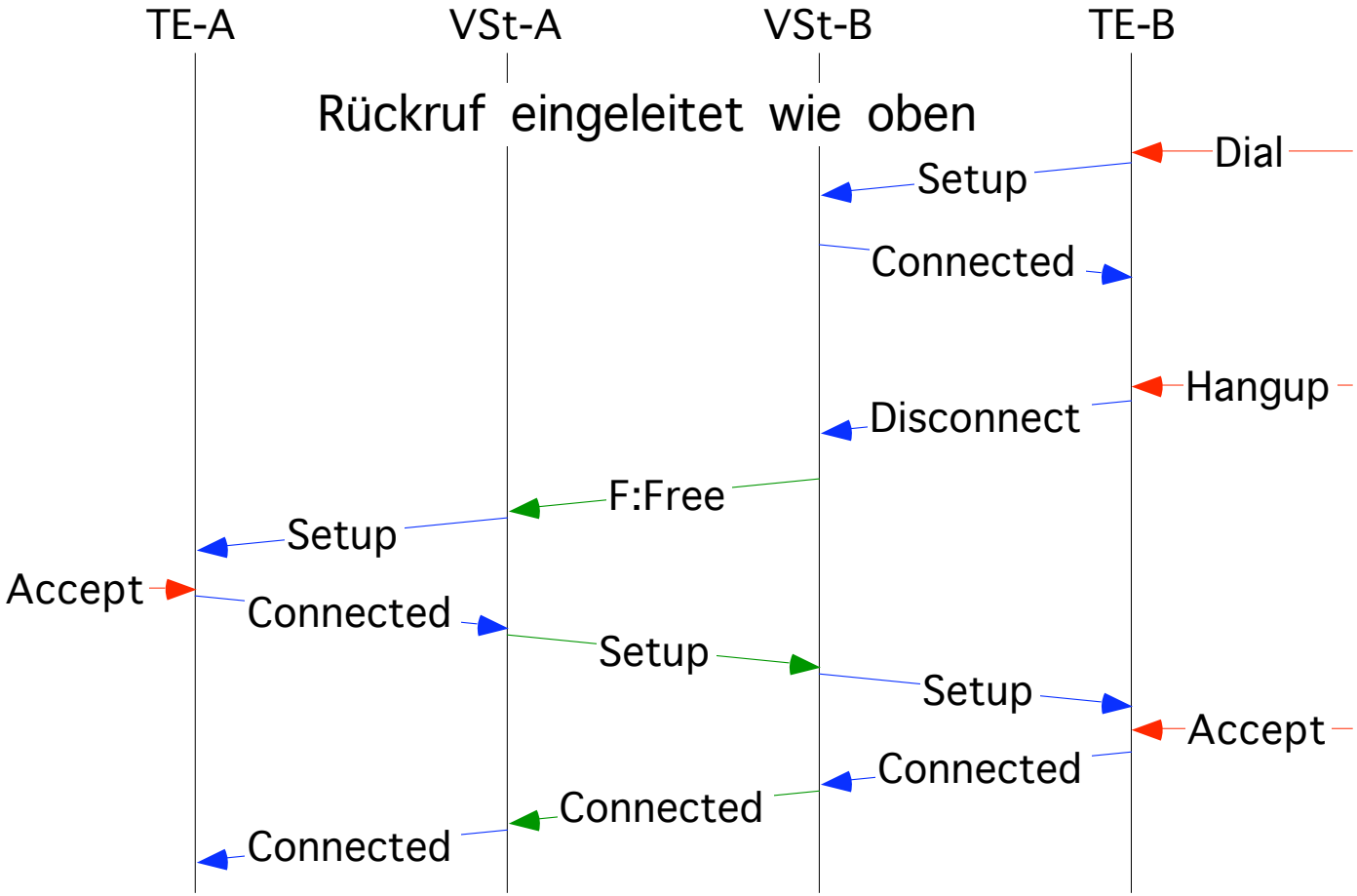


- Annahme des Rufes
- Rücknahme des Angebotes

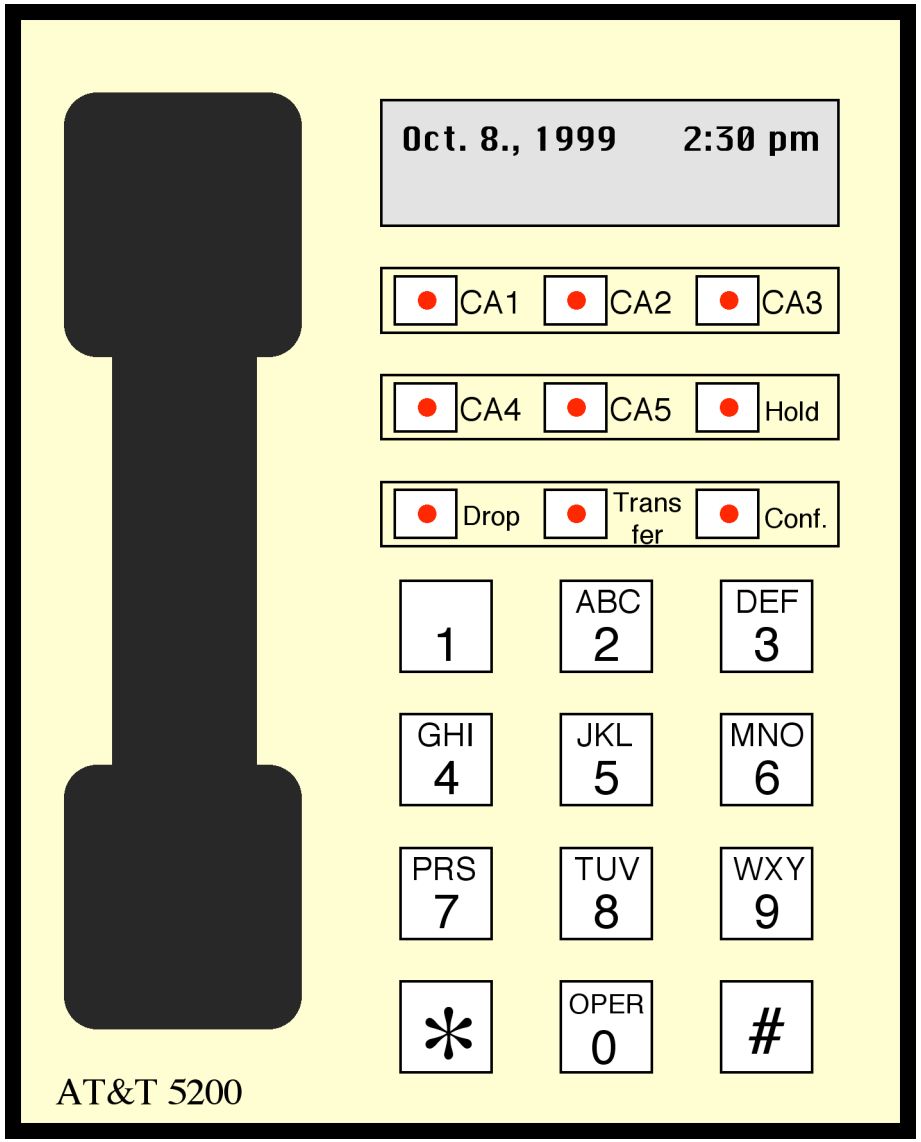
- Feature Rückruf bei besetzt



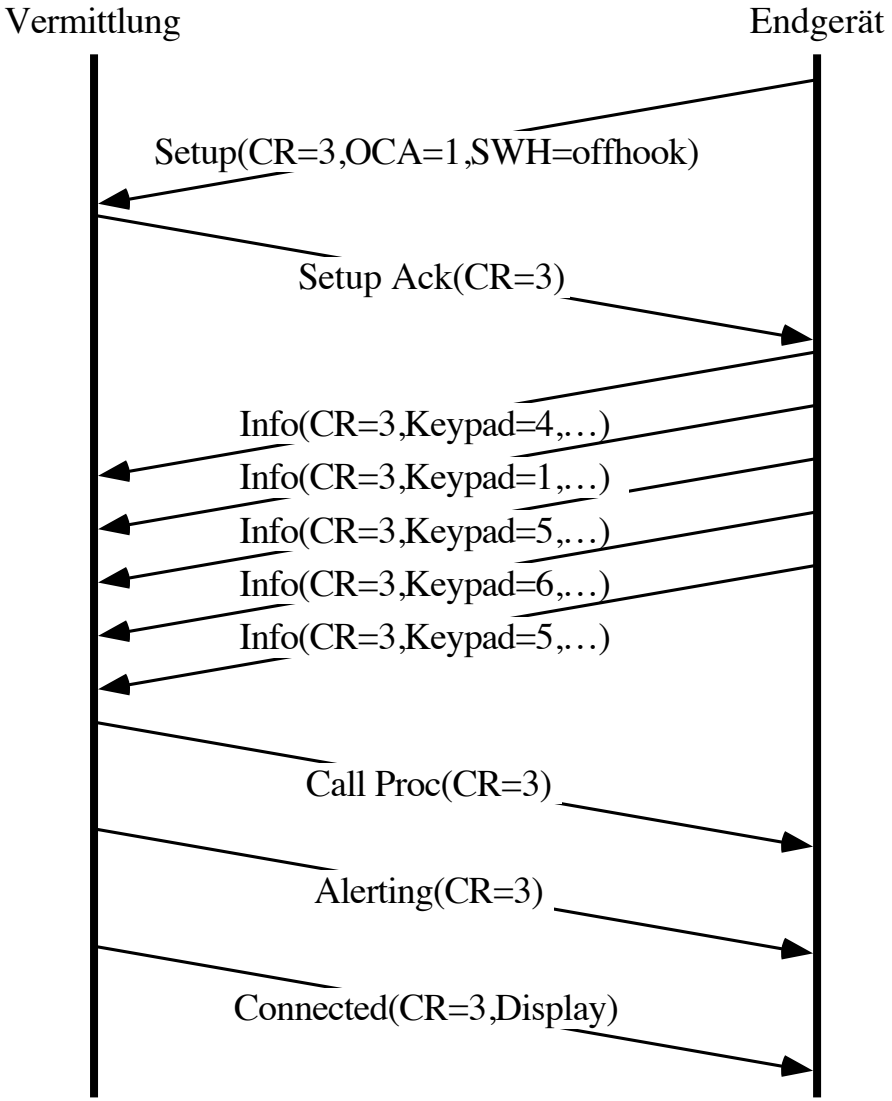
- Feature: Rückruf bei frei



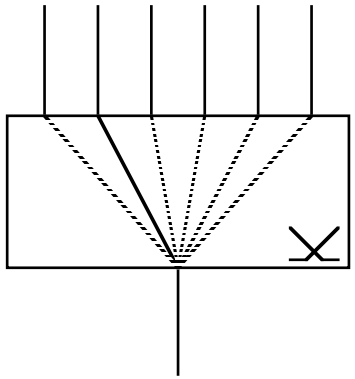
- Stimulus Signalisierung
  - Terminalbetrieb
  - Digital Centrex
  - AT&T, NT
- Prozedur für LM
  - INFO-Pakete oder huckepack.
    -> Knopf gedrückt (Info FA=Button 12)
    <- Lampe einschalten (Info FI=LED 12)
- Terminal Management
  - Call Appearance
    BNx := CAy aktivieren
    LEDx := CAy aktiv
  - Feature Activator und Feature Indicator
    FAz := Transfer
    FIz := Transfer aktiv
- Bsp: Telefon mit 9 Knöpfen
  5 Verbindungen (CAs)
  4 Features

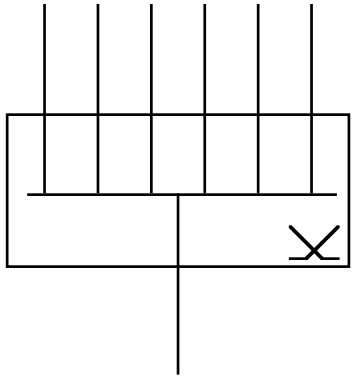


- z.B. Verbindungsaufbau:

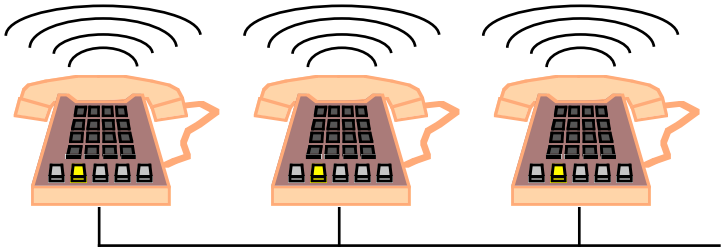


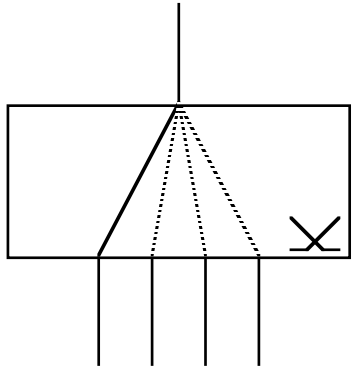
- Verbindungsbezogene LM
- Verbindungsaufbauphase
  - Rückruf
  - Ring Again
  - Anruferindentifizierung (ICLID)
- Asynchronous Multipoint ("Dreierverbindung")
  - Flexible Call Offering:
    Anklopfen (call waiting)
    bevorrechtigte Anrufe (Chefruf)
  - Halten und Wiederaufnehmen
    Rückfrage
    Makeln
  - Transfer
- Synchronous Multipoint
  - Konferenz
  - LAN





- Key-System
  - Call Pickup (Rufübernahme)
  - Call Park
  - Bridging (Aufschalten)
  - Rufübergabe (Transfer)
  - Sammelanschluß (hunt group)
- Verbindungslose LM
  - Ruhe
  - Make Set Busy
  - Rufumleitung (immer, fallweise, …)
  - Information (Gebührenanzeige, Zeit)
  - Nachrichten
- Abfrageplatzfunktionen …

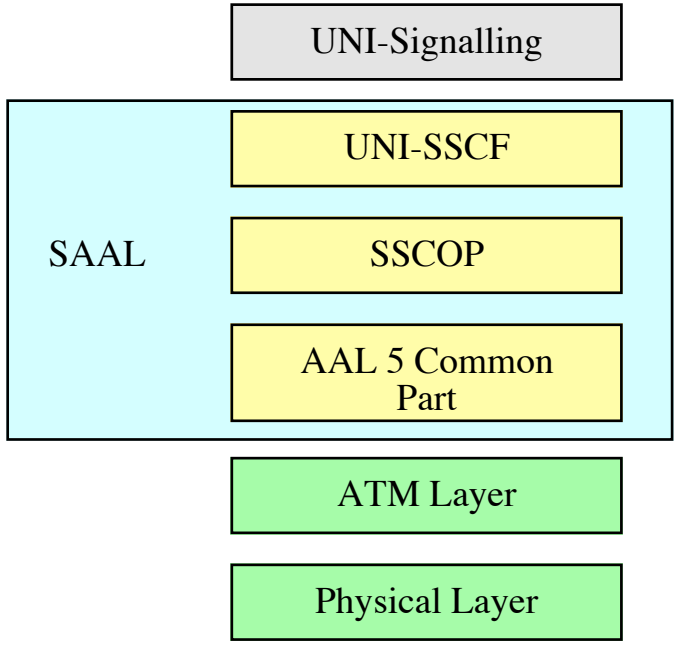




5.2 ATM-Signalisierung Q.2931 und ATM-Forum UNI

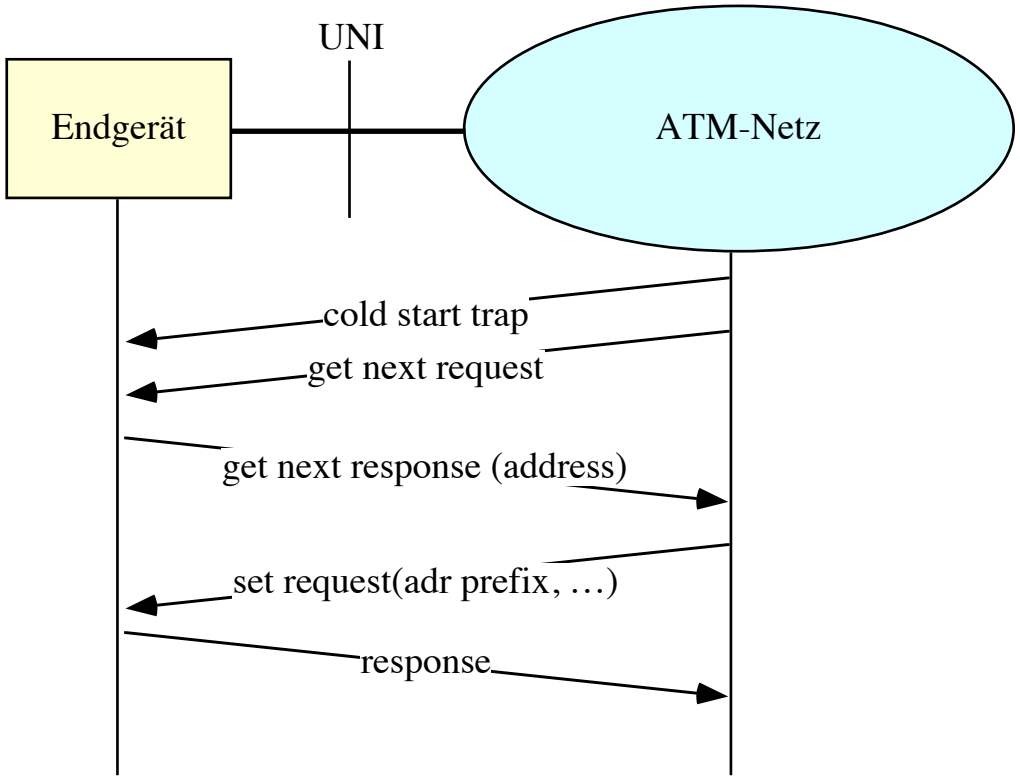
- PVC - Permanent Virtual Circuits
  - konfiguriert im Cross-Connect
  - mit Attributen cell-rate etc.
  - Änderung i.A. manuell
- SVC - Switched Virtual Circuits
  - geschaltet auf Anforderung
  - automatisch
  - Aufbau << 1 sec
- Attribute des Verbindungsaufbaus
  - Adressen (called party, calling party)
  - Verkehrscharakteristik, QoS
  - VPI/VCI als Antwort
- Standardisierung
  - Network Network Interface (NNI), PNNI
  - User Network Interface (UNI)
  - ITU Q.2931

- Fundament Q.931
- Umgebung im ATM
  - SAAL: Signalling ATM Adaptation Layer (AAL 5)
  - Service Specific Coordination Function (SSCF)
  - Service Specific Connection-Oriented Protocol (SSCOP)
  - SSCOP: zuverlässige Übertragung (LAP-D Ersatz, entfernt ähnlich)

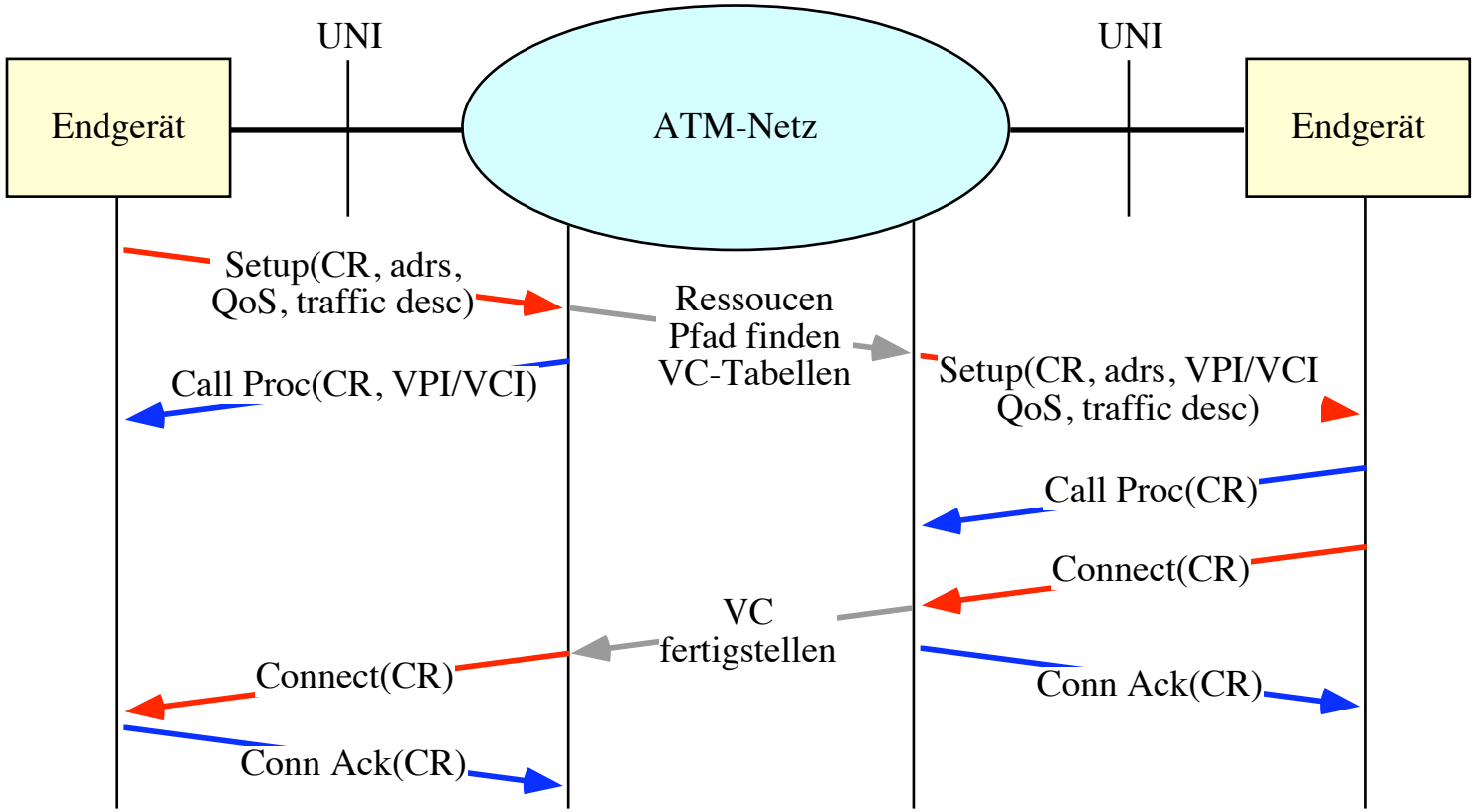


- Signalisierung mit VPI = 0 und VCI = 5

- ATM-Adressierung
  - NSAP-Verpackung
  - Domain und Authority Specification, Domain Specific Part
  - z.B. E.164 als Adresse
- Adressregistierung basiert auf SNMP
  - UNI Management Information Base (MIB) im Endgerät
  - ILMI: Interim Local Management Interface

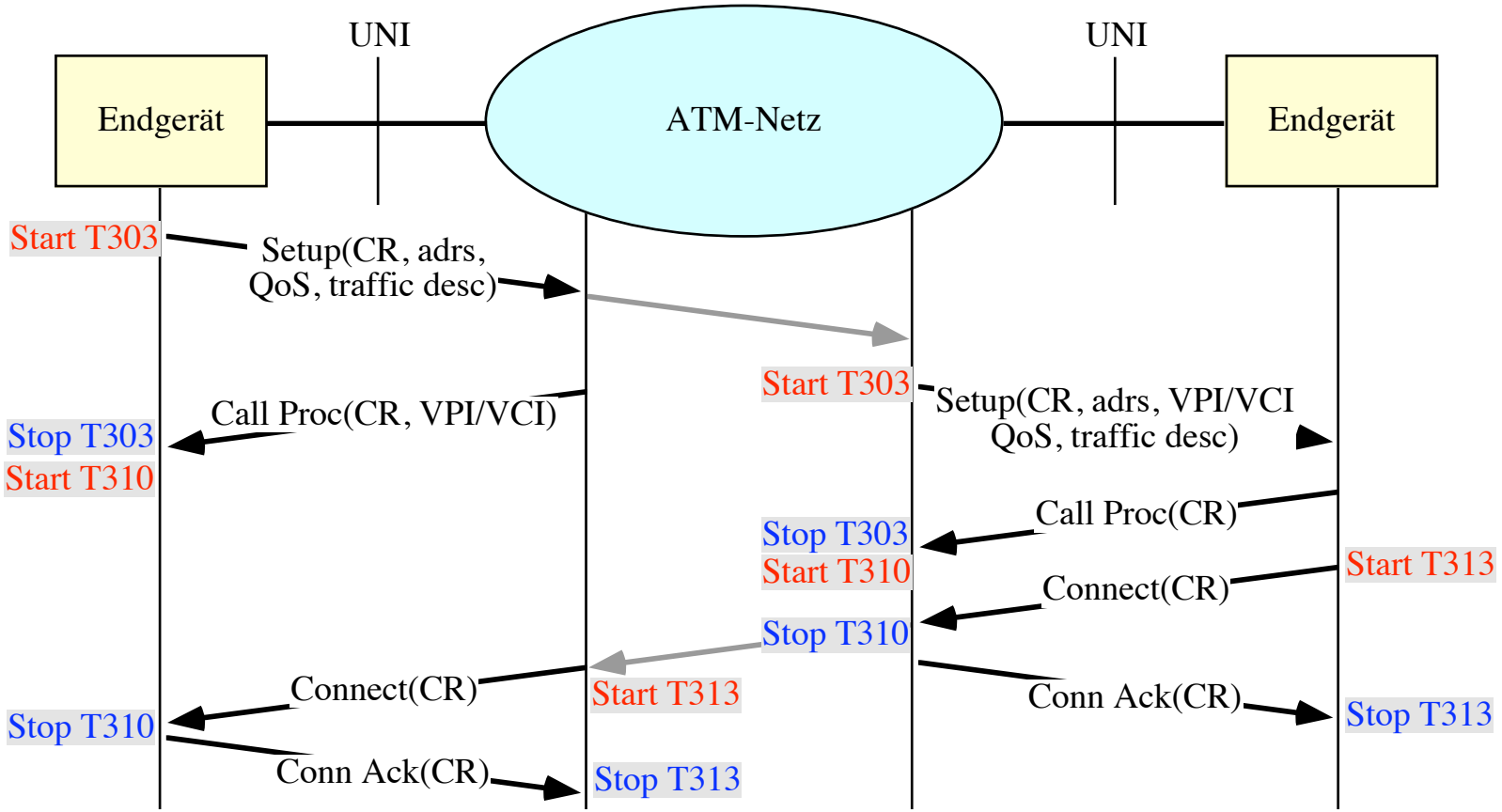


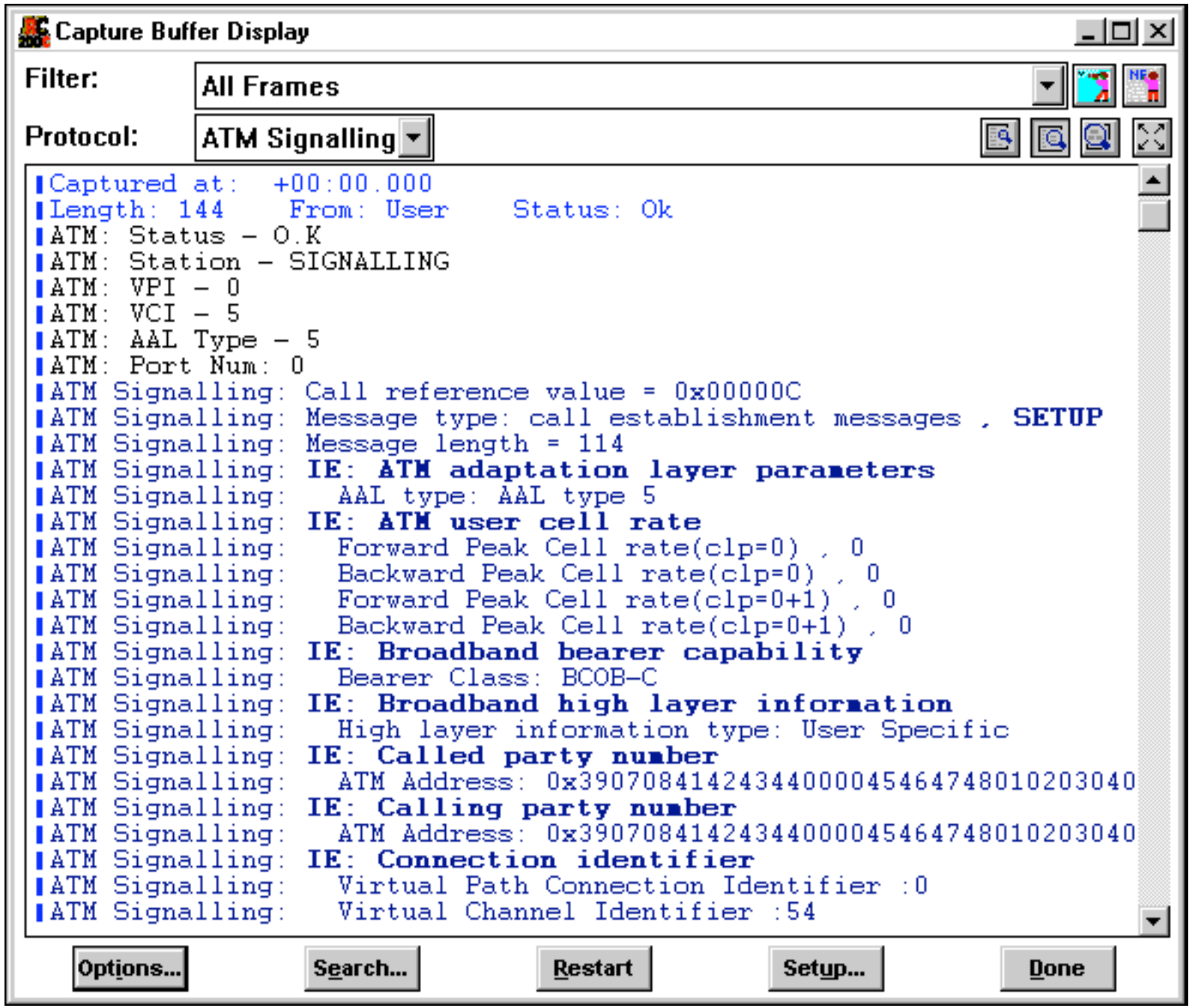
- Setup-Prozedur ähnlich ISDN
  - VPI/VCI statt B-Kanal-Identifikation
  - QoS und traffic descriptor statt Bearer Capabilities



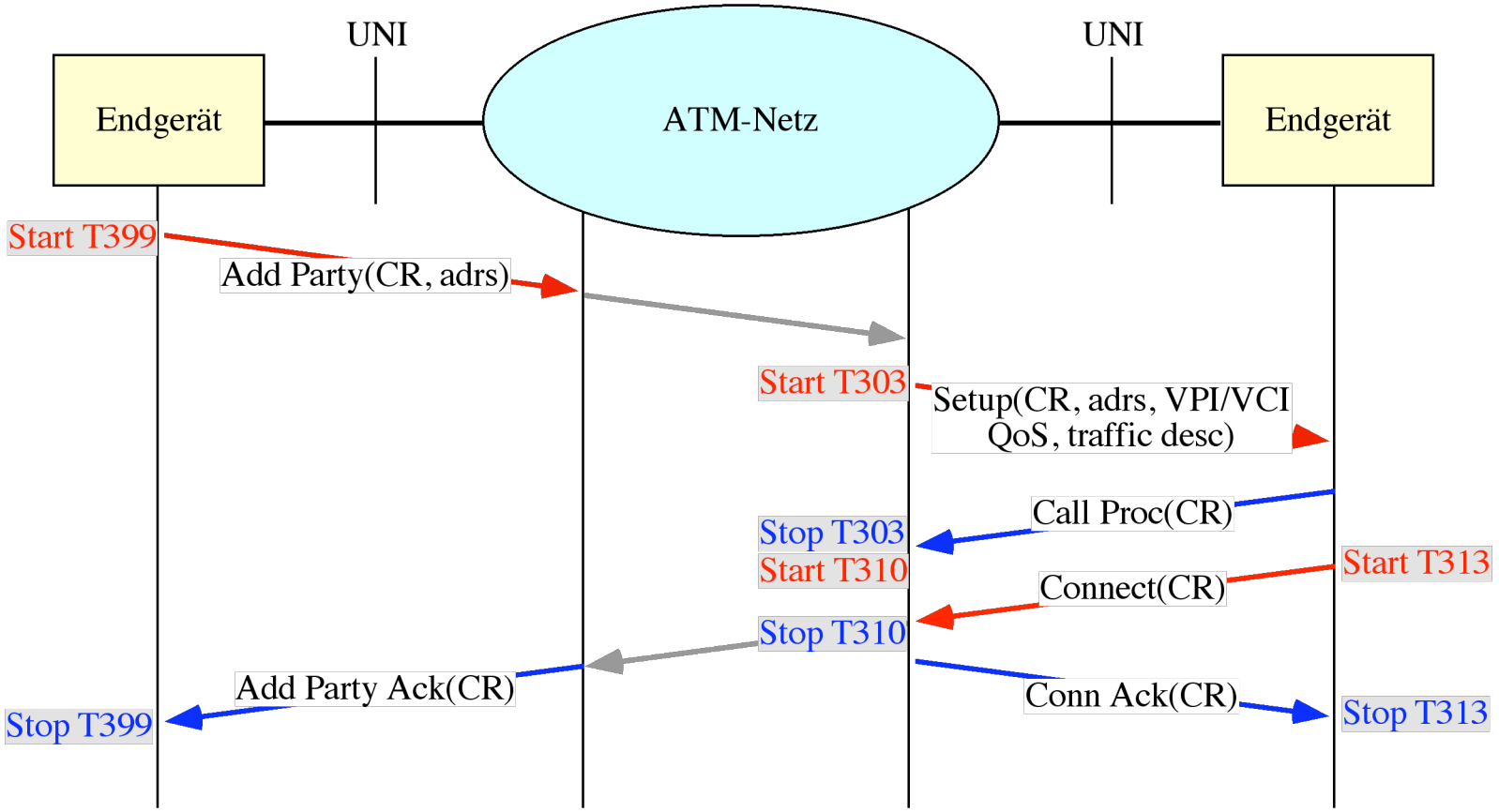
- Verbindungsabbau sehr ähnlich Q.931

- Überwachung durch Timer




Capture Buffer Display
Filter:
All Frames
Protocol:
ATM Signalling
Captured at: +00:00.000
Length: 144 From: User Status: Ok
ATM: Status - O.K
ATM: Station - SIGNALLING
ATM: VPI - 0
ATM: VCI - 5
ATM: AAL Type - 5
ATM: Port Num: 0
ATM Signalling: Call reference value = 0x00000C
ATM Signalling: Message type: call establishment messages , SETUP
ATM Signalling: Message length = 114
ATM Signalling: IE: ATM adaptation layer parameters
ATM Signalling: AAL type: AAL type 5
ATM Signalling: IE: ATM user cell rate
ATM Signalling: Forward Peak Cell rate(clp=0) , 0
ATM Signalling: Backward Peak Cell rate(clp=0) , 0
ATM Signalling: Forward Peak Cell rate(clp=0+1) , 0
ATM Signalling: Backward Peak Cell rate(clp=0+1) , 0
ATM Signalling: IE: Broadband bearer capability
ATM Signalling: Bearer Class: BCOB-C
ATM Signalling: IE: Broadband high layer information
ATM Signalling: High layer information type: User Specific
ATM Signalling: IE: Called party number
ATM Signalling: ATM Address: 0x39070841424344000045464748010203040
ATM Signalling: IE: Calling party number
ATM Signalling: ATM Address: 0x39070841424344000045464748010203040
ATM Signalling: IE: Connection identifier
ATM Signalling: Virtual Path Connection Identifier :0
ATM Signalling: Virtual Channel Identifier :54
Options...
Search...
Restart
Setup...
Done

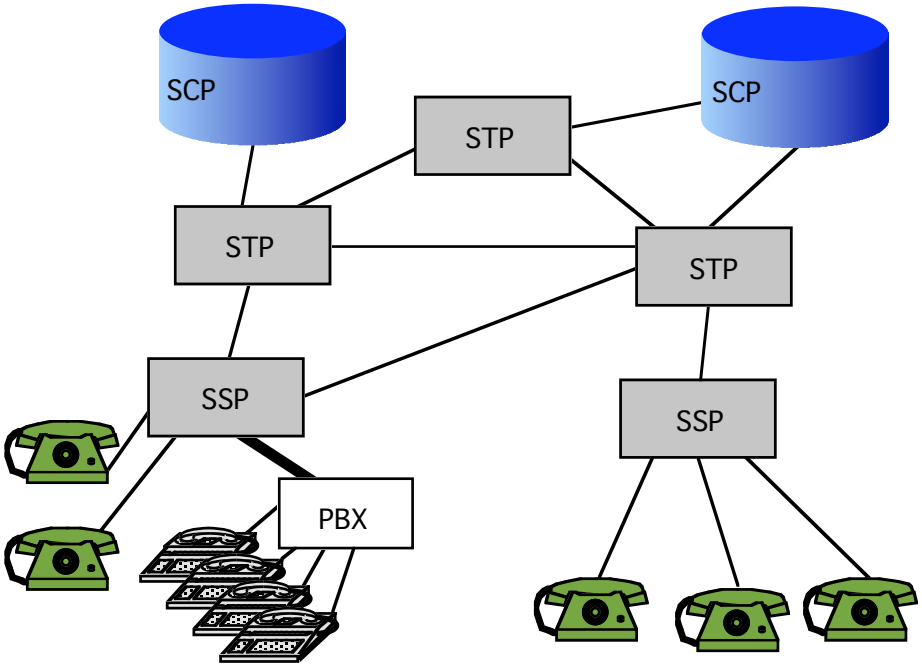
- Mehrpunkttopologien
  - basiert auf Punkt-zu-Punkt verbindung
  - add party, auch für mehrere Teilnehmer
  - drop party



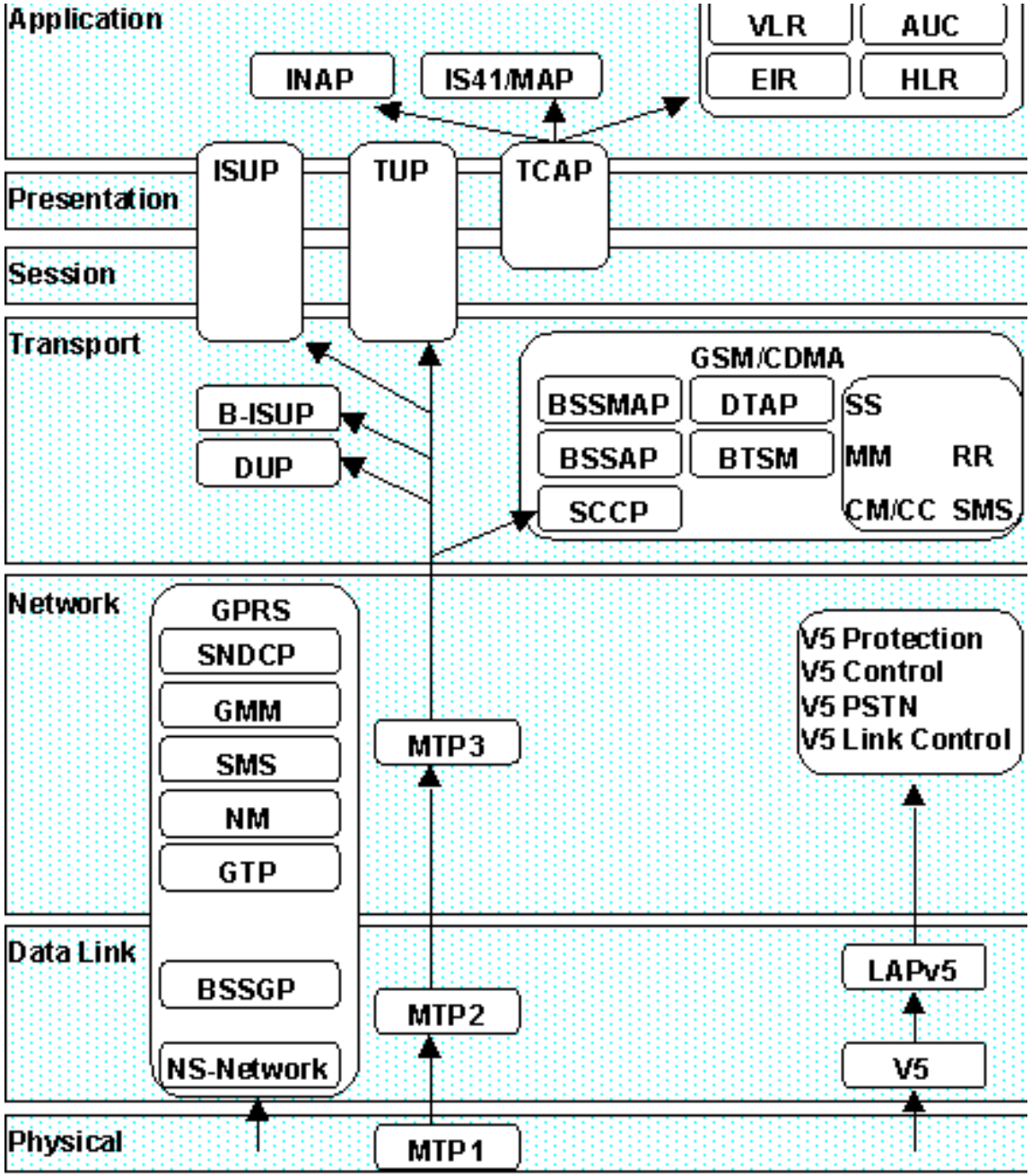
- Modifizierte Informations Elemente (Broadband …)
  - Bearer capability (delay, CBR, VBR, …)
  - higher und lower layer compatibility
  - repeat indicator
- AAL-Parameter (neu)
  - Typ: 1, 3/4, 5
  - circuit emulation, high quality audio, video, …
  - maximale SDU-Größe (AAL 3/4)
  - CBR: 64 kbit/s, DS1, E1, n*64, …)
  - AAL 5 mode: message oder streaming
  - clock und error recovery für AAL 1
- ATM user traffic descriptor
  - peak cell rate (forward und backward)
  - sustainable cell rate (forward und backward)
  - maximum burst size (forward und backward)
- QoS-Parameter: CBR, VBR, ABR, unrestricted
- Connection Identifier: VPI/VCI

5.3 Signalisierungssystem 7 (SS#7)

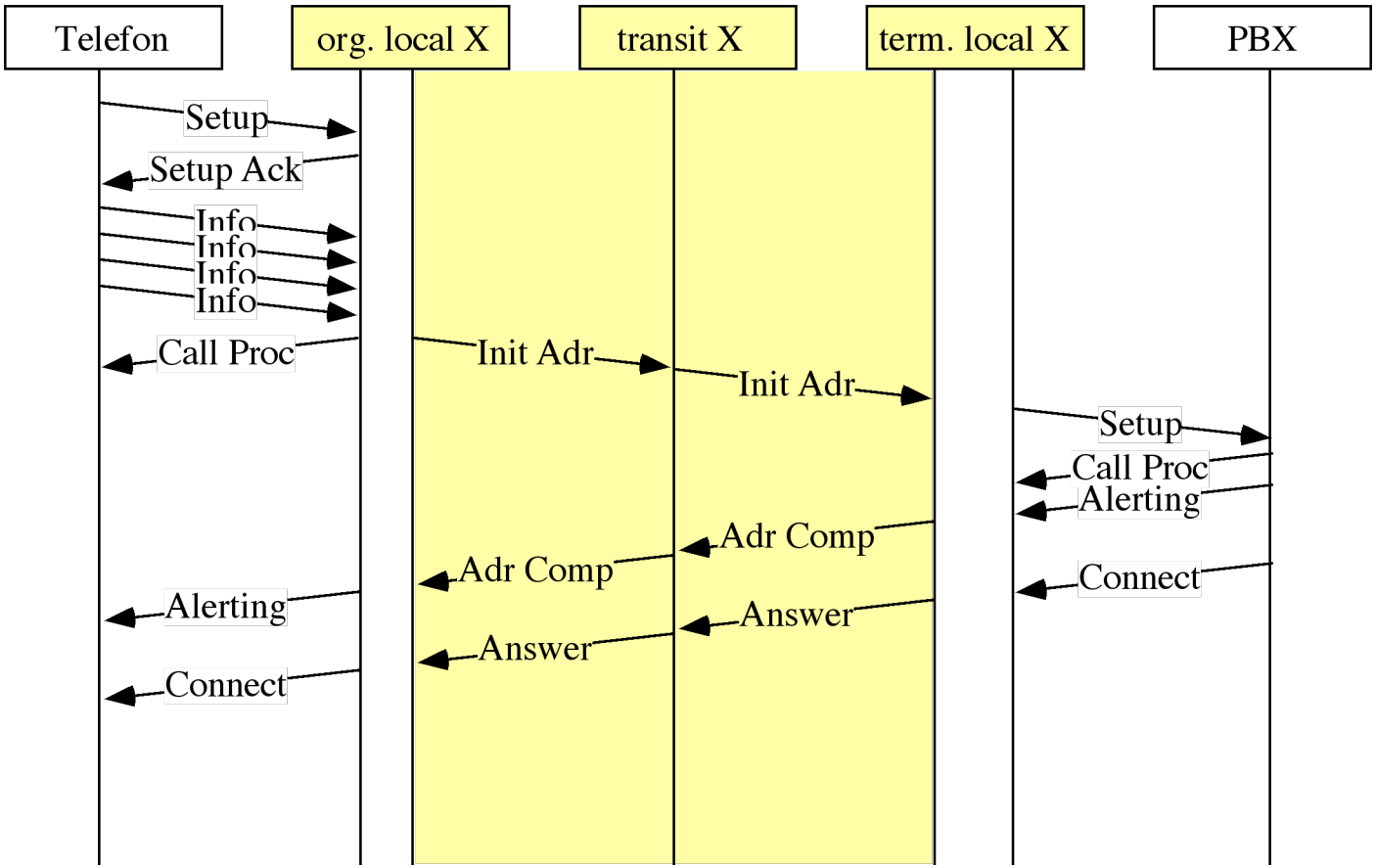
- Common Signalling Channel
  - getrennt vom Medienstrom-Netz
  - 56 oder 64 kbit/s Links
- Verbindungskontrolle im Netzwerk
  - Vermittlungen und Netzwerke
  - GSM-ISDN, GSM-GSM, …
- Service Switching Point (SSP)
  - Vermittlung
  - Netz-Übergabepunkte
  - verstecken Endgeräte
- Service Control Point (SCP)
  - Rufbewertung, Wegfindung
  - Management des Verbindungsaufbaus
  - Prozeduren und Datenbank (VLR, HLR)
- Signalling Transfer Point
  - Router zwischen SCPs und SSPs



- Message Transfer Part
  - MTP2/Q.703
  - entfernt HDLC ähnlich
  - MTP3/Q.704, Routing
- SCCP
  - Signalling Conn. Control
  - => OSI Schicht 3
- User Parts
  - Verbindungssteuerung
  - Pfade finden
  - Leitungen zuordnen
- ISDN User Part
- MAP: Mobile Application Part
  - IS-41 and GSM
  - MSC-VLR fragt HLR
  - Authentisierung
  - Geräte-Identification, Roaming



- ISDN User Part

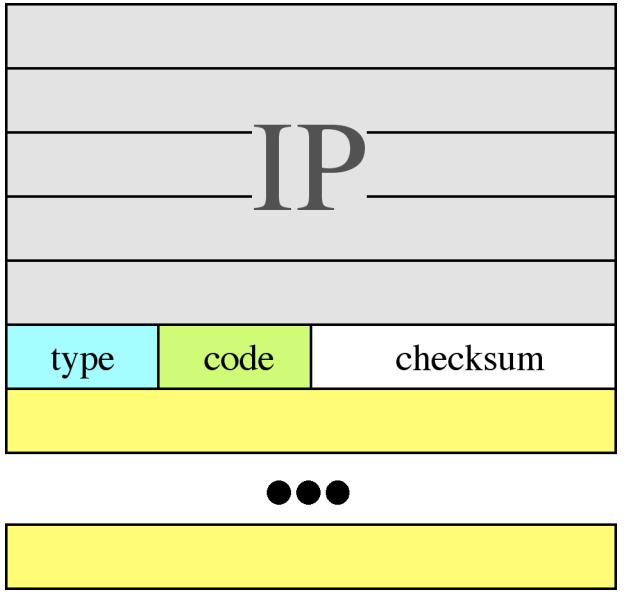


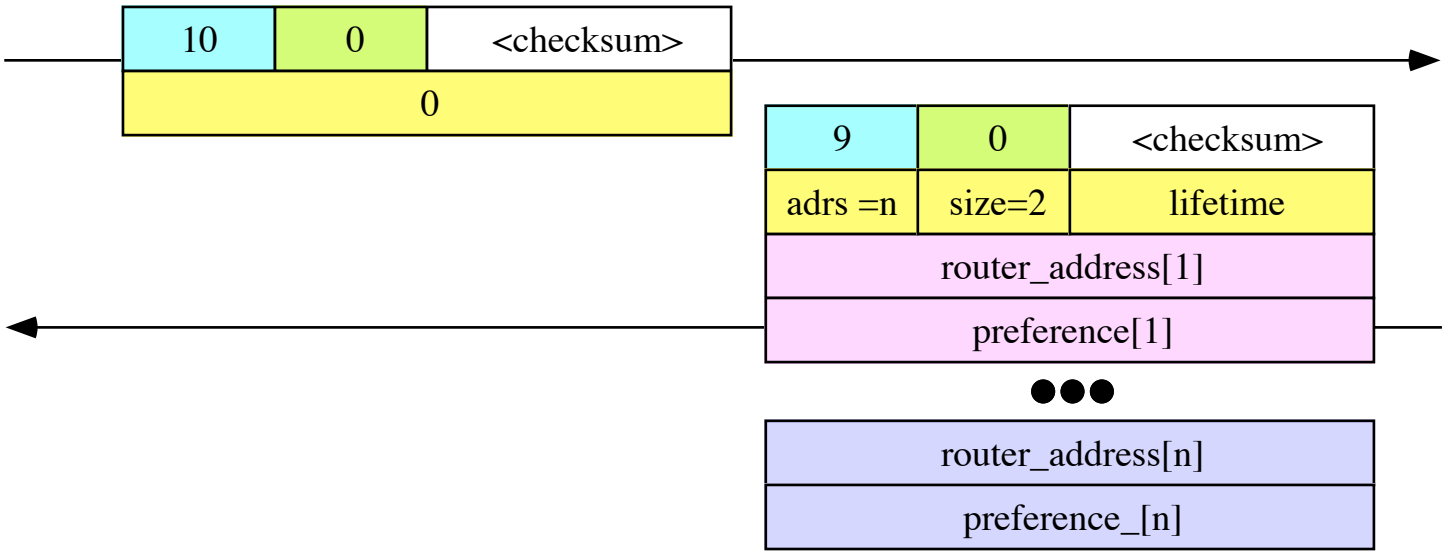
- Call Routing aus Datenbankabfrage
- local exchanges und transit exchanges sind SSPs
- IAM: Nummer bewerten, nächste SSP suchen, reservieren
- ACM: Kanal durchschalten
- ANswer Message

- Initial Address Message
  - called und calling party number
  - forward call indicator
  - nature of connection, user service
- Address Complete Message
  - charge indicator
  - called party status and category indicators (besetzt …)
  - echo control indicator
  - interworking, holding, …
- ANswer Message
  - backward call indicator
  - access und network transport
  - call reference
  - notification indicator
- Siehe: http://www.pt.com/page/tutorials/ss7-tutorial
- Heute auch über IP: IP-STP

5.4 ICMP: Internet Control Message Protocol

- Nachrichten der über IP-Datagramme und IP-Router
  - echo request/reply für ping
  - destination unreachable (code: network, host, protocol, port, …)
  - Verstopfungskontrolle (source quench) veraltet
  - Umleitung (redirect),
  - 'time exceeded' (TTL oder reassembly-timeout)
  - Zeitstempel
  - transportiert in der Nutzlast von IP-Paketen
- Router-Bekanntgabe und -Suche

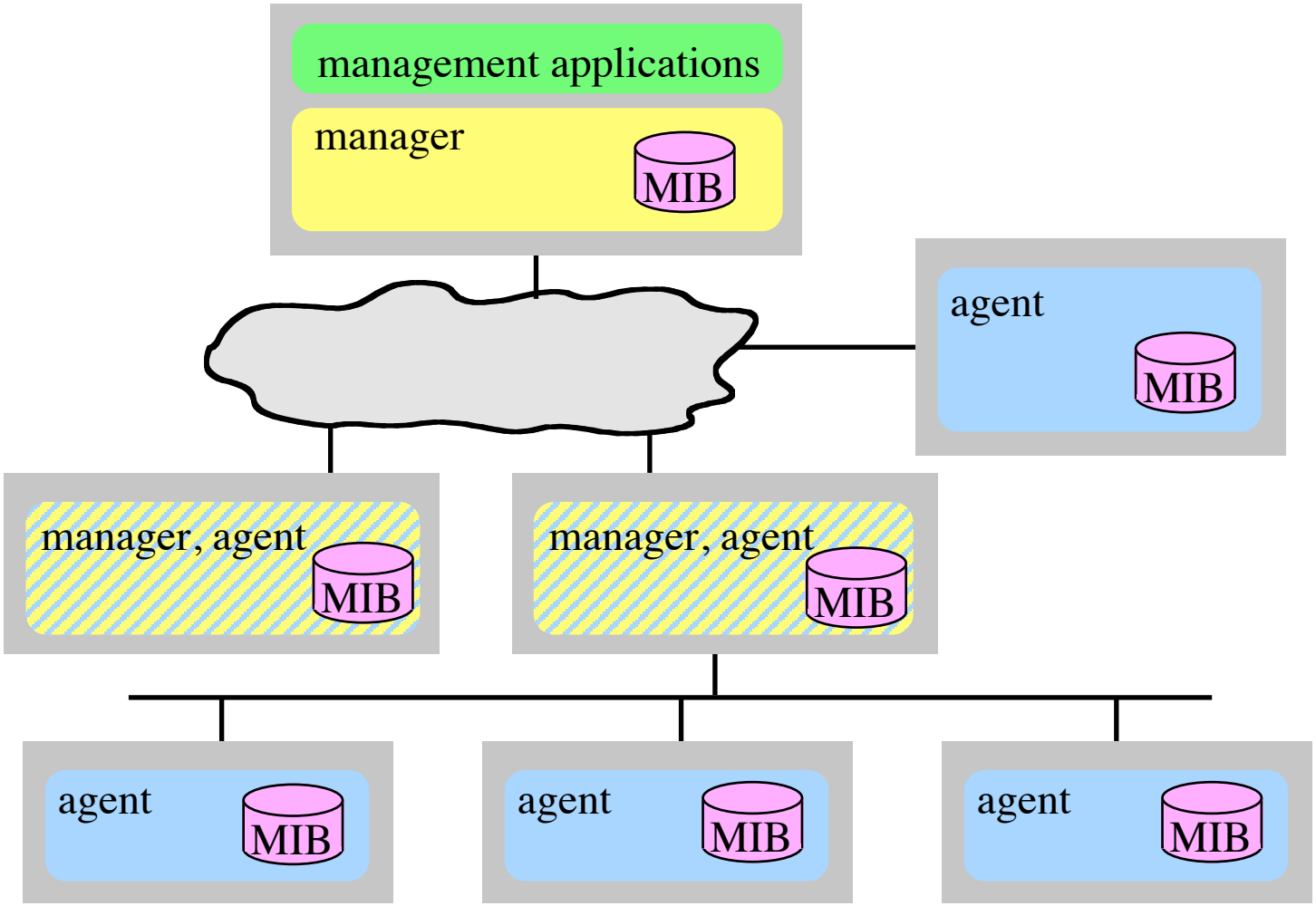




5.5 Netzwerkmanagement SNMP

- Infrastruktur für Netzwerkmanagement
- Management von Netzwerk-Elementen
  - Hosts, X-Terminals, Terminal Server
  - Drucker
  - Router
  - alle Netzwerk-verbundenen Geräte (Variablen-Definition …)
- Funktionen
  - Statusabfrage => Object-Variable lesen (get request)
  - Funktionen konfigurieren: Object-Variable setzen (set request)
  - Alarme (traps)
- Komponenten
  - Management Information Base (MIB): RFC 1213
  - Structure of Management Information (SMI): RFC 1155
  - Simple Network Management *Protocol* (SNMP): RFC 1157

- Management-Station
  - Protokoll, MIB, Grafisches UI

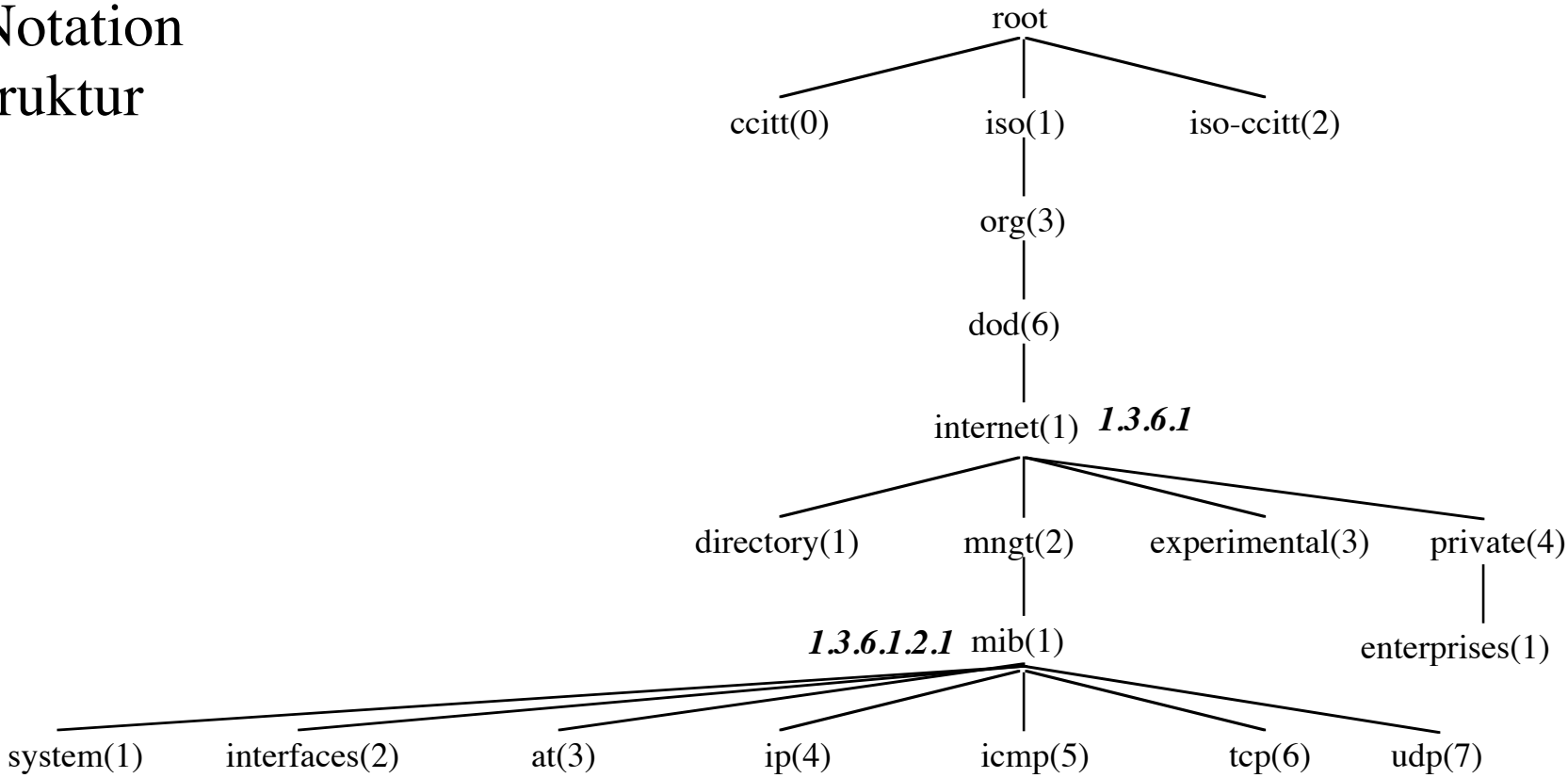


- Agent
  - Protokoll, MIB

- Structure of Management Information (SMI)
  - Datentypen
  - skalare Typen
  - Tabellen: 2-dimensionale Arrays

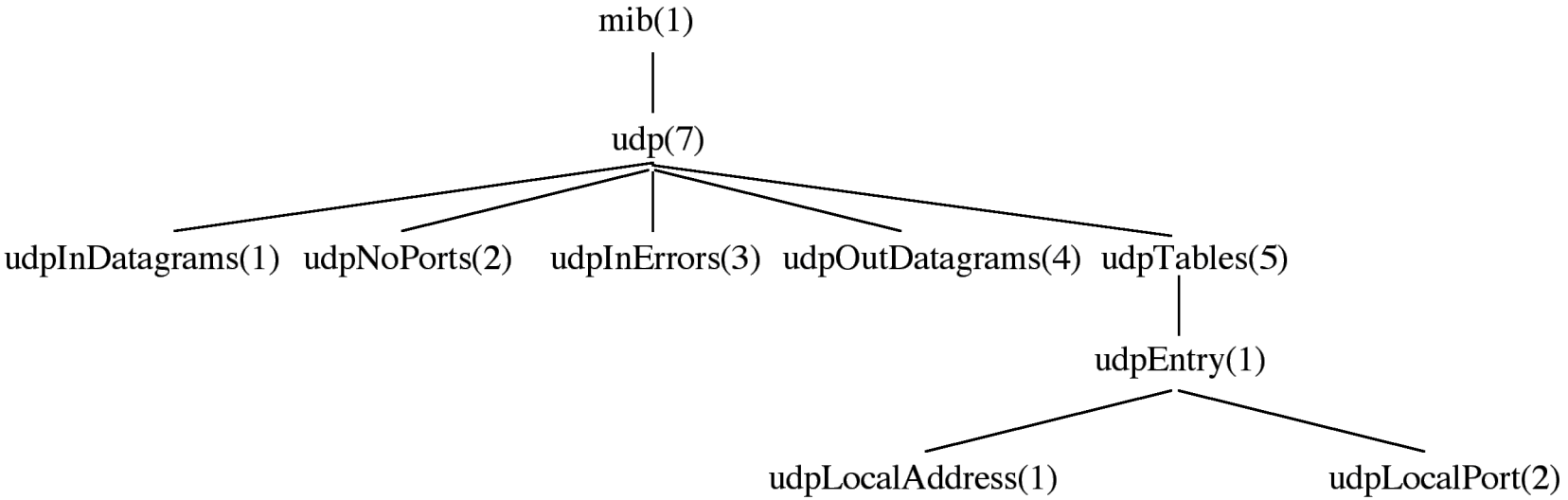
INTEGER	$-2^{31} .. 2^{31}-1$
UInteger32	$0 .. 2^{32}-1$
Counter32, Counter64	nur aufwärts zählen, Wrap around
Gauge32	zu- und abnehmen, bleibt bei $2^{32}-1$ stehen
TimeTicks	zählt 10 msecs
OCTET STRING	
IpAddress	4 Byte
PhysAddress	z.B. 6 Byte Ethernet Adresse
DisplayString	String, NVT-ASCII
OBJECT IDENTIFIER	
Opaque	Bitfeld
BIT STRING	Bits mit Namen
SEQUENCE	zur Definition von Datenstructuren

- Namensgebung: Object Identifier
  - Punkt-Notation
  - Baumstruktur



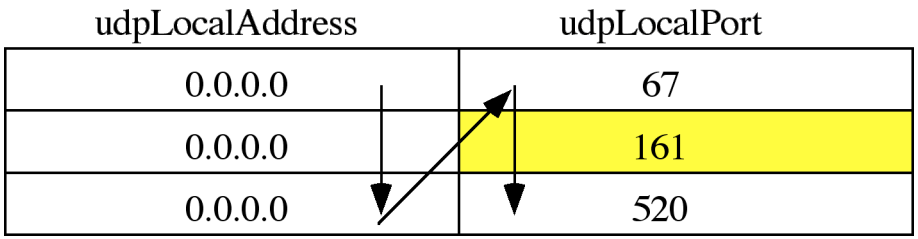
- ASN.1 Makro zur Definition von Objekten (Basic Encoding Rules)
  - SEQUENCE, SEQUENCE OF, OBJECT TYPE
  - Vergabe von Namenselementen
  - not accessible, read, read/write

- Vordefinierte Objekte (RFC 1213, MIB-II)
  - system: Gerätedaten (displaystring, sysUpTime, sysName…)
  - interface, at (address translation (ARP-cache))
  - ip, icmp, tcp, udp
  - routing tables
  - traps (siehe unten)
- Beispiel udp



  - Anzahl Pakete für Ports ohne Listener
  - Anzahl sonstiger Fehler (checksum, …)

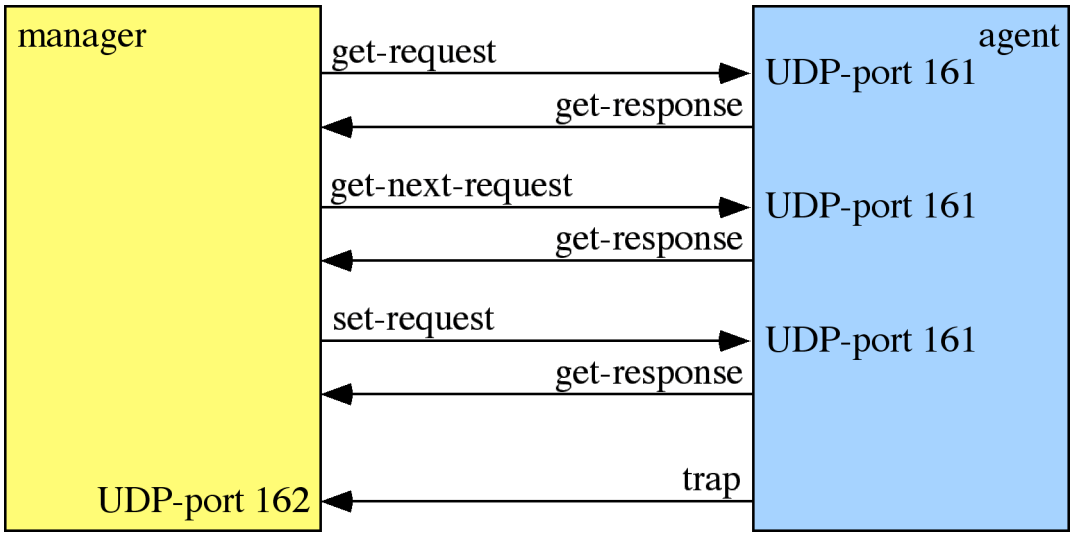
- Bezeichnung
  - 1.3.6.1.2.1.7.4.0
  - iso.org.dod.internet.mngt.mib.udp.udpOutDiagrams.0
  - abgekürzt: udpOutDiagrams.0
  - normale Variablen: 0 anhängen
- Tabellen-Objekt-Bezeichnung
  - Tabellen immer aufsteigend sortiert
  - Zugriff Spalte-Zeile
  - 1.3.6.1.2.1.7.5.1.1.0.0.0.0.161 - udpLocalPort.0.0.0.0.161



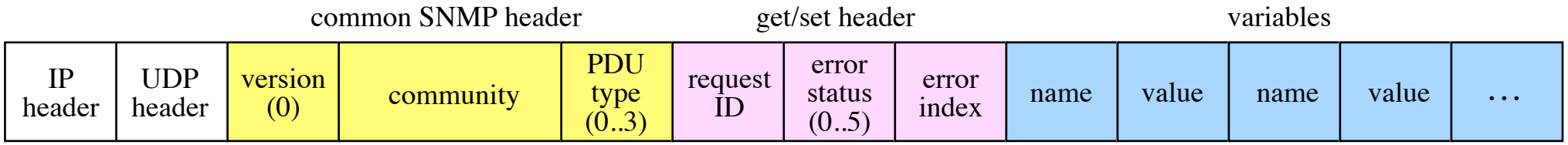
- Bsp: snmpi

```
c:\iptools\> snmpi -a eincomputer -c geheim
snmpi> get udpInDatagrams.0 udpNoPorts.0
udpInDatagrams.0=2345677
udpNoPorts.0=12
snmpi> next
udpInErrors.0=33
snmpi> quit
c:\iptools\>
```

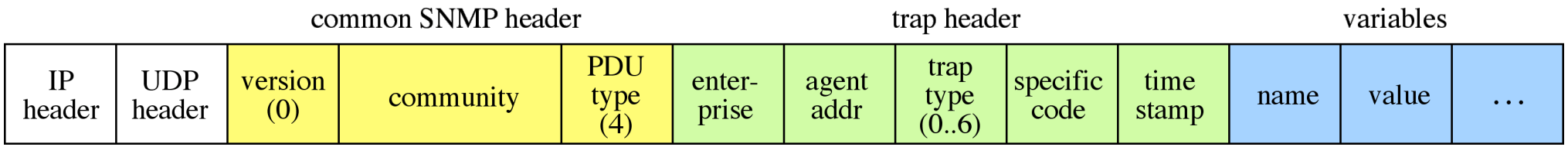
- SNMP Protokollelemente



- SNMP PDU



- trap

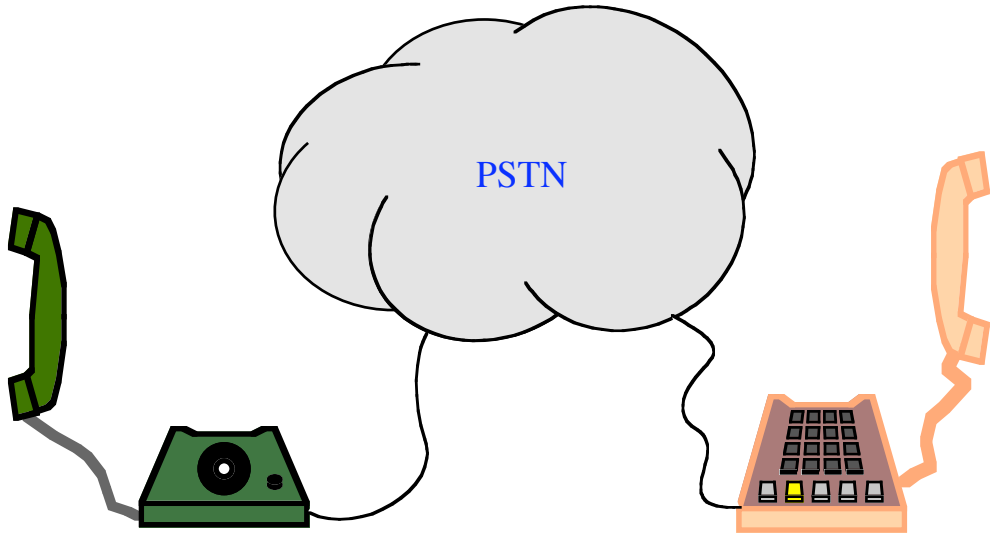


tryp type: coldStart, warmStart, linkDown, linkUp,…

6. Anwendungen

6.1 Kommunikation Person-zu-Person

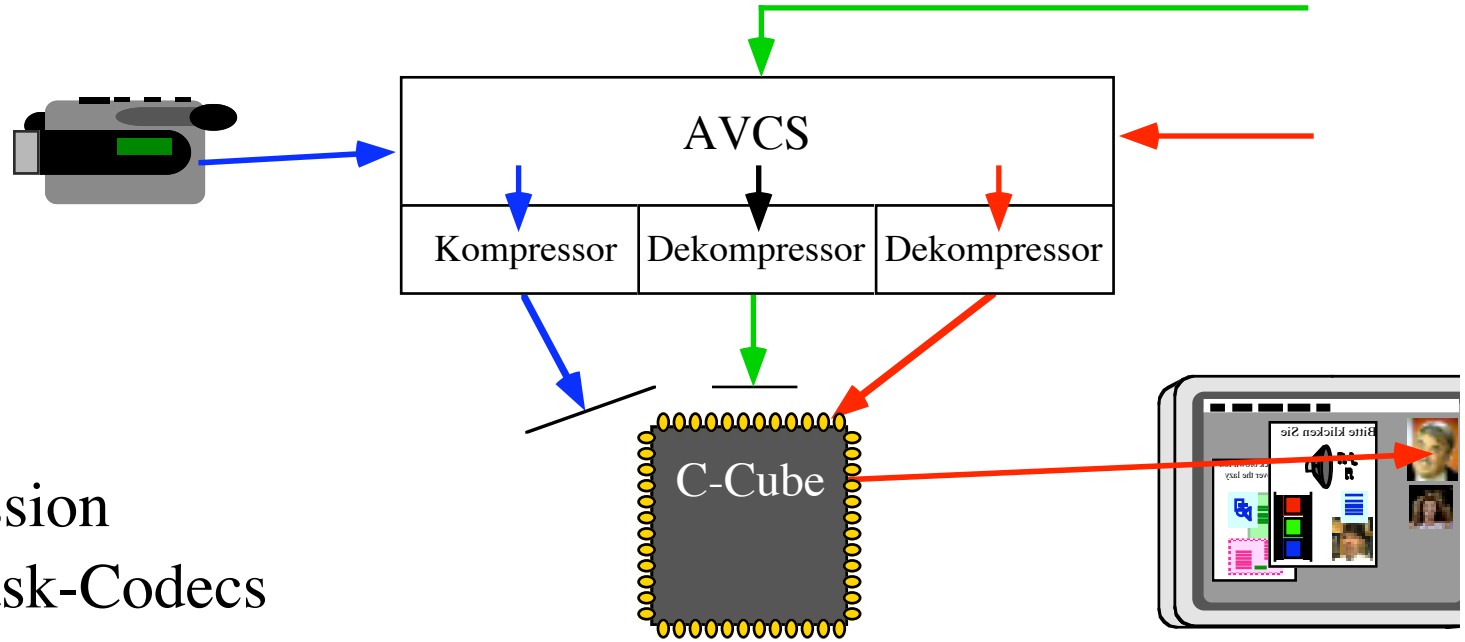
6.1.1 Telefondienst

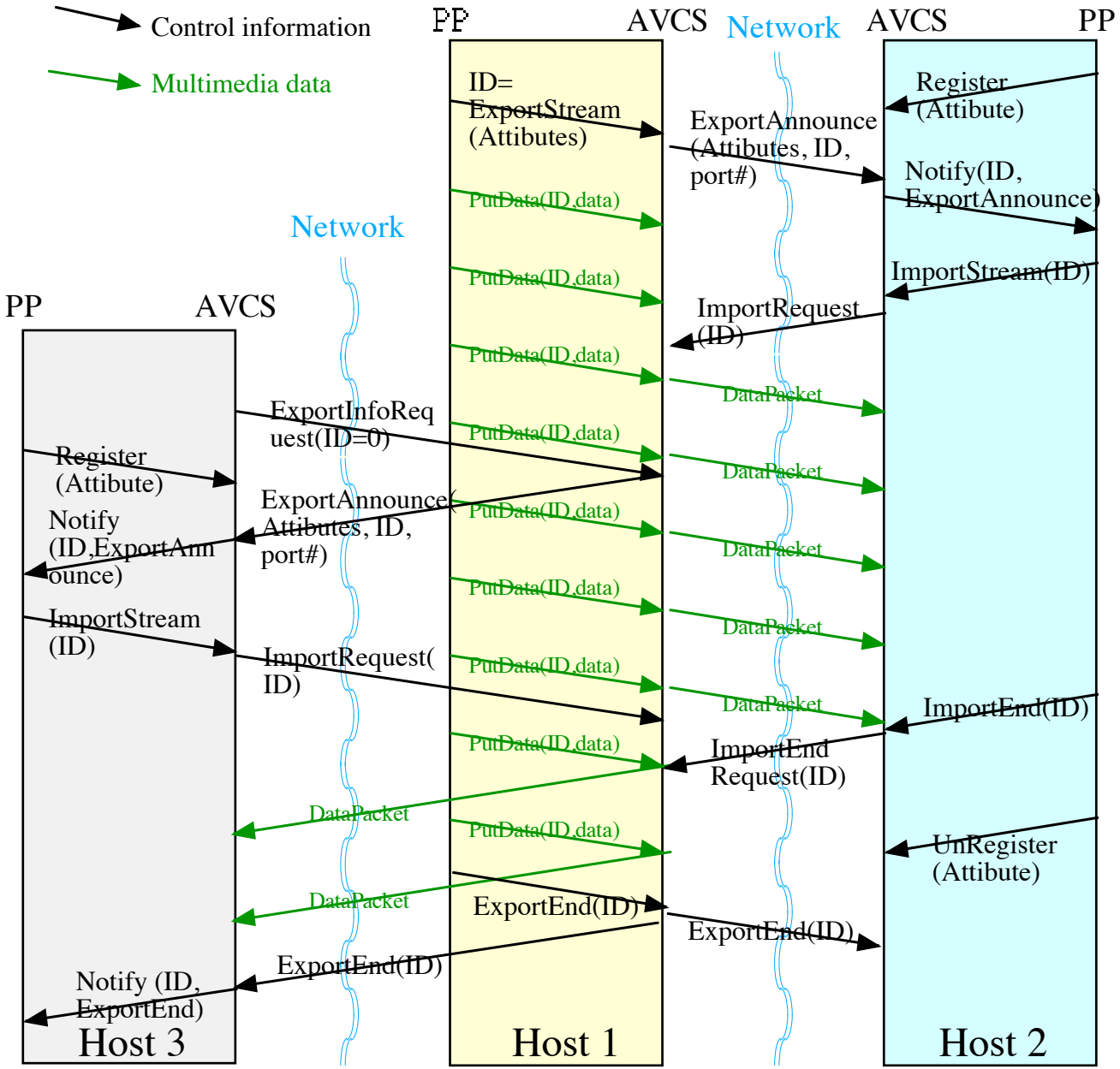


- 10^9 Teilnehmer?
- Endgeräte standardisiert
  - Mikrofon
  - Lautsprecher
  - Bedienprozedur
- Netzwerk heterogen

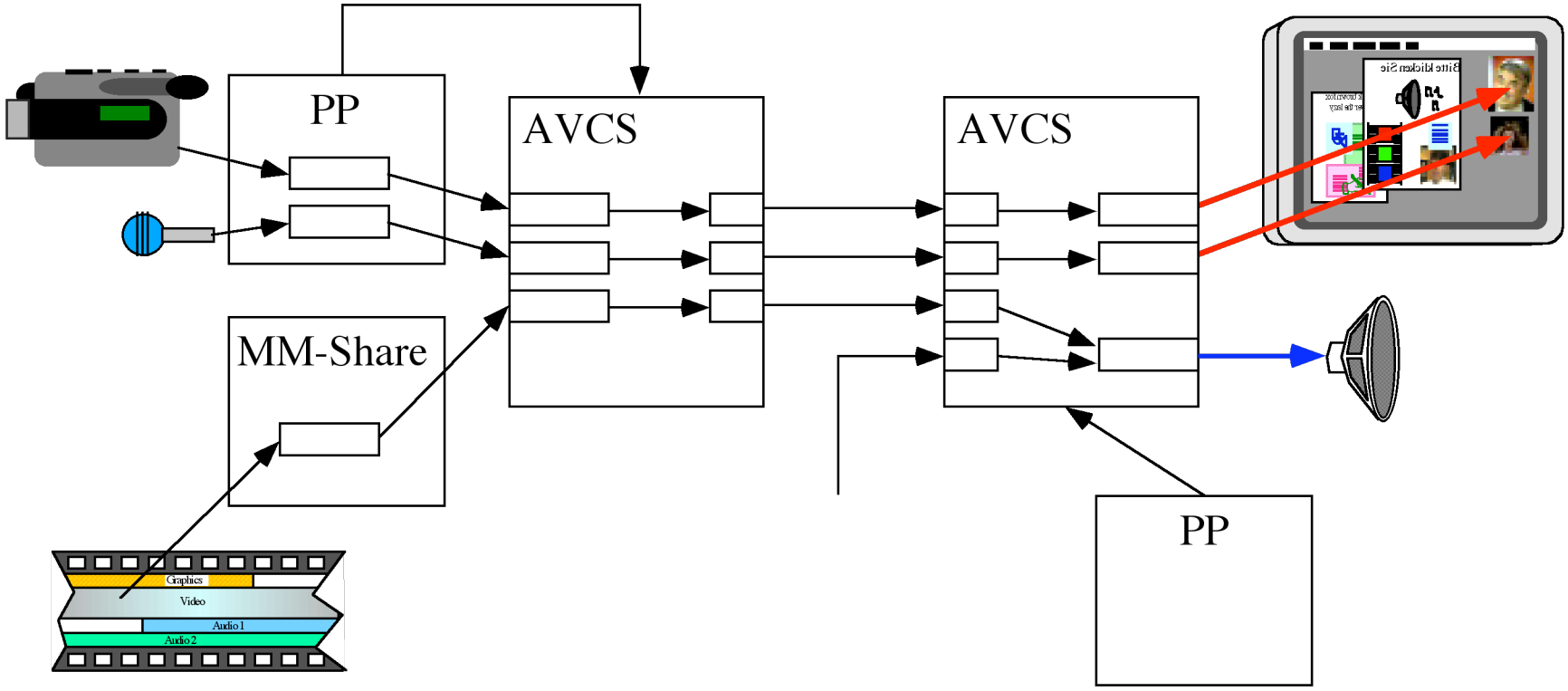
6.1.2 CIO-Phone

- Einfache Videotelefonapplikation
- Audio-Video Communcation Service
  - für VideoPhone und Multimedia-Ströme der Applikationen
  - UDP-basiert, Punkt-zu-Punkt
  - eventuell Datenreplikation
- AVCS: Echtzeitorientiert
  - kein Delay absichtlich eingefügt
  - Pakete rigoros verwerfen:
    Ankunft zu spät
    Sendepuffer voll
- Audio: 16 bit, 11.025 samples
- M-JPEG
  - meist Hardware
  - Softwaredekompression
- Problem der Single-Task-Codecs
- AVCS Verbindungsmanagement




Control information
Multimedia data
PP
AVCS
Network
AVCS
PP
ID=
ExportStream
(Attibutes)
ExportAnnounce
(Attibutes, ID,
port#)
Register
(Attibute)
Notify(ID,
ExportAnnounce)
PutData(ID,data)
Network
ImportStream(ID)
PP
AVCS
ImportRequest
(ID)
PutData(ID,data)
DataPacket
ExportInfoReq
uest(ID=0)
PutData(ID,data)
DataPacket
Register
(Attibute)
ExportAnnounce(
Attibutes, ID,
port#)
PutData(ID,data)
DataPacket
Notify
(ID,ExportAnn
ounce)
PutData(ID,data)
DataPacket
ImportStream
(ID)
ImportRequest(
ID)
PutData(ID,data)
DataPacket
ImportEnd(ID)
PutData(ID,data)
ImportEnd
Request(ID)
DataPacket
PutData(ID,data)
UnRegister
(Attibute)
DataPacket
ExportEnd(ID)
ExportEnd(ID)
ExportEnd(ID)
Notify (ID,
ExportEnd)
Host 3
Host 1
Host 2

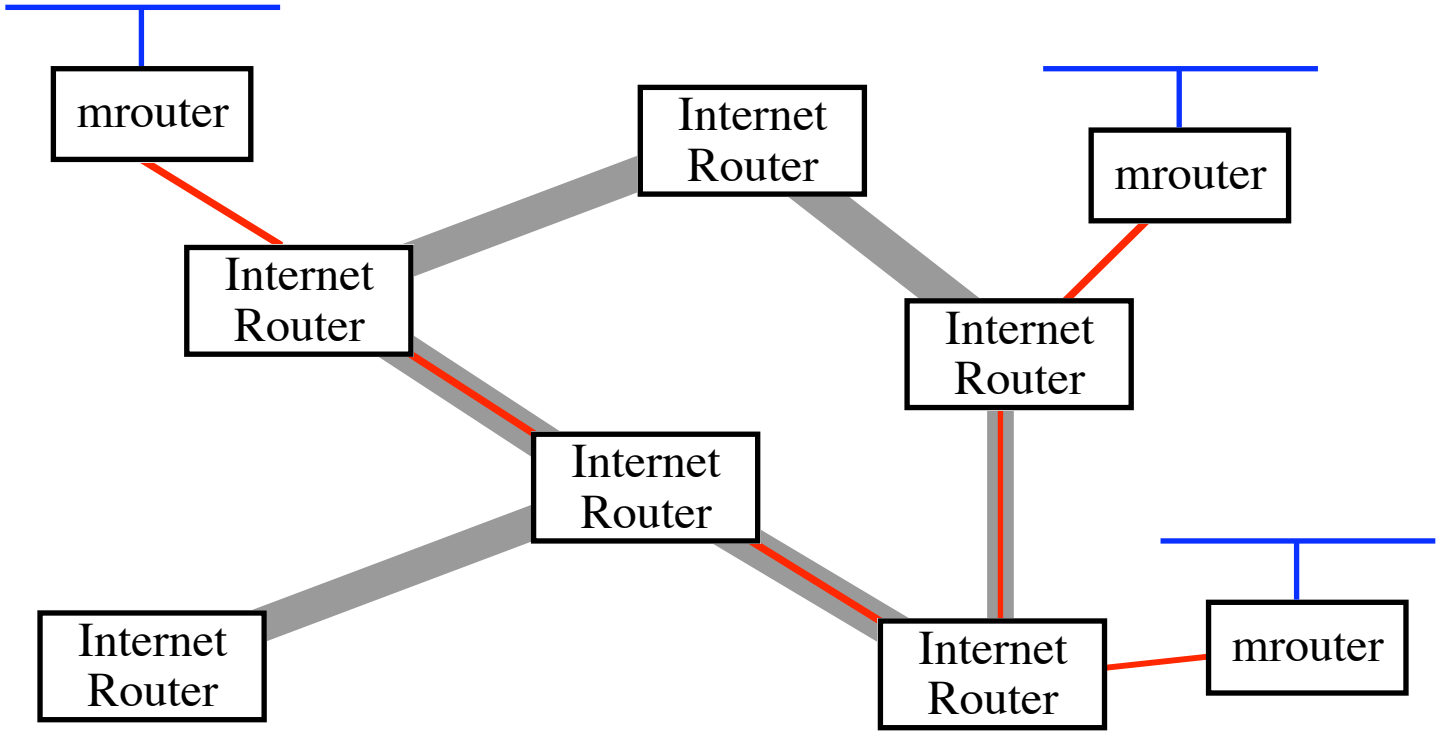
- PicturePhone = User Interface
- AVCS
  - (komprimiert) und dekomprimiert
  - mischt
  - plant Wiedergabe



- Verzögerungen bei der Datenmanipulation
- Senden
  - PP öffnet Digitizer
  - PP bekommt Daten
  - PP übergibt Daten an AVCS
  - AVCS -> Ethernettreiber
- Empfangen
  - Ethernettreiber empfängt Paket
  - wartet auf Paketende!
  - Ethernettreiber -> AVCS
  - AVCS dekomprimiert
  - AVCS mischt
  - AVCS -> Präsentations-Treiber
- UDP-Modell auch im AVCS
  - Gleicher Sampling-Takt?
  - => (fast) keine Pufferung
  - Probleme mit differenzkomprimierten Daten

6.1.3 - 5.1.5 Mbone-Tools

- MBone
  - mrouted tunneln Pakete durch das Internet und replizieren
  - Multicast-Pakete in normale IP-Pakete kapseln
  - besondere Applikationen für Video und Audio
  - Management der Multicasts: finden, subscribe, QoS, …

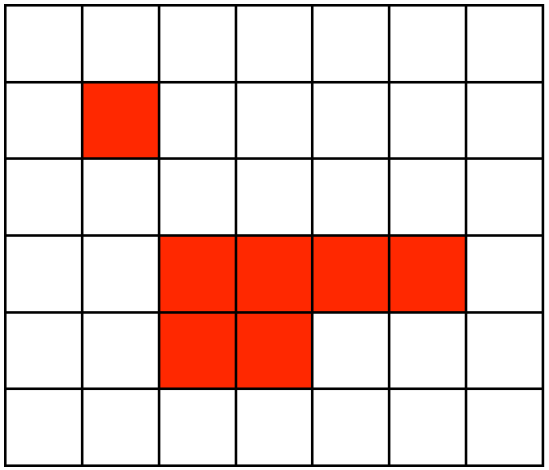


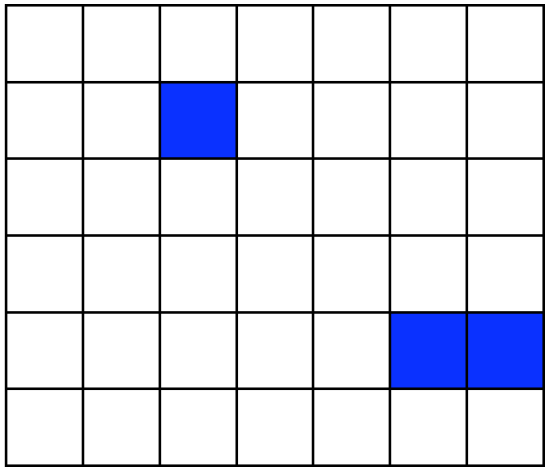
6.1.3 vat

- Visual Audio Tool (Van Jacobsen, Steve McCanne)
- Audioformate
  - PCM: 78 kbit/s = 64 kbit/s μ-law + Paketoverhead
  - DVI (ADPCM): 46 kbit/s
  - GSM: 17 kbit/s
  - LPC: 9 kbit/s
- Internet als Übertragungsnetzwerk
  - Mehrpunktfähigkeit durch Multicast-IP (MBone)
- UDP als Übertragungsprotokoll
  - unzuverlässig: Paketverlust 10 - 50%, keine Wiederholung
  - nur IP-Delay
- Application Layer Framing
  - Zeitstempel
  - Delay abhängig vom Scenario
  - 'Playout Point'
- Pausenerkennung und -unterdrückung

6.1.4 vic

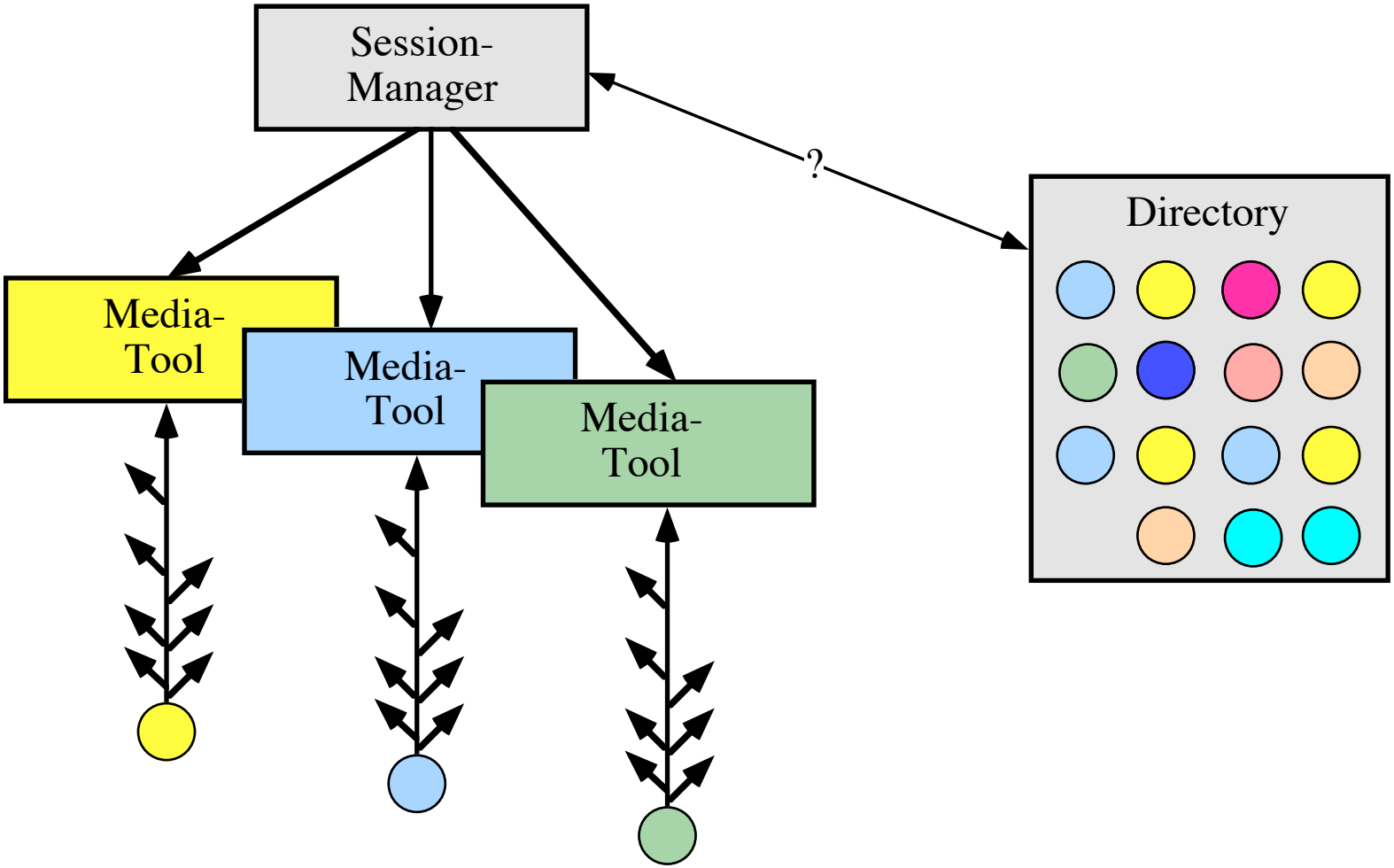
- Video conferencing
- Video-Formate
  - M-JPEG
  - Cell-B
  - nv
  - Intra-H.261
- Intra-H.261
  - Conditional replenishment: nur Änderungen
  - H.261 Mechanismen, um Blöcke zu überspringen
  - keine Differenzkodierung (intercoded blocks)



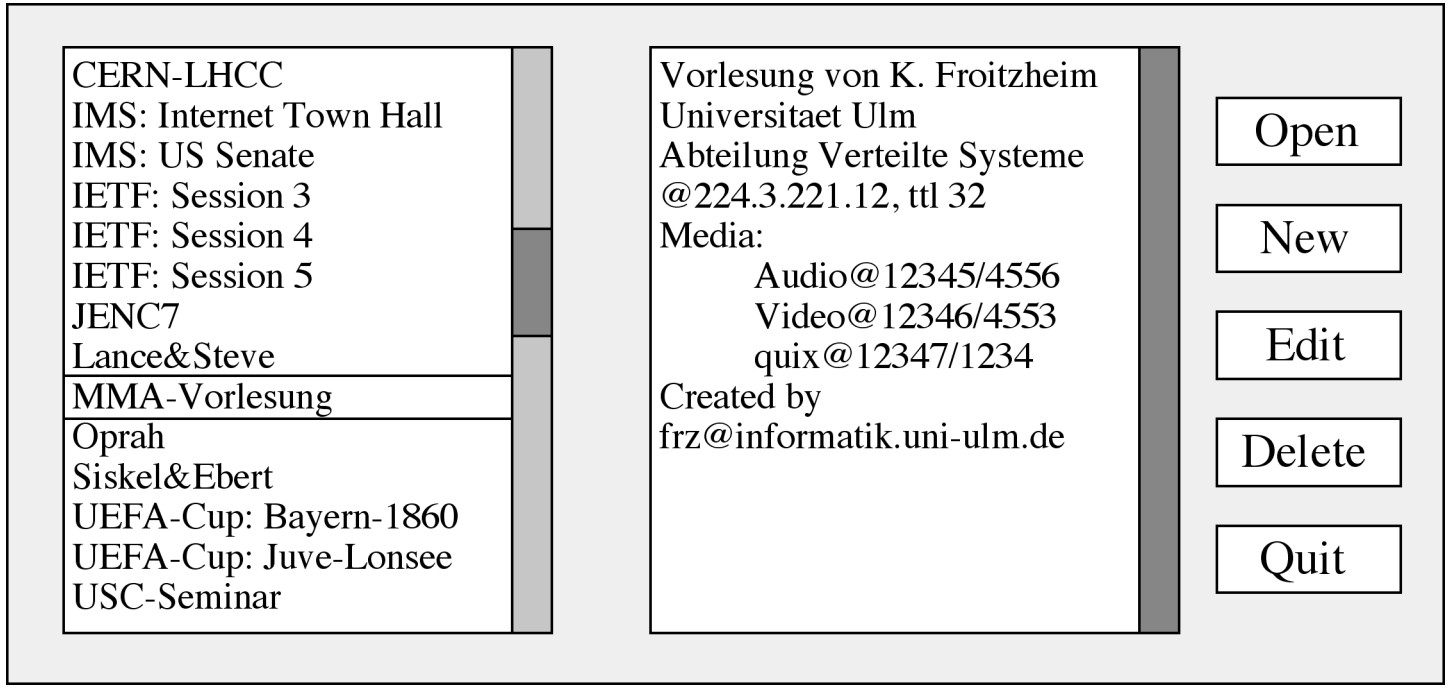


6.1.5 Management der MBone-Tools

- Multicast ≈ Rundfunk
- Media-Tools sind eigenständige Programme

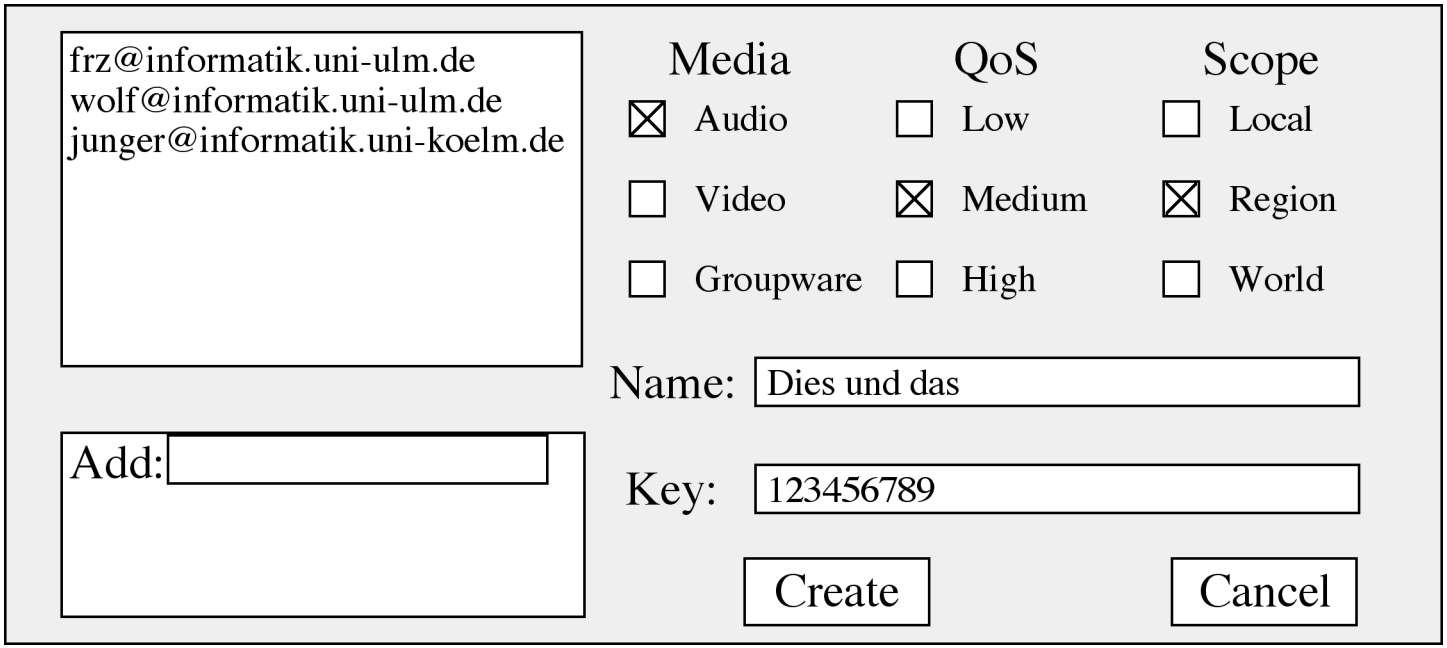


- sd: Session Directory
- Ankündigung von Konferenzen
  - Multicast-IP-Nummern-Vergabe
  - Strombeschreibung: port, ID, scope, Medium, Kodierung
  - Kommentare
- Programmzeitung als Paradigma
- Import: Start des entsprechenden Tools



- Neue Version: sdr

- mmcc: Multimedia Conference Control
  - Anrufparadigma
  - mmcc ruft mmcc
- Verbindungsaufbau
  - Session-Namen, Security und Privacy eingeben
  - Partner aus Liste wählen
  - Medien wählen
  - *Connect*



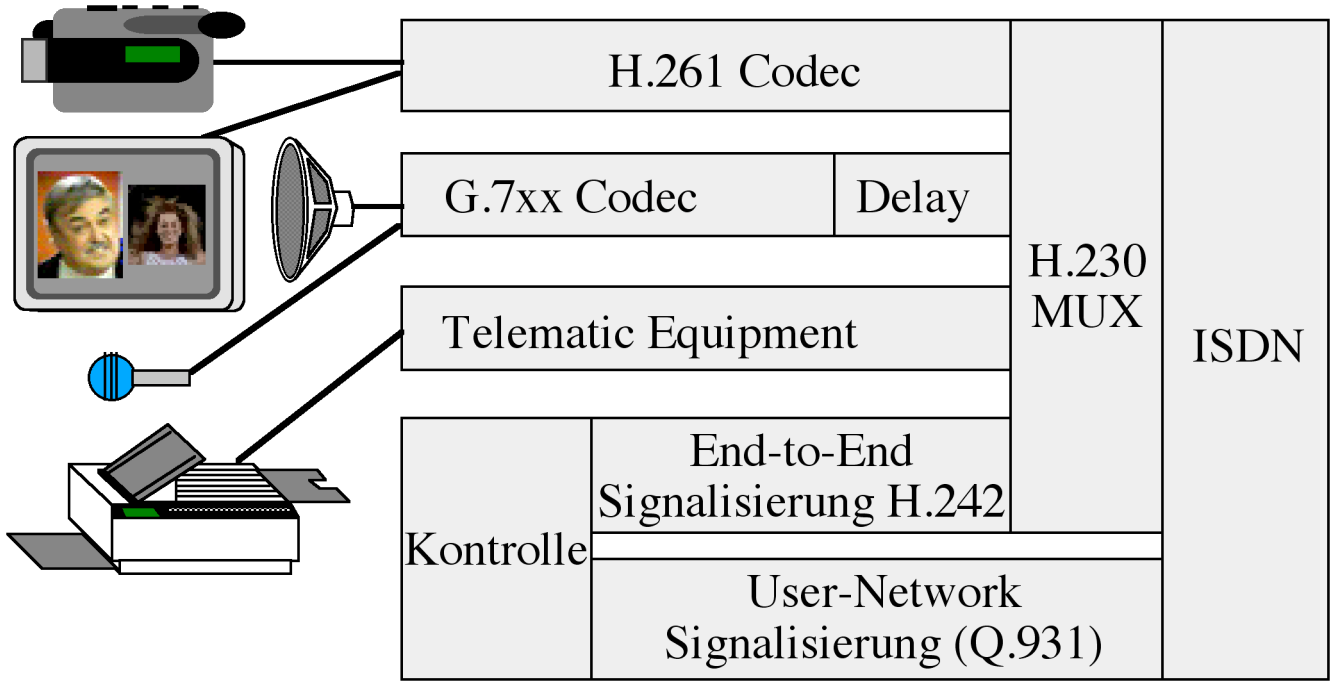
6.1.5 Kontrolle der Mehrpunkt-"Verbindung" im Mbone

- RTCP: RTP Control Protocol
  - Session-Kontrolle
  - Datenverteilung messen
- Skalierbarkeit der Session (Multicast)
  - Multicast von RTCP-Paketen
  - Sendezeit 'zufällig' wählen
  - Gesamtbandbreite limitieren
- Sender Report (SR)
  - Zeitstempel
  - Gesamtzahl Pakete und Bytes gesendet
- Receiver Report (RR)
  - Gesamtzahl Pakete empfangen
  - Gesamtzahl Pakete erwartet
  - Jitter
  - Letzter SR

- Source Description (SDES)
  - Source-ID
  - Canonical Endpoint Identifier: <User>@<DNS-name>
  - Name
  - E-Mail Adresse
  - Ort und Beschreibung (Prosa)
  - eventuell wiederholt
- Payload type mapping (FMT)
  - Source-ID
  - Typ 8 bit
  - Theoretischer Sample-Takt
  - IANA-ID (Internet Assigned Numbers Authority)
- Endpaket: BYE
  - Quelle beendet Sendung

6.1.6 H.320 Videotelefonie

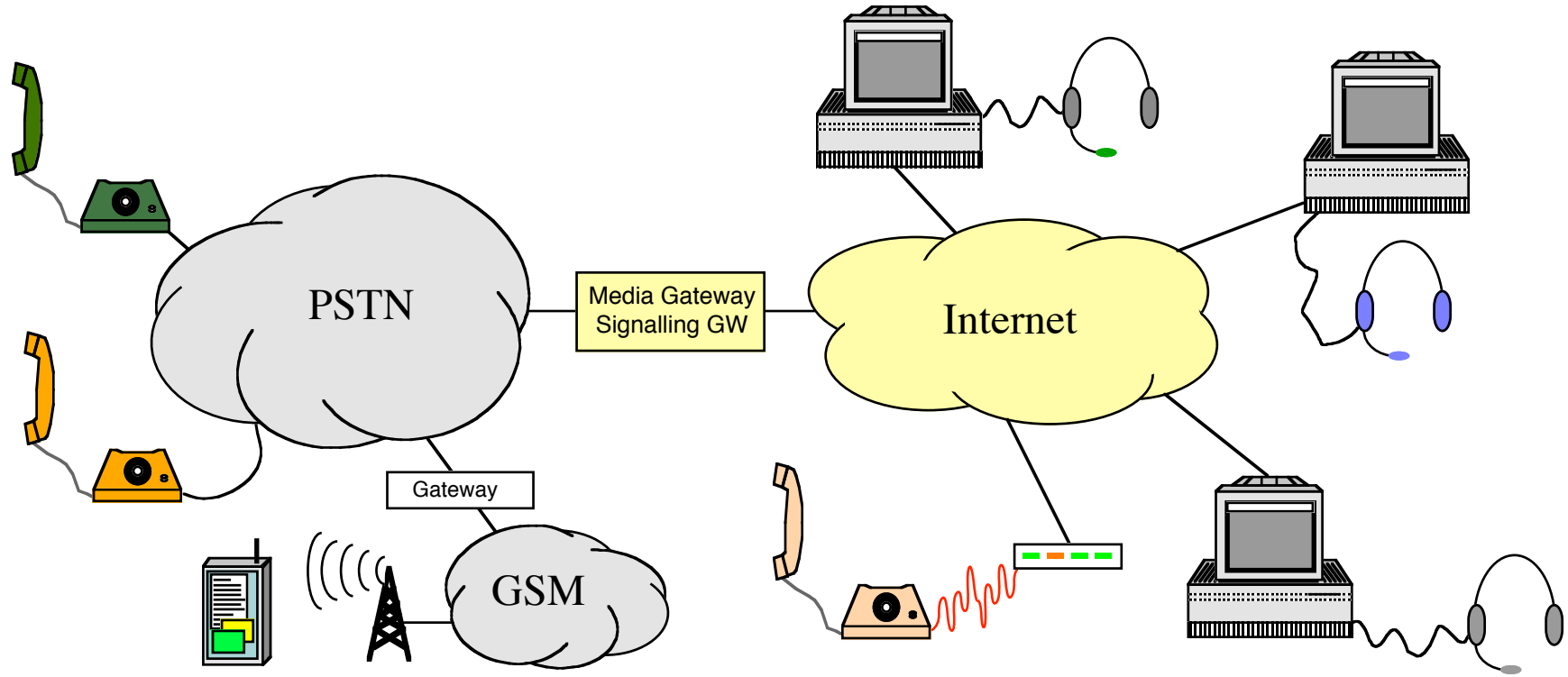
- ITU-Standard
- Zusammenfassung existierender Standards



- H.320 ist Rahmenspezifikation
  - End-to-End Signalisierung (H.242)
  - Multiplexen von Audio und Video: H.221
  - Multiplexen von Audio,Video und ITU-Daten: H.230
- Erfolgreich zur Verbindung von Konferenzstudios

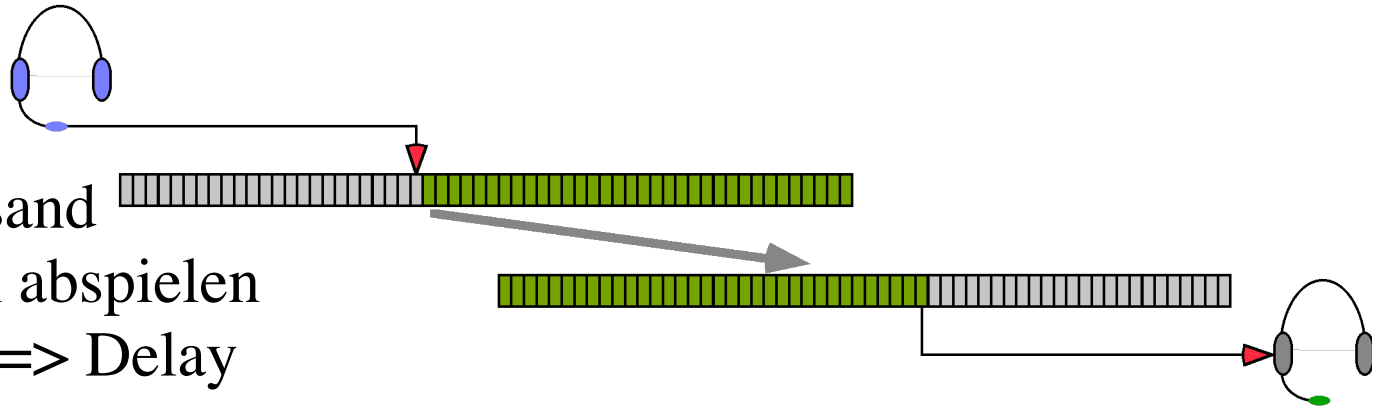
6.1.7 IP Telephony (Voice over IP, VoIP)

- Scenarien
  - Computer-Computer
  - Computer-POTS
  - Mobile-Computer



6.1.7.1 Audio im Internet

- Internet überträgt Pakete
  - Paket*größe* nur auf der Anschlußleitung relevant
  - im Backbone 'kostet' Paketanzahl



- Paketisierungsdelay
  - Paket füllen, dann Versand
  - Paket empfangen, dann abspielen
  - Anzahl Samples/Paket => Delay
- Beispiel Paketisierungsdelay: G.711+Ethernet
  - PCM + Ethernet: 8000 Samples/sec, 1500 Samples/Paket
  - Paketisierungsdelay: 187,5 msec
  - Roundtripdelay = 2* PakDelay + Internet-roundtrip + Endgerätedelay
  - München-Frankfurt: down 5301 kbit/s, up 509 kbit/s, ping 21 msec
  - München-Washington: down 5247 kbit/s, up 498 kbit/s, ping 120 msec
  - Roundtrip Frankfurt > 400 msec
  - Roundtrip Washington > 500 msec

- Audiokodierungen

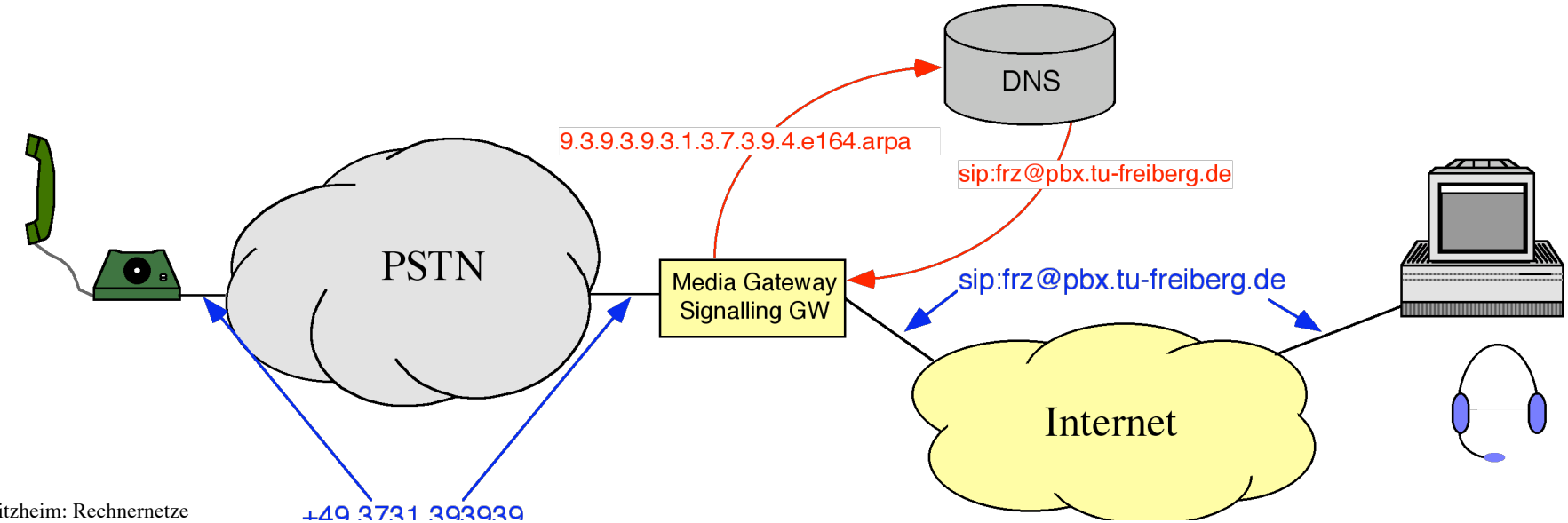
Verfahren	Qualität	kbit/sec	msec/Paket 500 Byte	Byte pro 20 msec
MPEG	sehr gut	192	21	500
ADPCM	mittel	32	125	80
ADPCM	Telefon	16	250	40
GSM	mäßig	13	307	32.5
hrGSM	schlecht	4	1000	10

- UDP zum Transport
  - Audio evtl. in RTP verpackt
- Verschlüsselung?
  - POTS: Provider verhindert Mithören
  - Teilnehmeranschlußleitung kann abgehört werden
  - GSM: Verschlüsselung im Funkkanal
  - im Internet könnte mitgehört werden
  - im LAN mithören einfach (auch bei Email, Web, Chat, …)
- Interface zum PSTN (Softswitch)
  - Signalling Gateway siehe unten
  - Media gateway: ADPCM-G.711 etc.

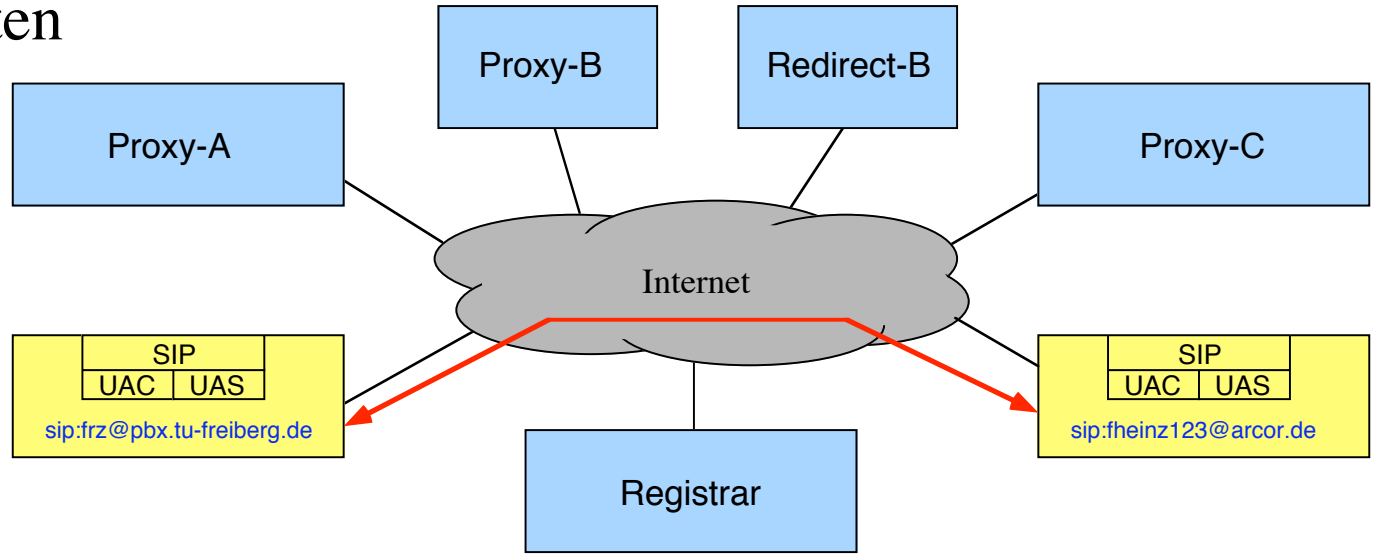
6.1.7.2 Verbindungskontrolle

- Adressierung
  - URI: sip:frz@pbx.tu-freiberg.de
  - Adresstabelle in den Endgeräten
  - Verzeichnisdienste (-> Skype)
  - E.164 NUmber Mapping
- ENUM
  - Abbildung E.164 -> URI
  - NAPTR (Name Authority Pointer) Record im DNS

```
$ORIGIN 9.3.9.3.9.3.1.3.7.3.9.4.e164.arpa.
IN NAPTR 100 10 "U" "E2U+sip" "!^.*$!sip:frz@pbx.tu-freiberg.de!i".
```



- Session Initiation Protocol SIP
  - RFC 2543, 3261
  - UDP oder TCP, Ports 5060, 5061
  - Session bzw. application layer
- Ähnlickeiten zu http
  - request - response
  - textbasiert, Felder wie RFC821 oder http: **`<name>:<wert>`**
  - Adressen in URIs: **`sip:frz@pbx.tu-freiberg.de`**
- User Agent im Endgerät: client und server
- SIP-Server
  - Proxy für öffentliche Dienste: Zugangskontrolle, ständig aktiv
  - Redirect Server: umleiten
  - Registrar
- Methoden
  - INVITE, BYE
  - ACK
  - REGISTER
  - OPTIONS, CANCEL



- Responses
  - 1xx: provisional: searching, ringing, …
  - 2xx: success
  - 3xx: redirect
  - 4xx, 5xx, 6xx: failures
- Invite-Message
  - Request-Typ
  - from und to wie email
  - tag: lokale call-id
  - call-id global eindeutig
  - Seqnr im Dialog
  - Leerzeile beendet header
  - Medienbeschreibung im Body
- Response
  - Status: response code
  - to und from vom request

```
INVITE sip:fheinz123@arcor.de SIP/2.0
From: KF <frz@pbx.tu-freiberg.de>; tag=12345678
To: Heinz F. <fheinz123@arcor.de>
Call-ID: 1234567890@frz.pbx.tu-freiberg.de
Cseq: 42 INVITE
Subject: Seminar

Content-Type: application/SDP
Content-Length: 124
```

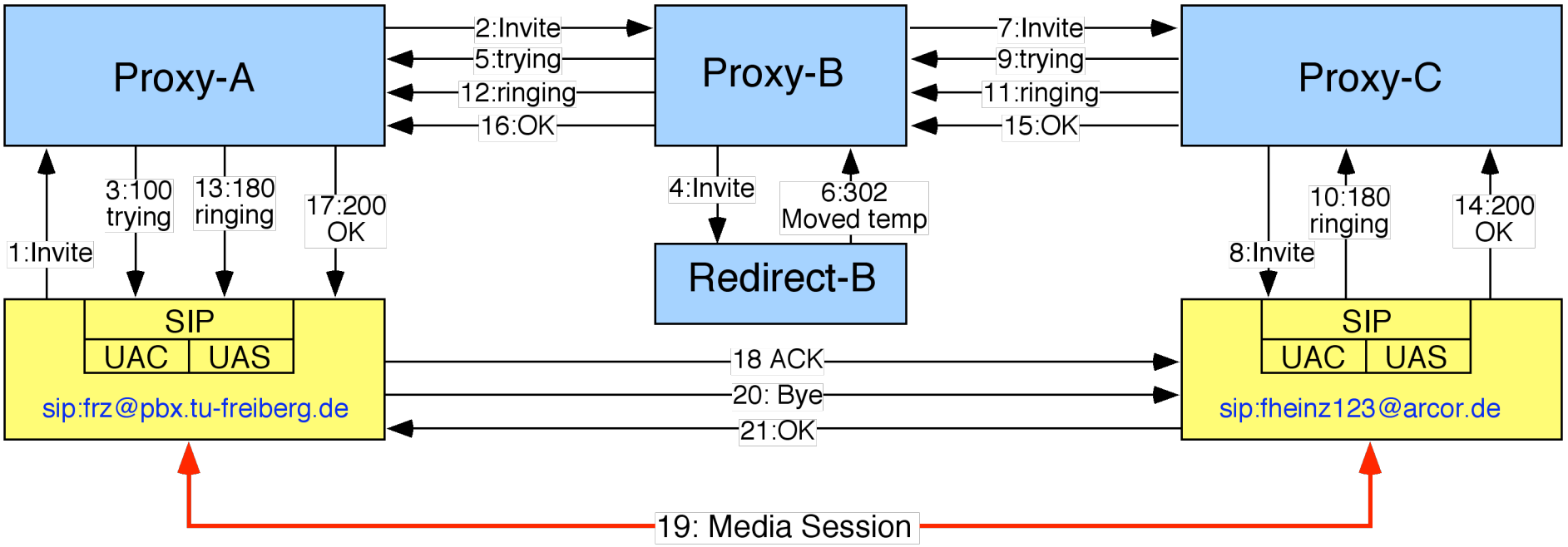
```
OK SIP/2.0
To: Heinz F. <fheinz123@arcor.de>; tag=AFFE1234
From: KF <frz@pbx.tu-freiberg.de>; tag=12345678
Call-ID: 1234567890@frz.pbx.tu-freiberg.de
Cseq: 42 INVITE

Content-Type: application/SDP
Content-Length: 160
nun kommt SDP, siehe unten
```

- Session Description Protocol SDP
  - ursprünglich mbone
  - Transport mit SIP, RTSP, Email+MIME, http
- Session Description
  - protocol version (v= …)
  - session name, information, URL
  - email, phone
  - connection information
  - bandwidth
  - timezone, encryption
  - Attribute (a=…)
- Time Description (NTP-time)
  - time active, repeat times
- Media Description

```
v=0
o=frz 1234567890 2890842807 IN IP4
10.47.16.5
s=SDP Seminar
i=A Seminar on …
u=http://www.example.com/seminars/sdp.pdf
e=frz@pbx.tu-freiberg.de (KF)
c=IN IP4 85.86.12.103
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
```

```
m=  (media name and transport address)
i=* (media title)
c=* (connection inf -- optional if included at session level)
b=* (zero or more bandwidth information lines)
k=* (encryption key)
a=* (zero or more media attribute lines)
```

• Signalisierungsablauf SIP

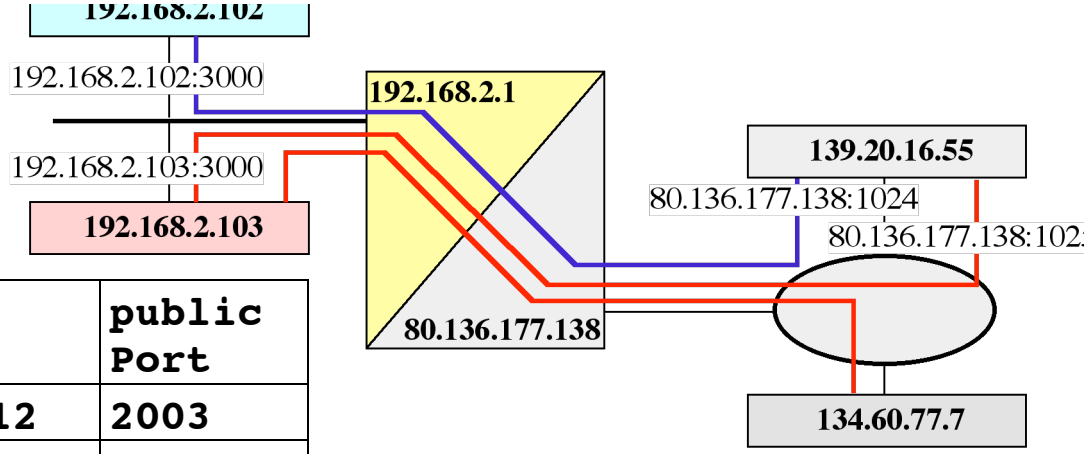


- Firewalls, NAT und andere Feinde
- Firewall
  - in Gateway, besonderem Rechner oder Arbeitsplatz
  - Paketfilter: store and possibly forward
  - stateful filter
  - Application layer filtering
- Regelbasiert
  - ankommend nur an registrierte (IP-Adr, Protocol,Port) weiterleiten
  - abgehend: z.B. Adressen blockieren

source addr	source Port	dest addr	dest port	action	
any	any	134.60.77.0	>1023	allow	incoming
134.60.77.0	any	any	any	allow	outgoing
any	any	134.60.77.5	any	allow	smtp-relay
any	any	any	any	deny	

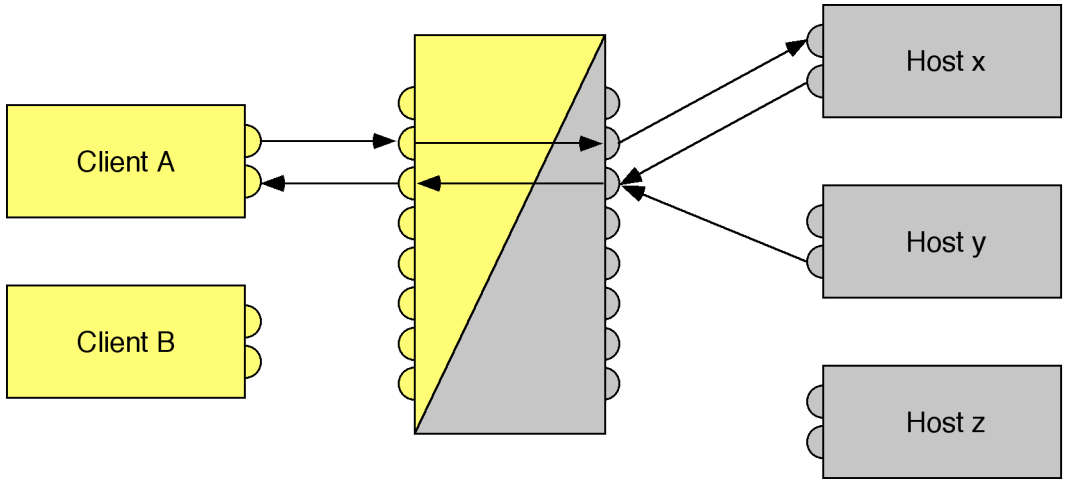
- Stateful
  - TCP threeway-handshake erlauben/blockieren
  - alle Pakete von zugelassenen Verbindungen schnell durchlassen
  - inbound und outbound
  - auch UDP-Ströme identifizieren

- Network Adress Translation
  - Router mit Adressübersetzung
  - lokale IP, globale IP

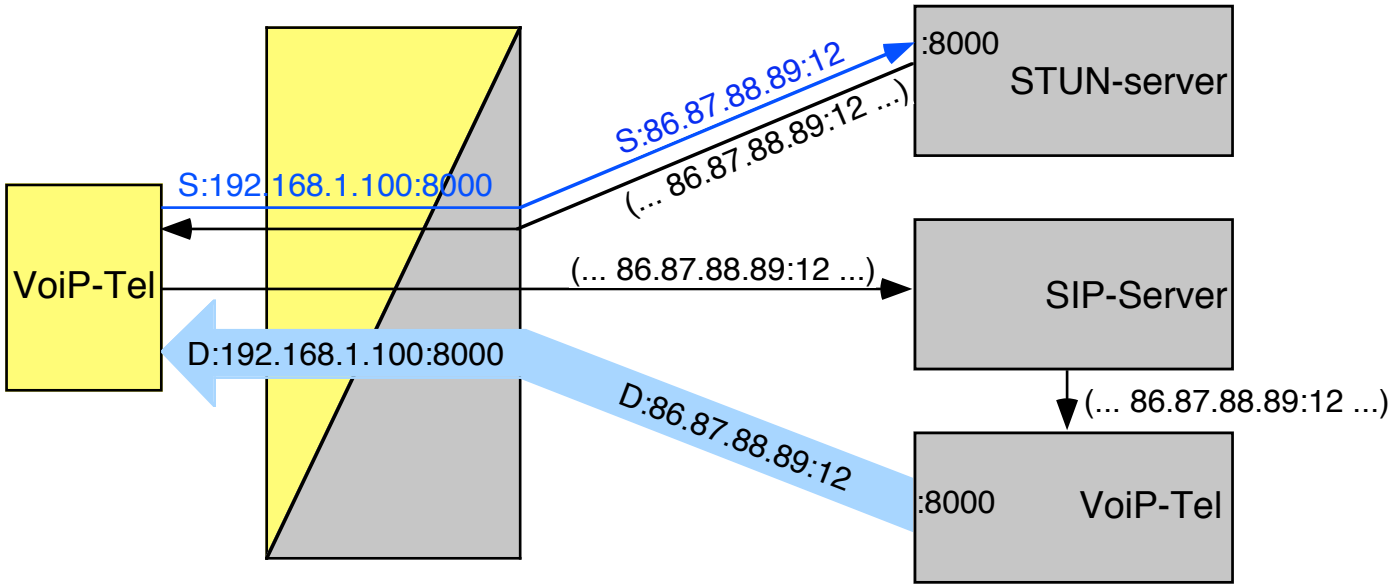


local IP	local Port	public IP	public Port
192.168.3.100	1777	88.87.169.12	2003
192.168.3.100	45054	88.87.169.12	2004
192.168.3.100	60123	88.87.169.13	2000
192.168.3.101	1777	88.87.169.12	2006
192.168.3.102	7112	88.87.169.12	2007
…			

- Problem kommende Anrufe
  - wie findet NAT die angerufene lokale IP+Port (SIP und Media)
  - wie sagt man dem Firewall: "durchlassen bitte"?
- NAT-Strategien
  - full cone
  - address restricted cone
  - port-restricted cone
  - symmetric



- Applikationsspezifische Lösungen (➔ Skype)
- Applikation Layer Gateway
- Simple Traversal of UDP through NATs (STUN)
  - RFC 3489, 5389
  - Client in der App, STUN-Server
  - Requests zum Server mit 2 bekannte Ports
  - Server antwortet mit öffentlicher (IP, Port) in der Nutzlast
  - full cone: alle anderen können (IP, Port) auch verwenden



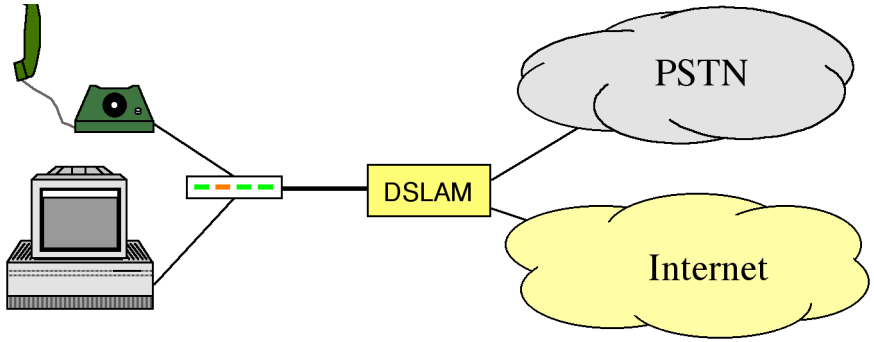
- NAT Classification Algorithm
  - testet ob Firewall oder Nat
  - 2 Echo-Server

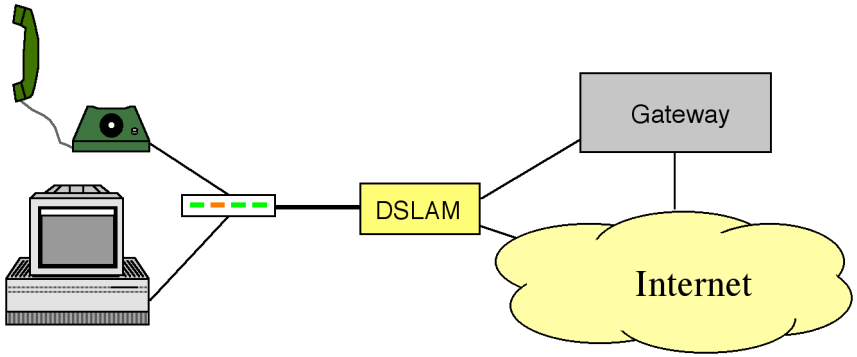
```
if (!request_echo(S1,IP1,port1,resIP,resPort)) return UDP_blocked;
else if (resIP == pubIP) /* no NAT, firewall? */
        if (request_echo(S1,IP2,port2,resIP,resPort)) return clear;
        else return symm_Firewall;
     else /* NAT … */
     if (request_echo(S1,IP2,port2,resIP,resPort)) return fc_nat;
        else
        { (request_echo(S2,IP1,port1,resIP_2,resPort_2));
          if (resIP != resIP_2) return symm_Nat;
          else if (request_echo(S1,IP2,port2,resIP,resPort))
               return rest_cone_nat;
               else return rest_port_nat;
        }
```

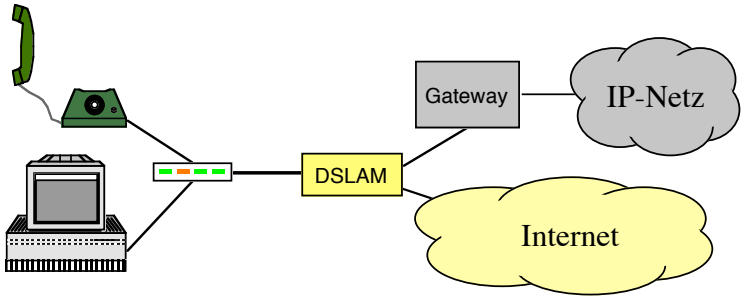
- Interactive Connectivity Establishment: ICE

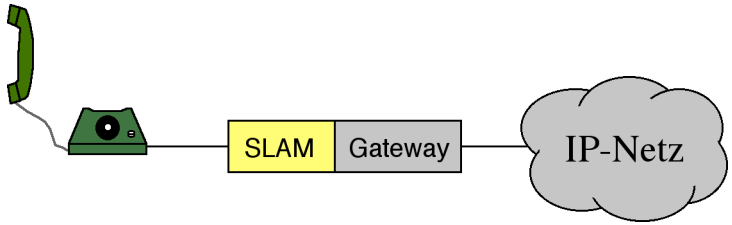
6.1.7.3 Bespiele

- DIY
  - Applikation an beiden Geräten
  - IP-Nummern in Verzeichnis
  - oder: SIP-Server
  - evtl. POTS-Gateway
- Skype
  - peer-to-peer Modell (-> Kazaa)
  - Verschlüsselung
  - eigenes NAT/Firewall Traversal System
- Providerbasiert
  - PC oder Telefon am Adapter
- im Backbone der Telefongesellschaft
  - SLAM: Subscriber Line Access Multiplexer (… Module)
  - analoges oder ISDN-Telefon
  - im SLAM weiterleiten an POTS, ISDN, VoIP

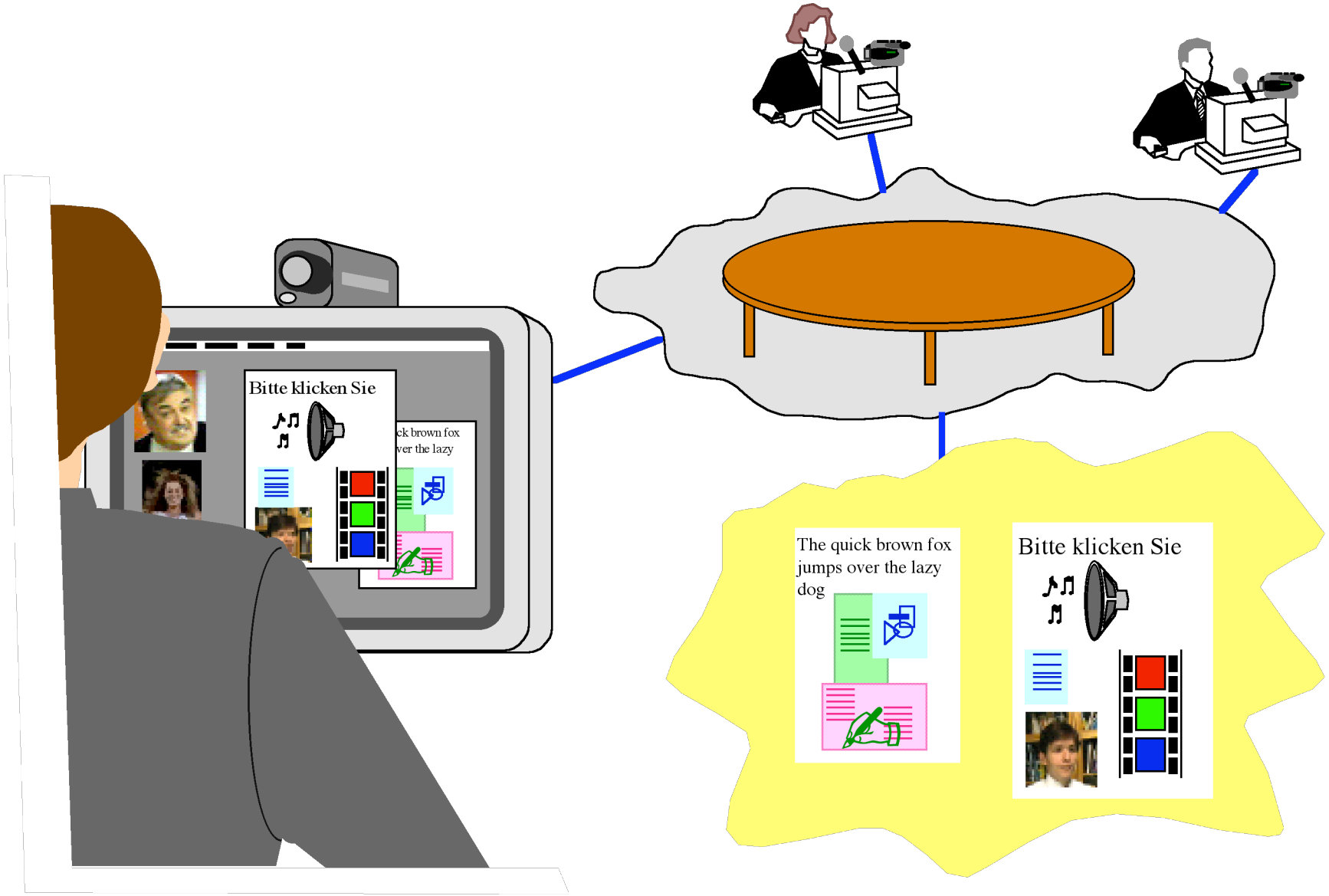






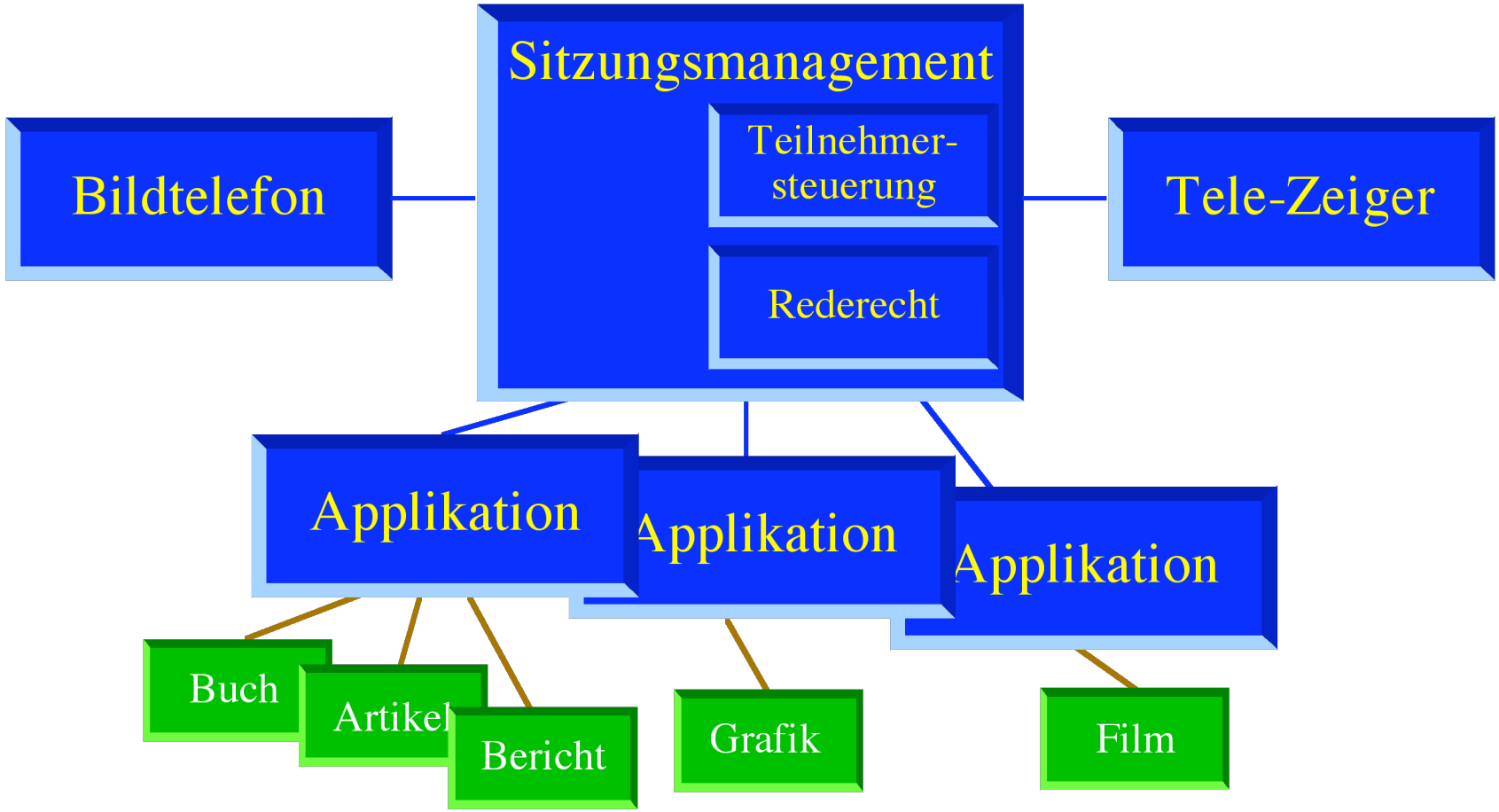


6.2 Telepräsenz



- 1 Teilnehmer
  - Telearbeit: Arbeit von 'zu Hause'
  - Fernwartung
  - Telnet etc.
  - Übertragen von Dokumenten
  - oder: Fernbedienung des Arbeitsplatzrechners
  - Screensharing: Timbuktu
  - Telefon und Videotelefon zur normalen Kontaktaufnahme
- n Teilnehmer: Telekonferenz
  - Sitzungen ohne Reisen (Reisezeit >> Sitzungszeit)
  - z.B. shared Whiteboard
  - gemeinsame Dokumentenbearbeitung
  - automatisches Protokoll

- Dokumentenbearbeitung
- Kooperationsunterstützung (Telefon + …)
- Komponenten



- Kooperationsfähige Applikation (cooperation-aware)
  - Spezialsysteme für Konferenzräume
  - Verteilter Editor

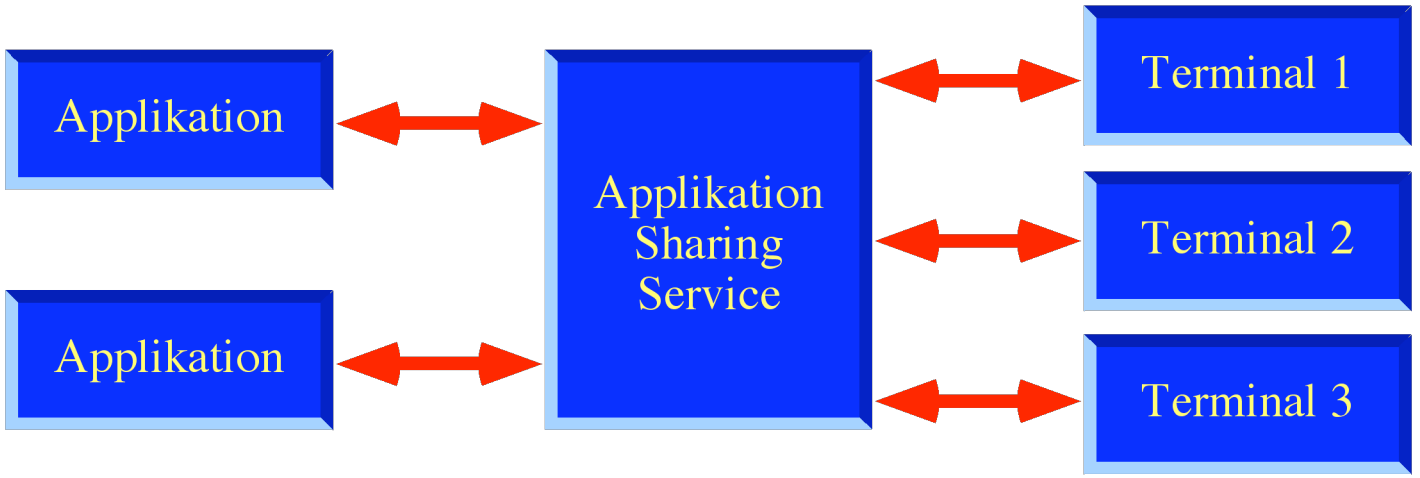
	Konferenzsystem	Benutzer
Hardware	homogen	heterogen
Betriebssystem	homogen	heterogen
Applikation	Groupware	Einzelplatz

- Applikation nicht kooperationsfähig (cooperation-unaware)
  - Standardprogramme
  - Word, FrameMaker, Corel-Draw, Premiere, …

- Kommunikation zwischen Anwendung und Terminal
- Ausgabe
  - Grafikausgabe
  - Tastaturlampen, …
  - Video
  - Audio (Systembeep!)
- Eingabe
  - Maus
  - Tastatur
  - Modifier-Tasten
  - Mikrofon
  - Kamera
- Metaeingaben
  - vom Betriebssystem
  - Suspend/Resume
  - Fensterinhalt sichtbar / unsichtbar

6.2.1 Applikation-Sharing

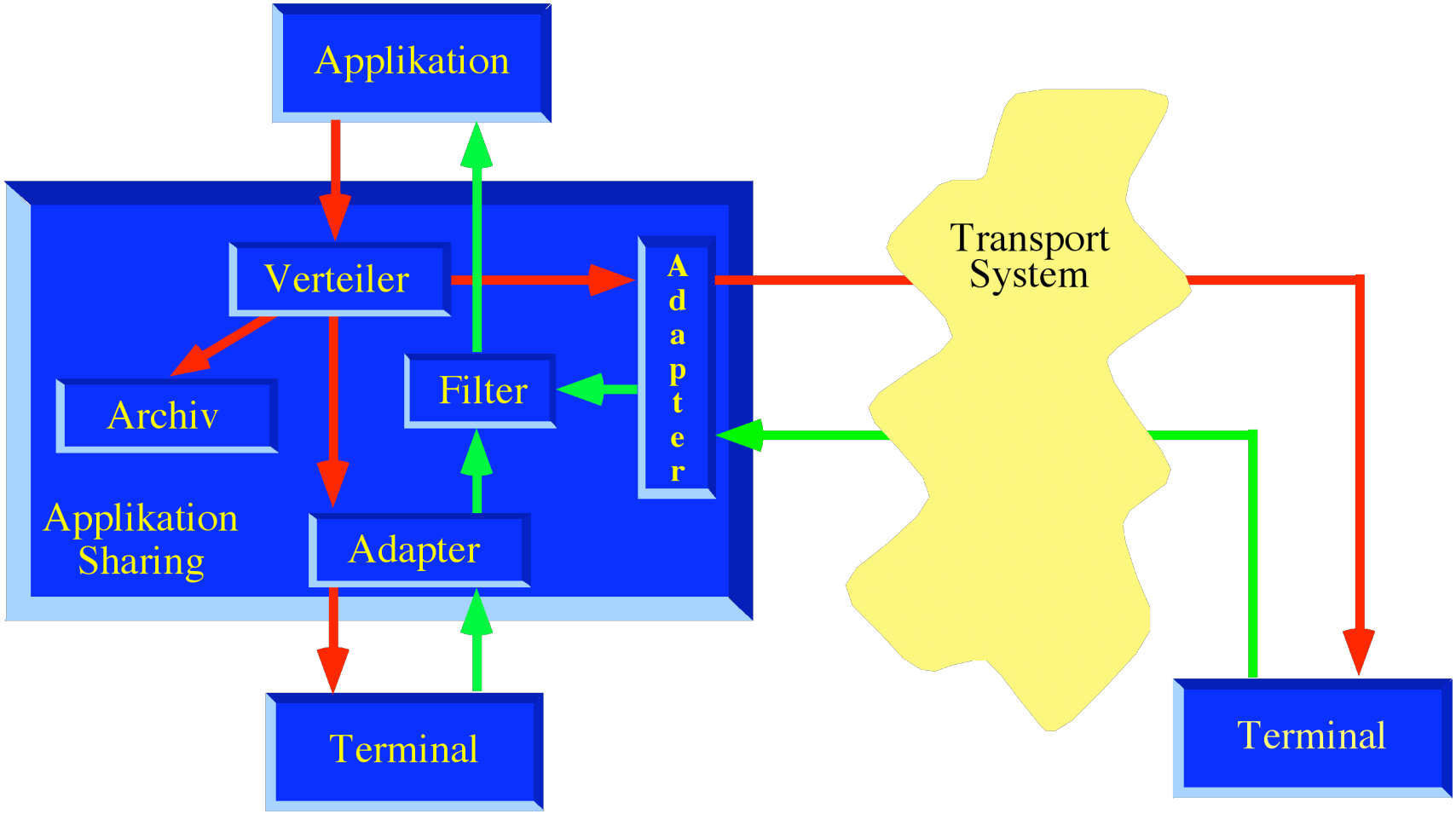
• Dienst: Vermittlung zwischen Programm und Terminal



- Timbuktu [Epard; 1985]

• Multimedia-Application Sharing [Dermler, Froitzheim; 1992]
- z.B. Präsentationen, Filmschnitt, Bildbearbeitung
- Vermittlung zwischen Programm und Gerät

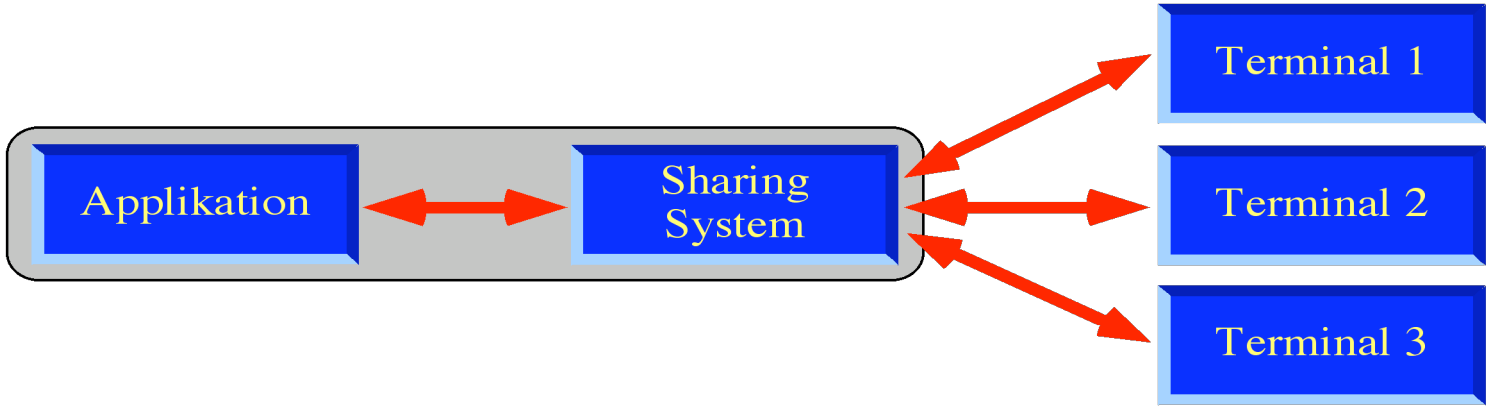
- Funktionen des Sharing - Dienstes (homogen)



6.2.2 Sharingkonzepte

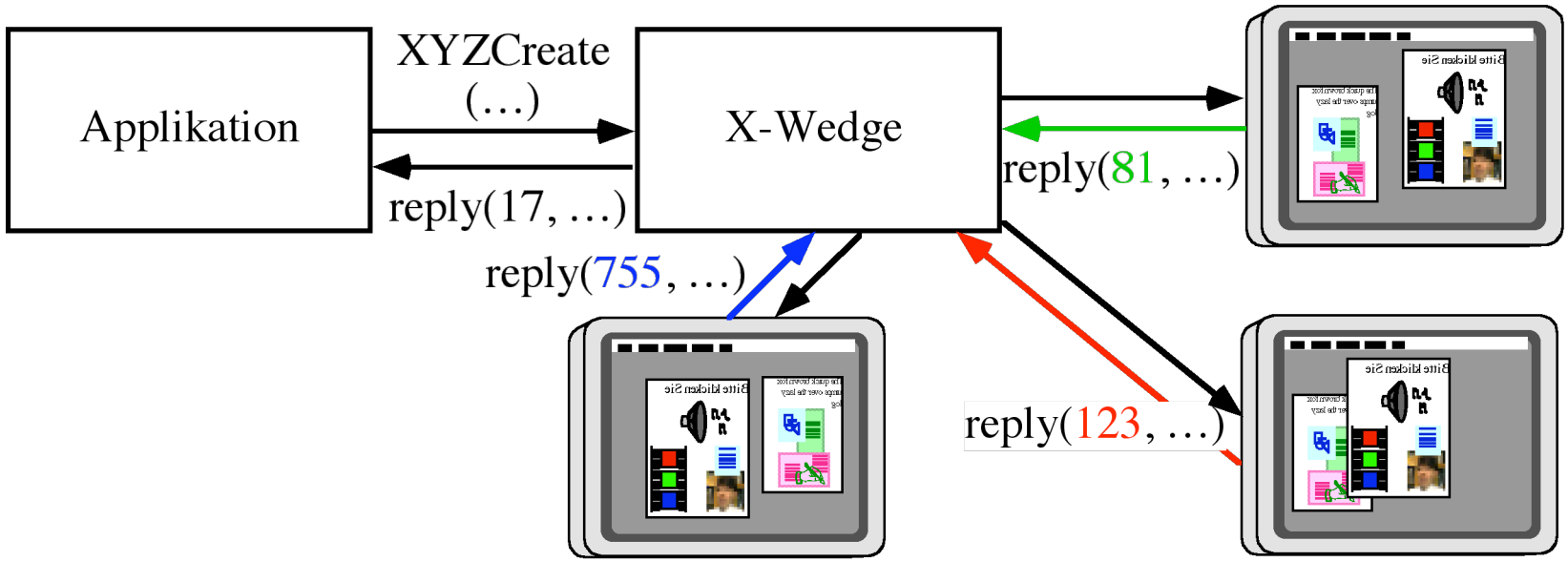
6.2.2.1 Verteilung der Grafikinformation

• Zentrale Architektur



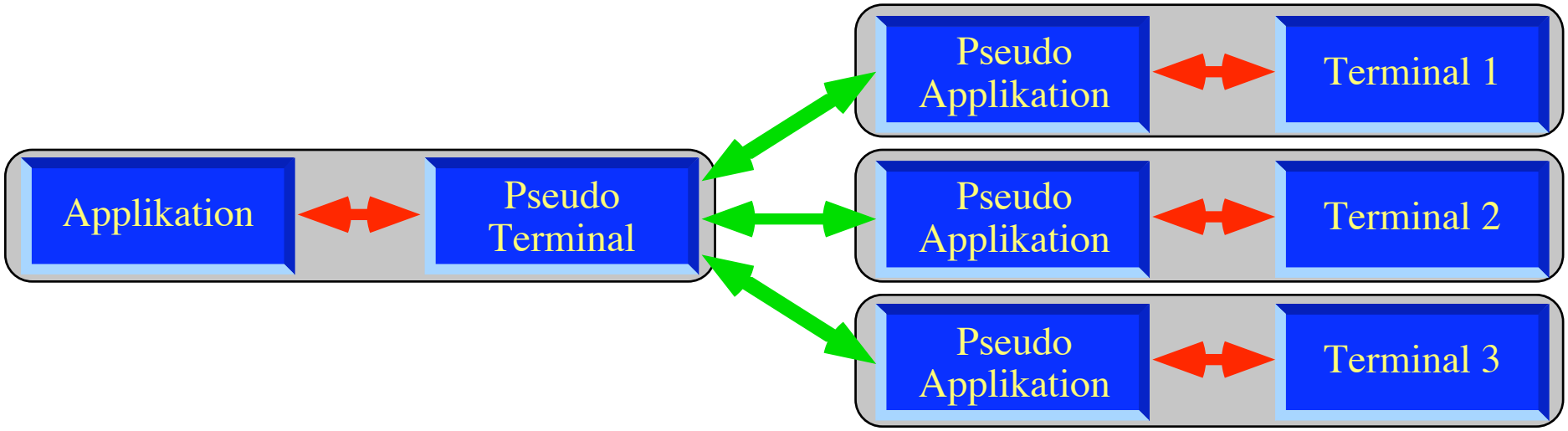
- Standardprotokolle
- Besonders geeignet für X Window
- XTV [Abdel-Wahab], SharedX [Altenhofen]

- Aufgaben der X-Wedge
- ResourceIds managen
  - eindeutig zwischen Client und Server
  - 'Pool'-Konzept

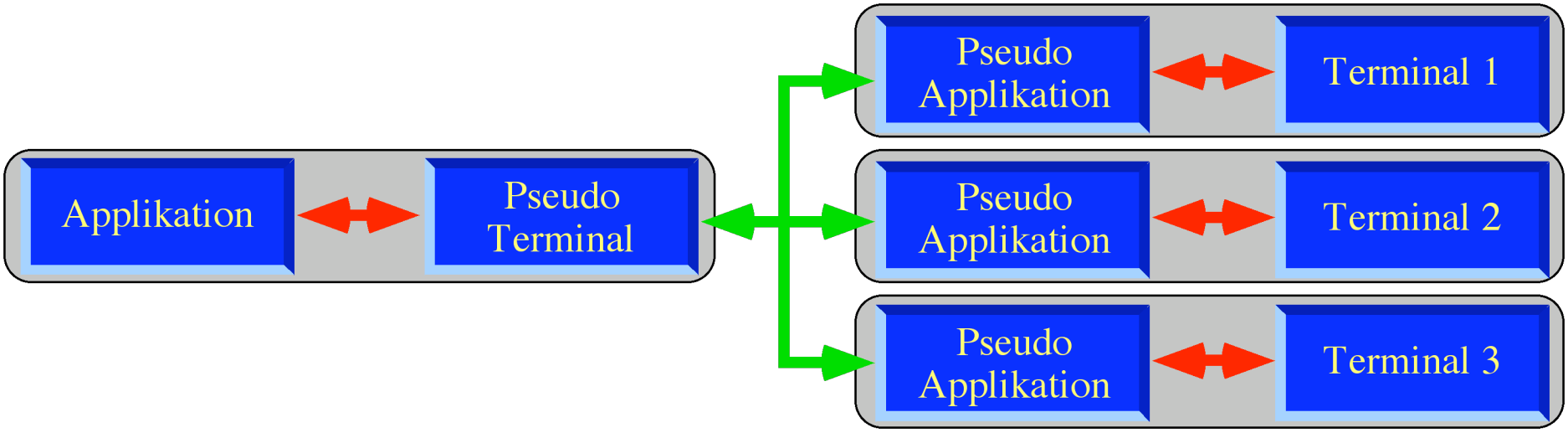


- Einheitliches Imaging-Modell
  - Farben
  - Fonts
- Bit-Order und Byte-Order
- Fensterüberlappung und Redraw konsolidieren

- Verteilte Architektur

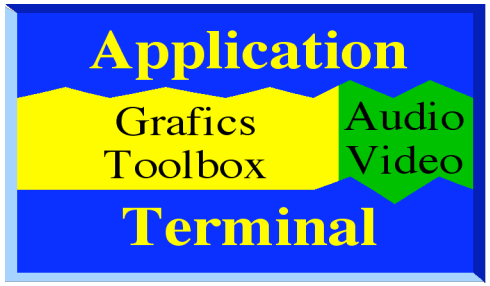


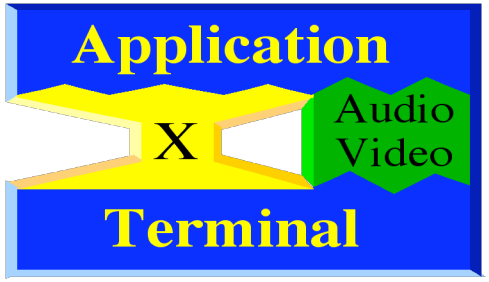
Eigene Verbindung zwischen Verteilungs-Komponenten
- 'eigene' Verbindung: Multicast, Verschlüsselung, Kompression
- X-Wedge [Gutekunst, CIO-JVTOS; 1994]



6.2.2.2 Redirector

- Abhören von Grafikoperationen
- Events einfügen (Tastatur, Maus, Verwaltung)
- Interface Applikation-Toolbox-Terminal

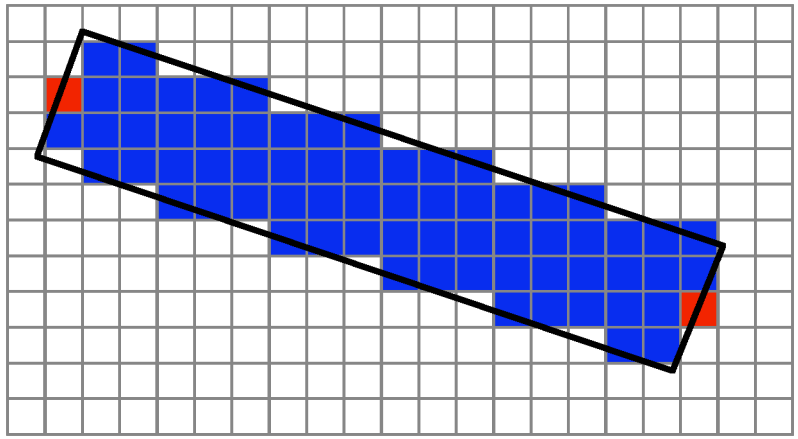




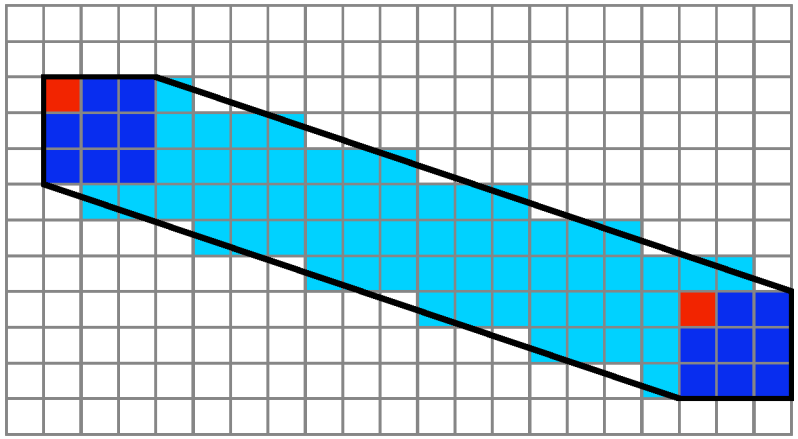
- Betriebssystemkomponente simulieren
  - toolbox
  - transport system
  - device driver
  - memory
  - low-intercept vs. high-intercept
  - Datenmenge vs. Prozedurmenge
  - Semantik

6.2.2.3 Übersetzung

- Grafikprimitive auf Sequenzen von Funktionen des Ziel-Grafiksystemes abbilden
- Funktionale und semantische Lücken

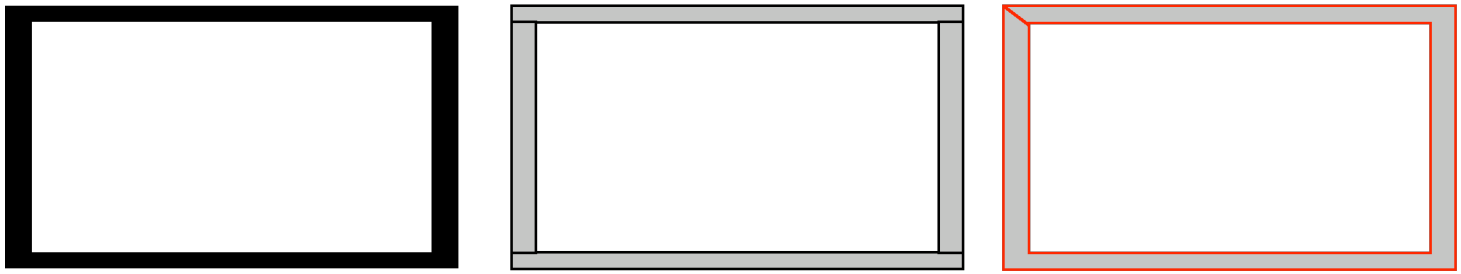


X-Window

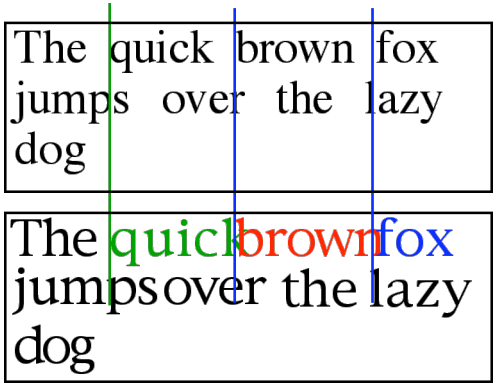


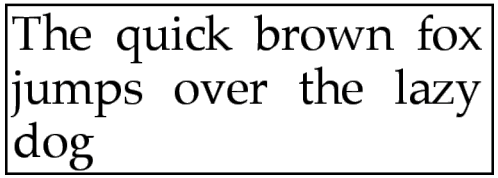
QuickDraw

- nicht-quadratische Stifte



- Text
- Zeichensatzanpassung
  - Ä, Ö, Ü und ihre ASCII-Werte
  - Übersetzungstabelle
  - fehlende Zeichen (□)
- Fonteigenschaften
  - Zeichenhöhe
  - Zeichenbreite
  - Randausgleich

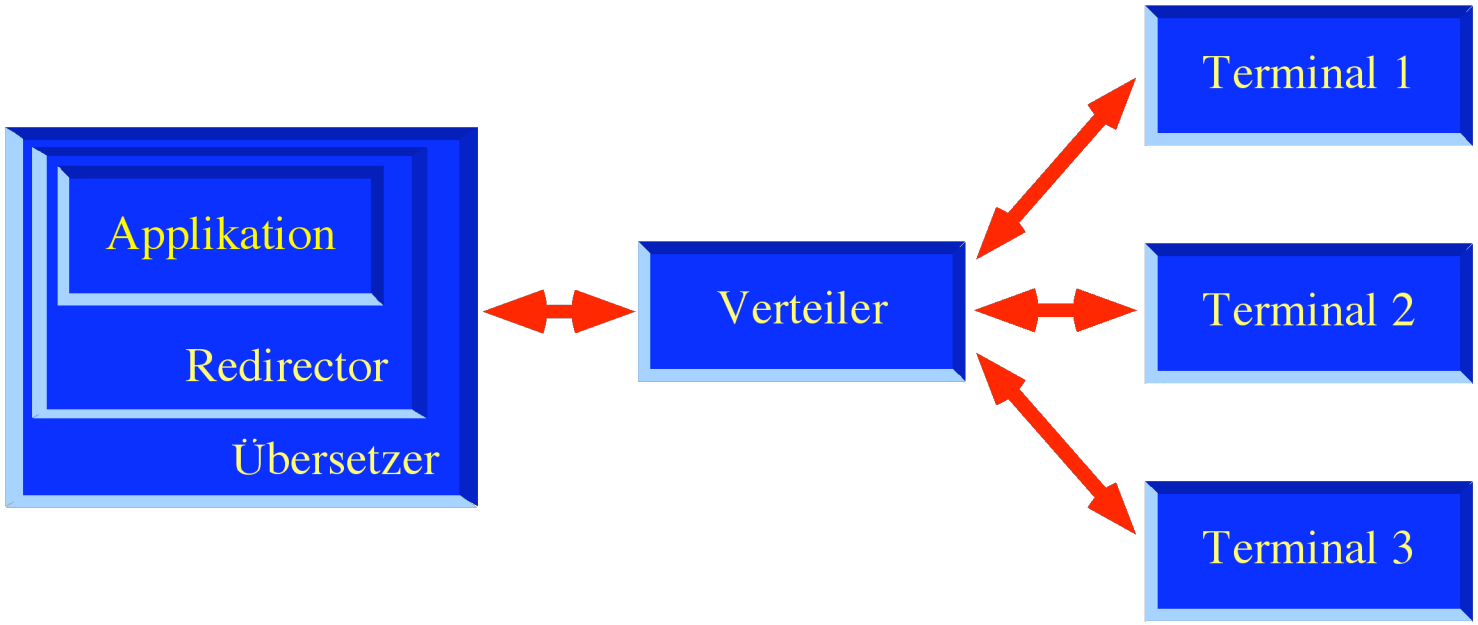




6.2.2.4 Anordnung der Komponenten

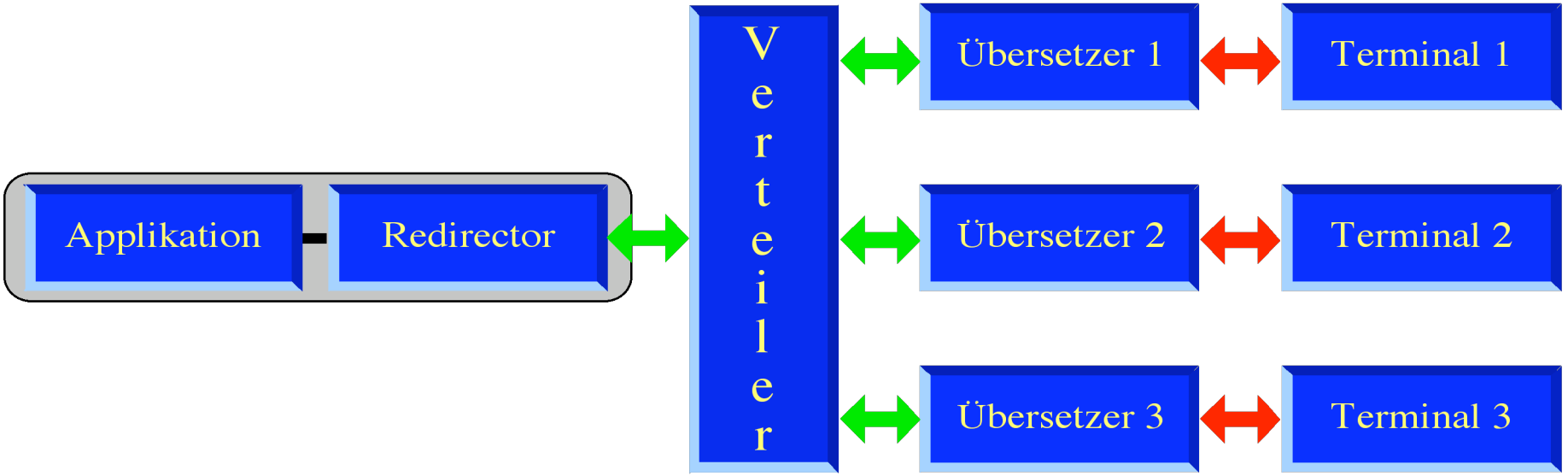
• Kombination mit Verteilung

Vor der Verteilung



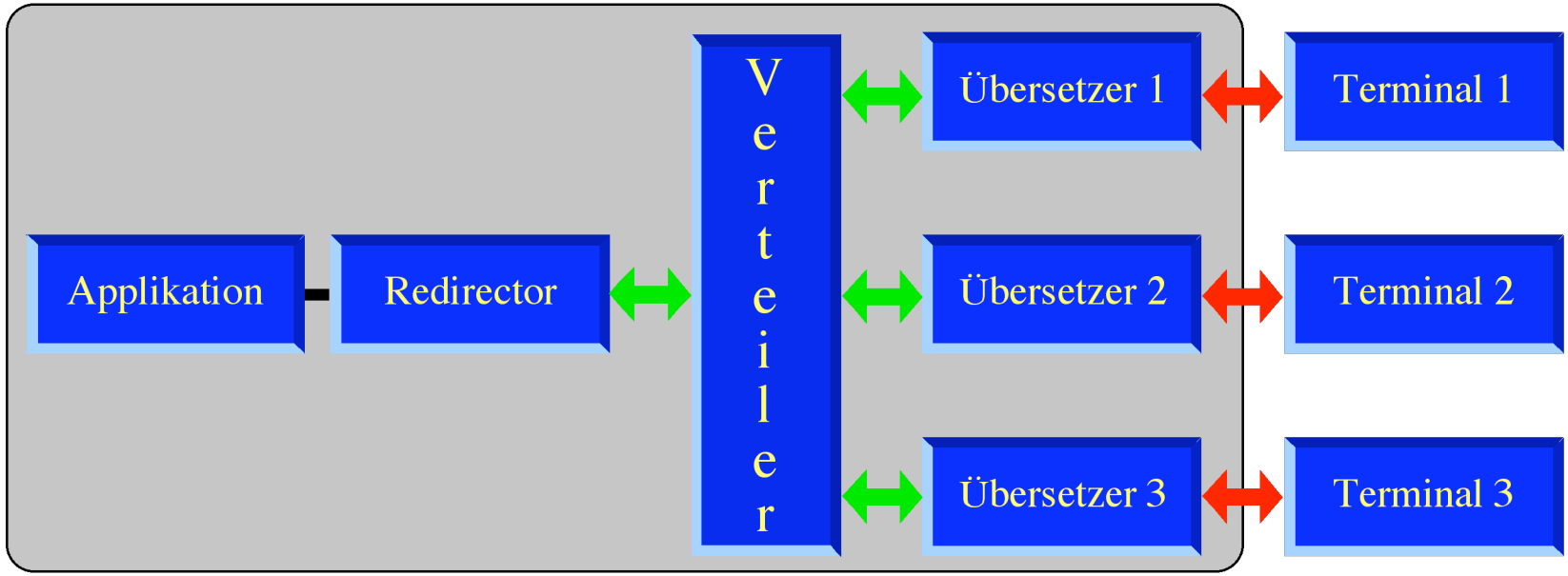
+ Komponentenwiederverwendung (QuiX + X-Wedge)
– Gleichbehandlung der Terminale (Farbtiefe, Zeichensätze, …)
– Struktur unflexibel

- Übersetzung nach der Verteilung



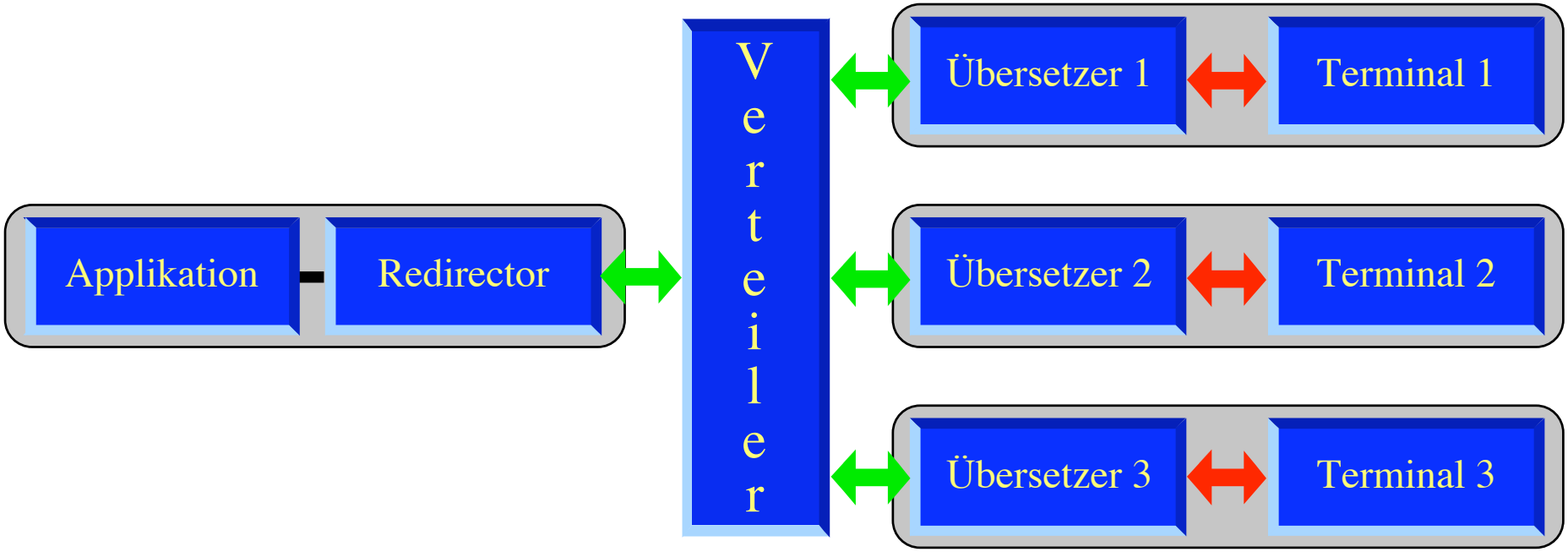
+ Flexible Verteilung der Funktionen
+ Konverter gut anpassbar an Terminaleigenschaften
  QuickDrawServer statt X-Server
  GDIServer statt X-Server
+ Objektorientierter Ansatz macht Verteiler einfach
– Mehrfache Übersetzung

- QuiX-M [Froitzheim, Wolf; 1995]



- Migration vom einfachen QuiX
  - Instanziierung des Übersetzers
  - Verteiler bietet nach oben Übersetzerschnittstelle
  - Seiteneingang zur Verteilungs-Steuerung
- Performanceprobleme
  - X-Server
  - Übertragung, Pufferstrategie, etc.
  - ab 1:4 Übersetzung

- Ausgelagerter Übersetzer: QuiQ [Maier, 1997]
  - Übersetzung von QuickDraw nach X
  - Zeichen von QuickDraw-Grafikkommandos mit der Xlib



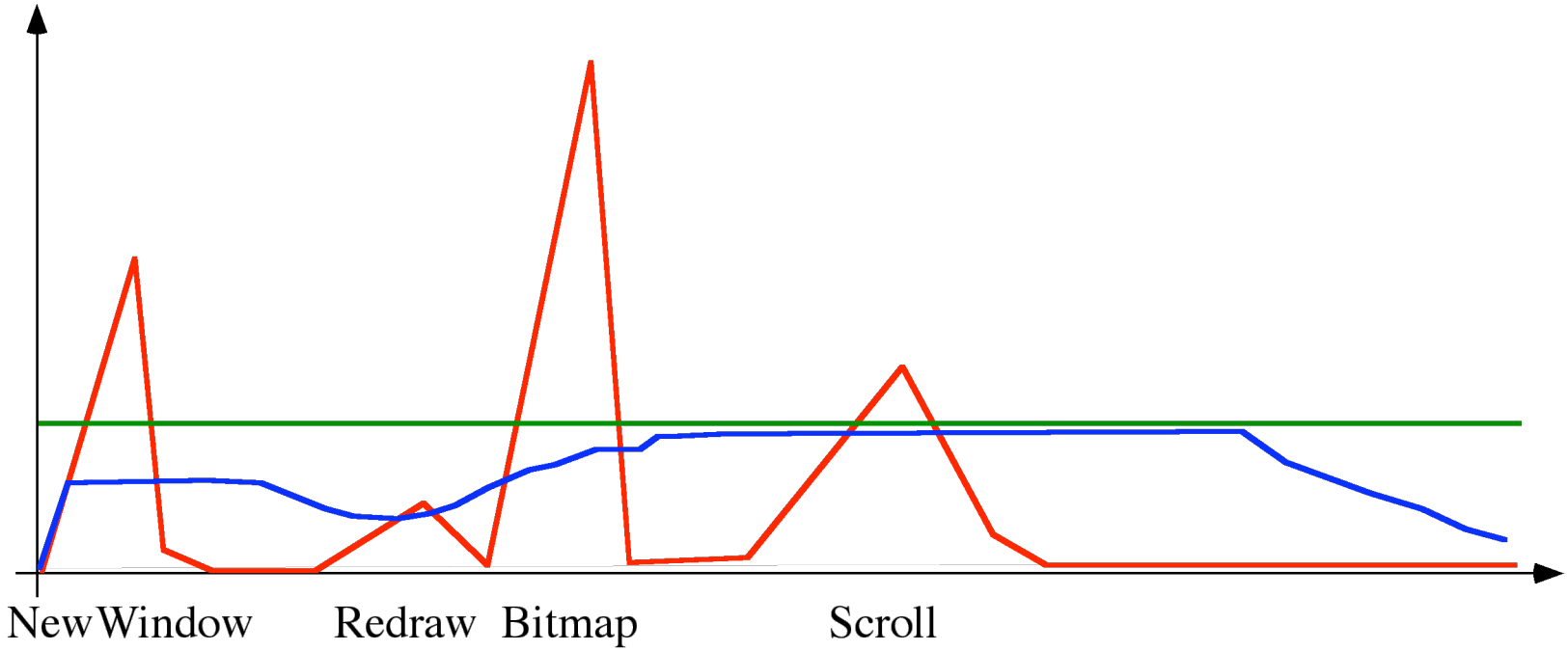
- Mehrpunktscenario
  - MBone, UDP
  - einfache Fehlerkorrektur (r=0,5) mit Bitpaar-Manipulation

    Q1 = P1; Q2 = P2;

    Q3 = P1+P2; (+ aus XOR)

    Q4 = P1 • P2 (• aus + und *; * Tabelle)

- Traffic-Shaping notwendig

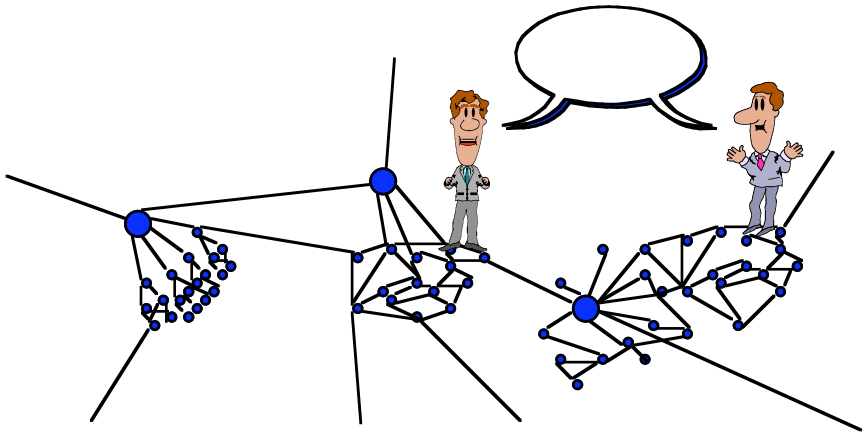


- Anständige Programme
  - Adobe PhotoShop, Premiere, Tools und Utilities
- Feinde
  - Menus, Dialogboxen, Popup-Menus, Scroller

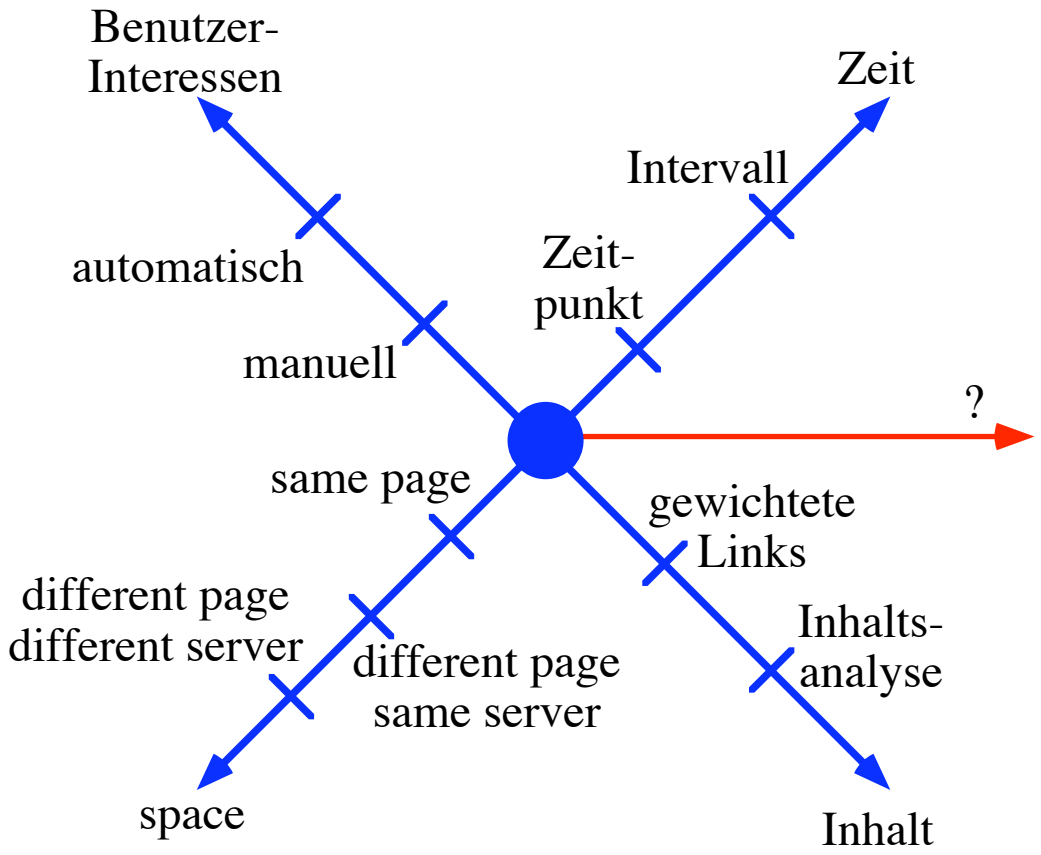
  => Redirection an der Toolboxschnittstelle
  - DIY-Toolbox (Microsoft, Claris)
  - direkter Aufruf von OS-Routinen

6.3.2 Virtuelle Präsenz

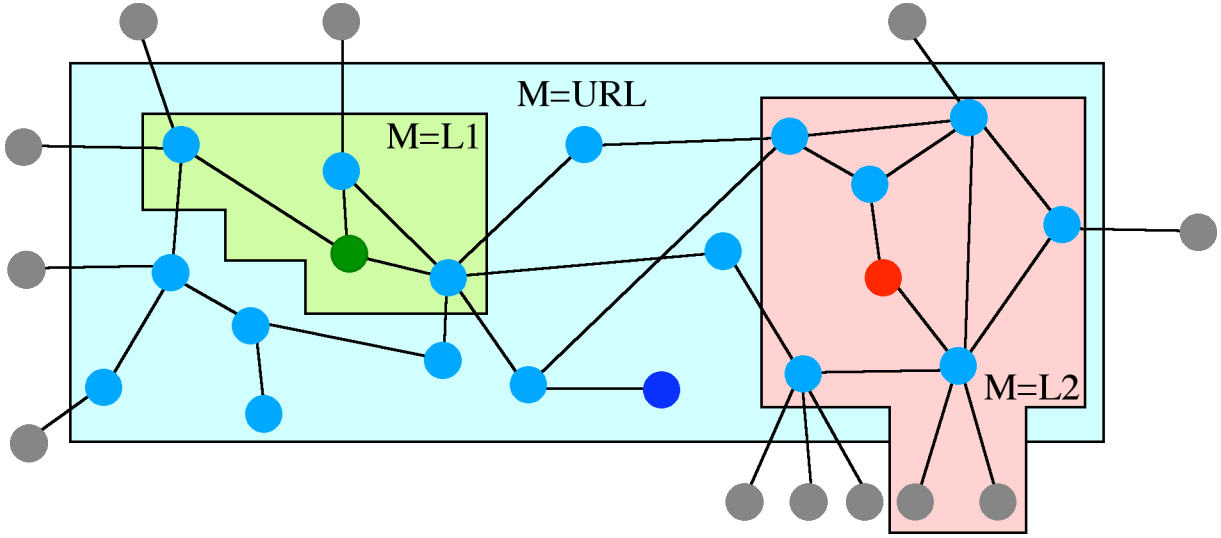
- WWW ist ein chaotischer Informationsraum
  - Benutzer allein
  - Suchmaschinen und Linkpages
- Kontakte herstellen
  - andere Brower sichtbar machen
  - Treffpunkte eröffnen
- Konferenzen eröffnen
  - Audio, Video, Application-Sharing, Chat, …
  - Co-Browsing
- Neues UI für Konferenzdienste
- Orthogonal zu Suchmaschinen



- Treffen
  - auf derselben Seite
  - auf (inhaltlich) benachbarten Seiten
  - Zeitspanne limitiert
- Vicinity
- Metriken

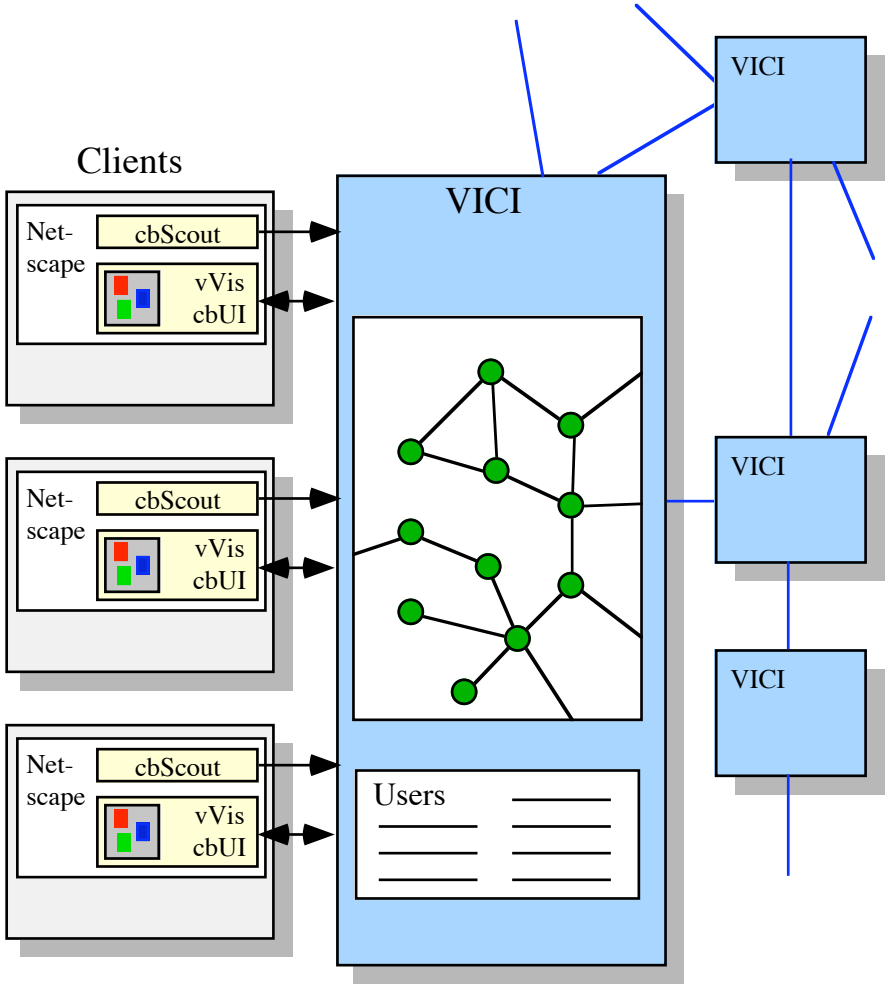


- URL-basierte Metriken
  - Schluß vom Aufenthaltsort des Benutzers auf Interessen des Benutzers
  - space: Linkdistanz < r
  - Inhaltsbezug: Gewichtete Links: $\sum w_i < r^*$;
    Contentmatching: $|C_i - C_k| < r'$
  - Auswertung mit Karte des WWW

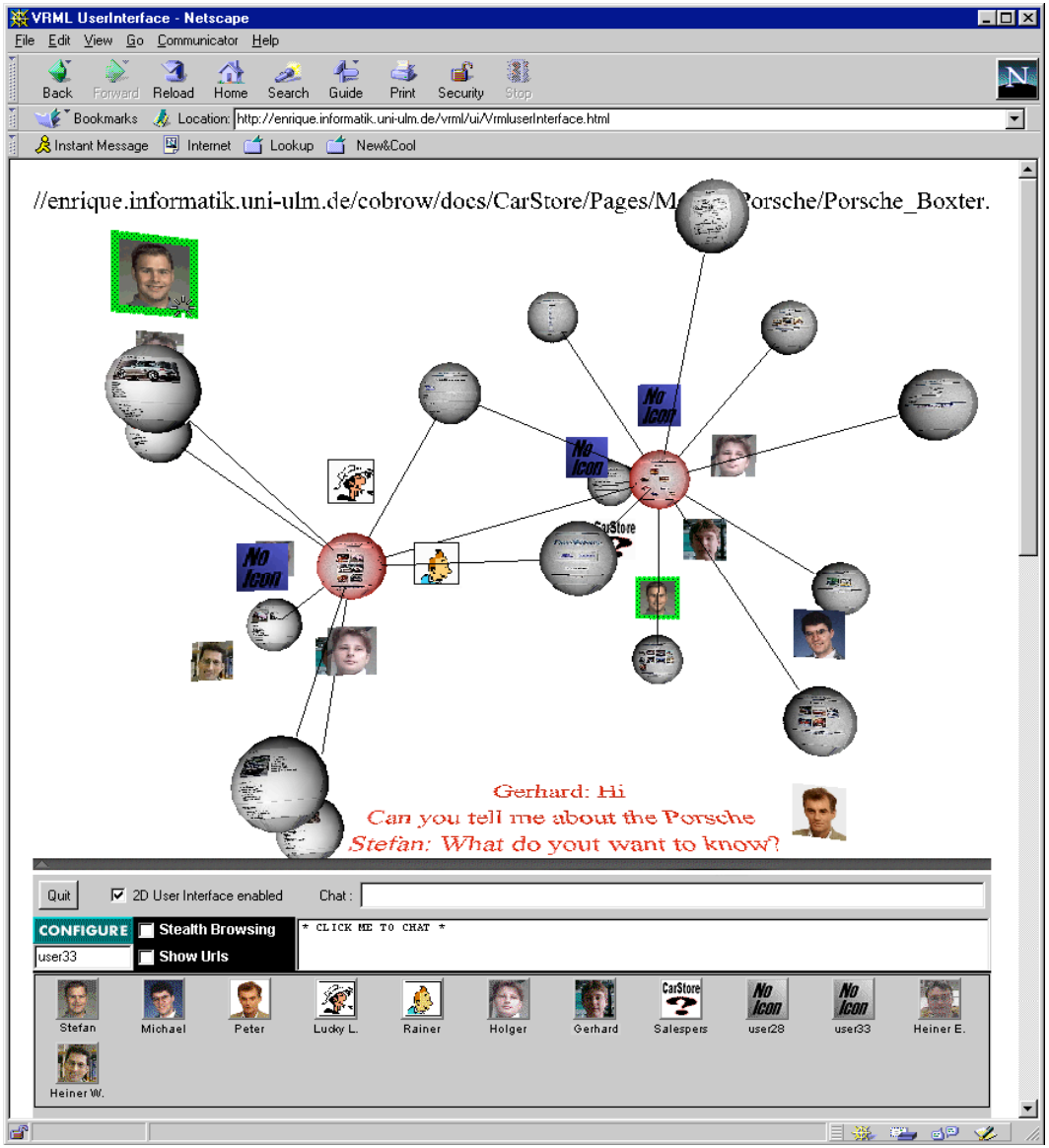


- Benutzerbasierte Metriken
  - Benutzerinteressen im CoBrow-Server registrieren
  - manuell eingeben oder/und automatisch ermitteln (Browserfilter)
  - Profile 'matchen'

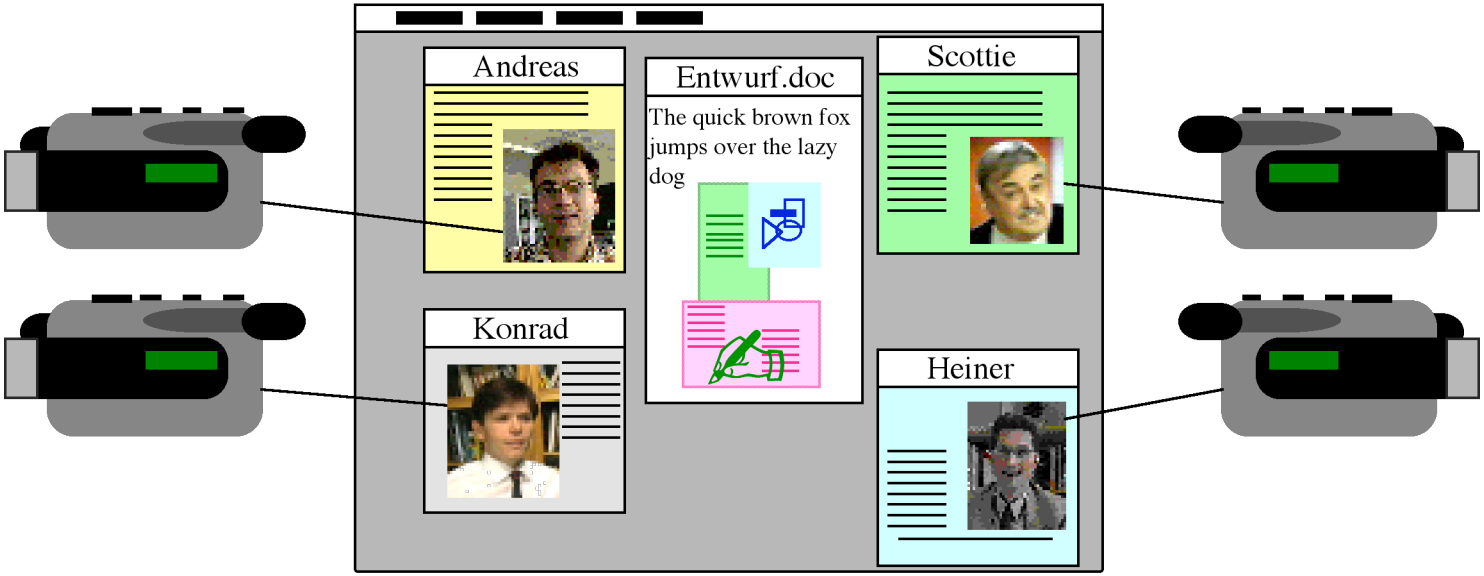
- Vicinity-Server
  - user-Position feststellen
  - Nachbarschaft berechnen
  - metrics: space, time, content
  - verteiltes System
- cbScout
  - Applet
  - hält Verbindung: endloses GIF
  - meldet betrachtete Seiten
  - alternativ: JavaScripts



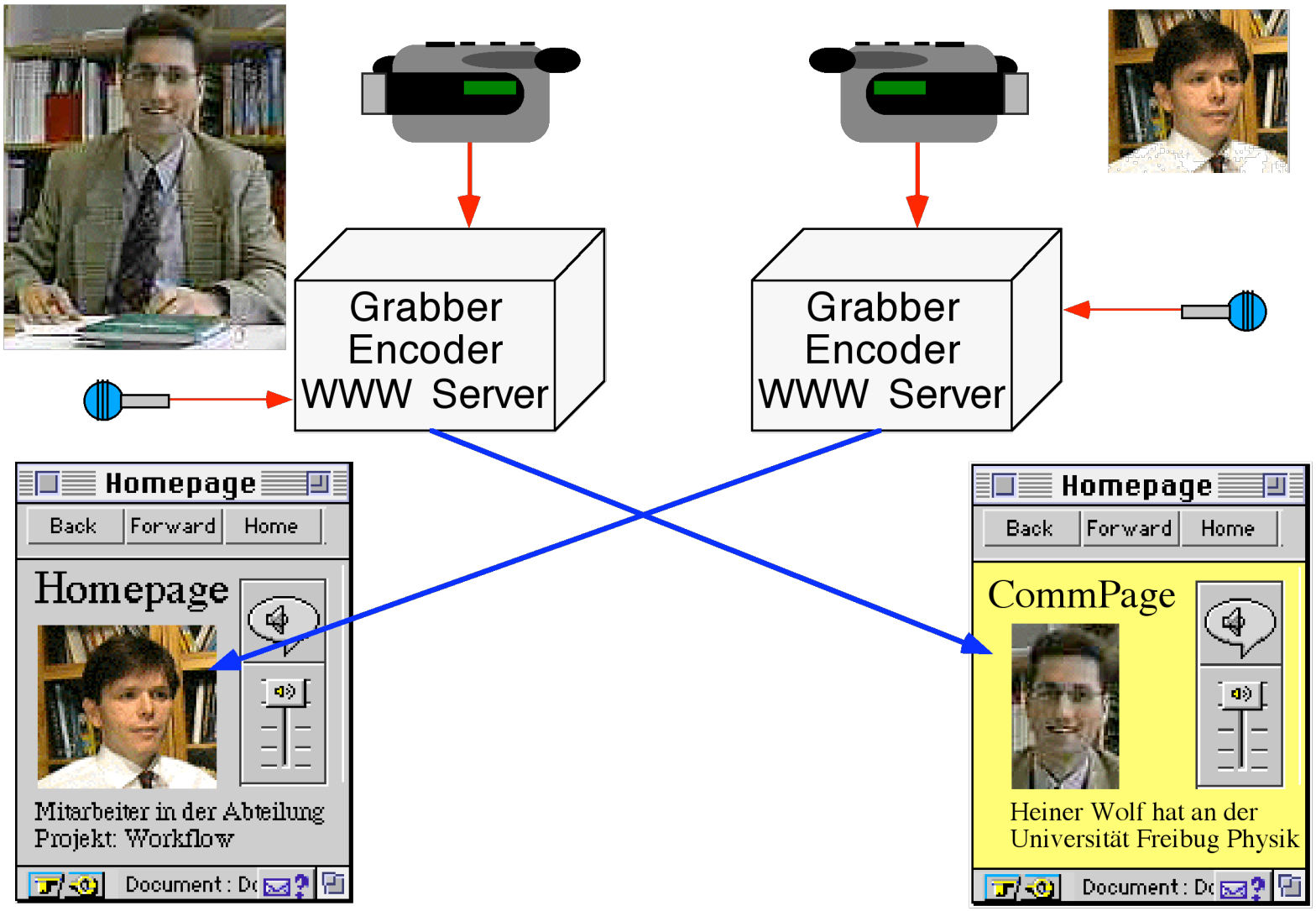
- Meeting Place
  - Visualisierung der Nachbarschaft
  - Konferenz-Management
- Primitive Variante
  - ASCII-Liste von Nachbarn
  - Button drücken: Konferenztool starten
- WWW-Variante
  - Icon-Liste mit URLs
  - Anklicken bringt Comm-Page
- VRML
  - Virtual Reality Modelling Language
  - MPEG-4, Java3D, X3D



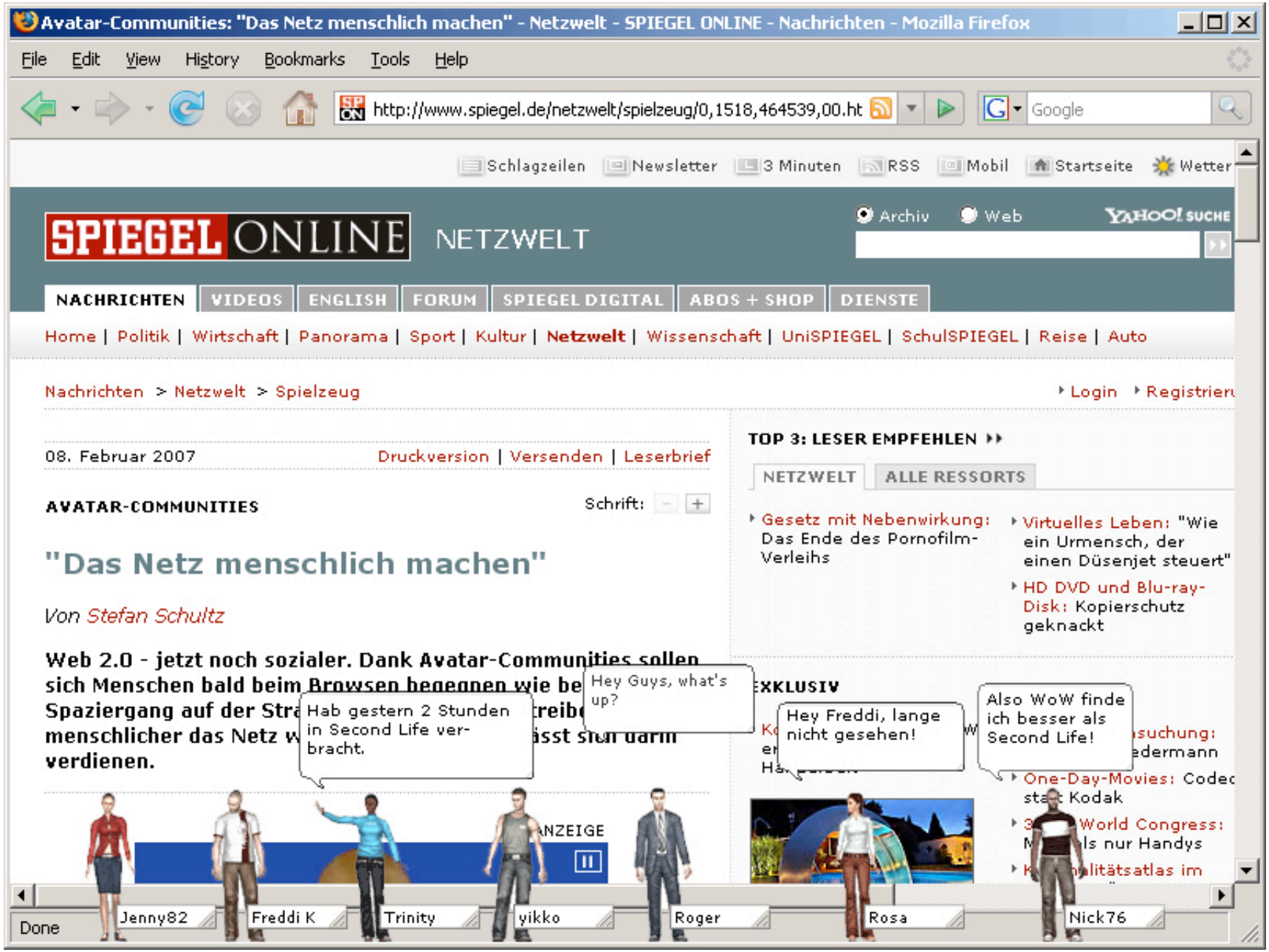
- Individuell zusammengestellte Konferenz
  - Anklicken der CommPages
  - In Browser-Fenstern
  - Nicht unbedingt vollständig



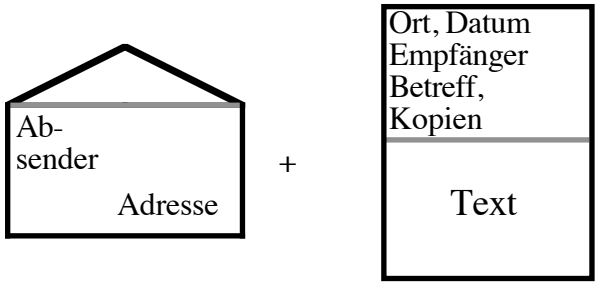
- WebMedia als Basis



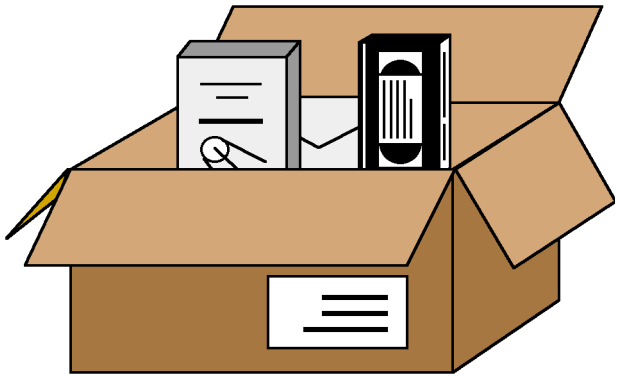
- kommerzielle Versionen terra (cyland), weblin, …



6.4 Electronic Mail



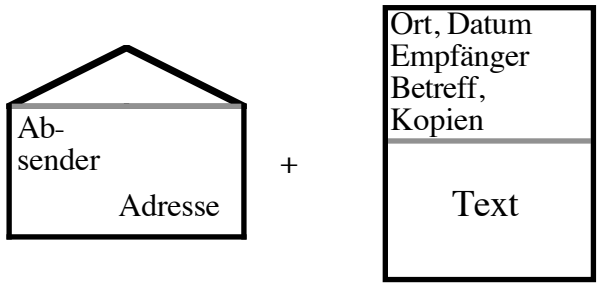
- Asynchrone, paketisierte Kommunikation
- E-Mail: Versand elektronischer Dokumente
  - Brief und Fax (Text, Grafik, Photo)
- Multimedia-Mail
  - Päckchen oder Paket
  - Text, Grafik, Photo, Audio, Video, …



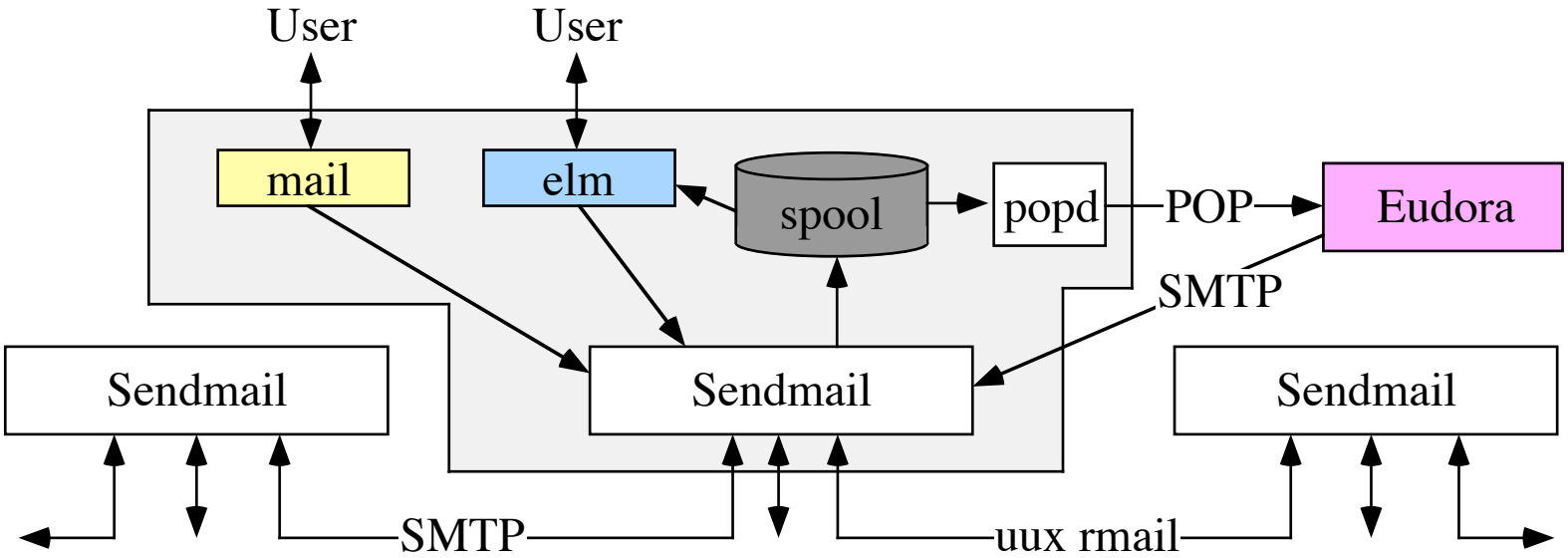
- Standards über Standards:
  - X.400 (ITU),
  - SMTP (Simple Mail Transfer Protocol)
  - MIME (Multipurpose Internet Mail Extensions)
  - Herstellerformate: PROFS, All-in-one, MAPI, VIM, Lotus Notes, …
- Formate
  - Adressen, Weglenkung und andere Versanddaten
  - Zeichensätze, Textformatierung
  - Multimedia-Inhalte, Komposition
  - Verschlüsselung (PGP …), Authentisierung

6.4.1 Internet Mail

- Umschlag
  - Empfängeradresse
  - Absenderadresse
  - Poststempel
- Briefbogen
  - Briefkopf
  - Datum
  - Betreff
  - Text
  - Unterschrift etc.
- RFC 822 Standard for the Format of ARPA Internet Text Messages
  - Syntax
  - Message = Envelope + Content
  - nur Format und Semantik für Content
- RFC 821 SMTP: Simple Mail Transfer Protocol
  - Übertragung von Nachrichten



- Modell: Store and Forward



- Mailer zur Bearbeitung der Nachrichten
- Relay Funktion zentral
- RFC 1939: POP Post Office Protocol
  - Rechner nicht immer eingeschaltet
  - Ressourcen im PC zu knapp
- List-Server
  - spezielle Empfänger
  - Kopien an konfigurierte Liste im Server

RFC 821 SMTP: Simple Mail Transfer Protocol

- End-to-end Verbindung z.B. mit TCP
- SMTP-Server nimmt mail entgegen
  - für bekannte user, eventuell forward
- Einfaches Beispiel

```
S: MAIL FROM:<pschulthe@adobe.com>
R: 250 OK

S: RCPT TO:<frz@informatik.uni-ulm.de>
R: 250 OK

S: RCPT TO:<hertrampf@informatik.uni-ulm.de>
R: 550 No such user here

S: RCPT TO:<lupper@informatik.uni-ulm.de>
R: 250 OK

S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Hallo ins kalte Schwaben ...
S: ...bei uns ist es schoen warm.
S: <CRLF>.<CRLF>
R: 250 OK
```

RFC 822 Format of ARPA Internet Text Messages

- Syntax der Nachricht
- Format und etwas Semantik für Header
- einfachster Header

```
Date:       13 Aug 96 1413 EDT
From:       frz@informatik.uni-ulm.de
To:         fatboy@apple.com
```

- Normaler Header

```
Date:       26 Aug 76 1430 EDT
From:       George Jones<Group@Host>
Sender:     Secy@SHOST
To:         "Al Neuman"@Mad-Host,
             Sam.Irving@Other-Host
Message-ID:  <some.string@SHOST>
```

- Komplexer Header

```
Date       :  27 Aug 76 0932 PDT
From       :  Ken Davis <KDavis@This-Host.This-net>
Subject    :  Re: The Syntax in the RFC
Sender     :  KSecy@Other-Host
Reply-To   :  Sam.Irving@Reg.Organization
To         :  George Jones <Group@Some-Reg.An-Org>,
              Al.Neuman@didel.dadel.dudel.de
cc         :  Important folk:
                Tom Softwood <Balsa@Tree.Root>,
                "Sam Irving"@Other-Host;,
              Standard Distribution:
                /main/davis/people/standard@Other-Host,
                "<Jones>standard.dist.3"@Tops-20-Host>;
Comment    :  Sam is away on business. He asked me to handle
              his mail for him.  bla bla bla.
In-Reply-To: <some.string@DBM.Group>, George's message
X-Special-action:  This is a sample of user-defined field-
              names.  There could also be a field-name
              "Special-action", but its name might later be
              preempted
Message-ID: <4231.629.XYzi-What@Other-Host>
```

- Sendmails fügen Pfadinformation ein

Return-Path: <mccreigh@Adobe.COM>
Received: from hermes.informatik.uni-ulm.de by julia.informatik.uni-ulm.de (4.1/UniUlm-info-1.1r)
  id AA23213; Tue, 29 Oct 96 18:58:14 +0100
Received: from smtp-relay-2.Adobe.COM by hermes.informatik.uni-ulm.de (4.1/UniUlm-info-1.1r)
  id AA09748; Tue, 29 Oct 96 18:58:20 +0100
Received: by smtp-relay-2.Adobe.COM (8.7.5) with ESMTP id JAA27883; Tue, 29 Oct 1996 09:57:37 -0800 (PST)
Received: by inner-relay-1.Adobe.COM (8.7.5) with ESMTP id JAA20209; Tue, 29 Oct 1996 09:56:52 -0800 (PST)
Received: by mail-303.corp.Adobe.COM (8.7.5) with ESMTP id JAA29067; Tue, 29 Oct 1996 09:57:20 -0800 (PST)
Received: by mondial (8.6.9) with SMTP id KAA01118; Tue, 29 Oct 1996 10:02:53 -0800
Message-Id: <199610291802.KAA01118@mondial>
To: Konrad Froitzheim <frz@informatik.uni-ulm.de>
Subject: Re: Peter Schulthess?
In-Reply-To: Your message of "Tue, 29 Oct 1996 15:12:45 +0100."
        <l03010608ae9bdee7c6fe@[134.60.77.22]>
Date: Tue, 29 Oct 1996 10:02:52 PST
From: Ed McCreight <mccreigh@Adobe.COM>

- Probleme des Verteilmodelles
  - Relay-Funktion kann missbraucht werden
  - Relay-Liste mit n-Empfängern: Relay-*Server* sendet mail n-mal
  - Relay-*Server* zahlt gesendetes Datenvolumen

  => Server: Test ob From/Reply-to Domain 'erlaubte' Domain

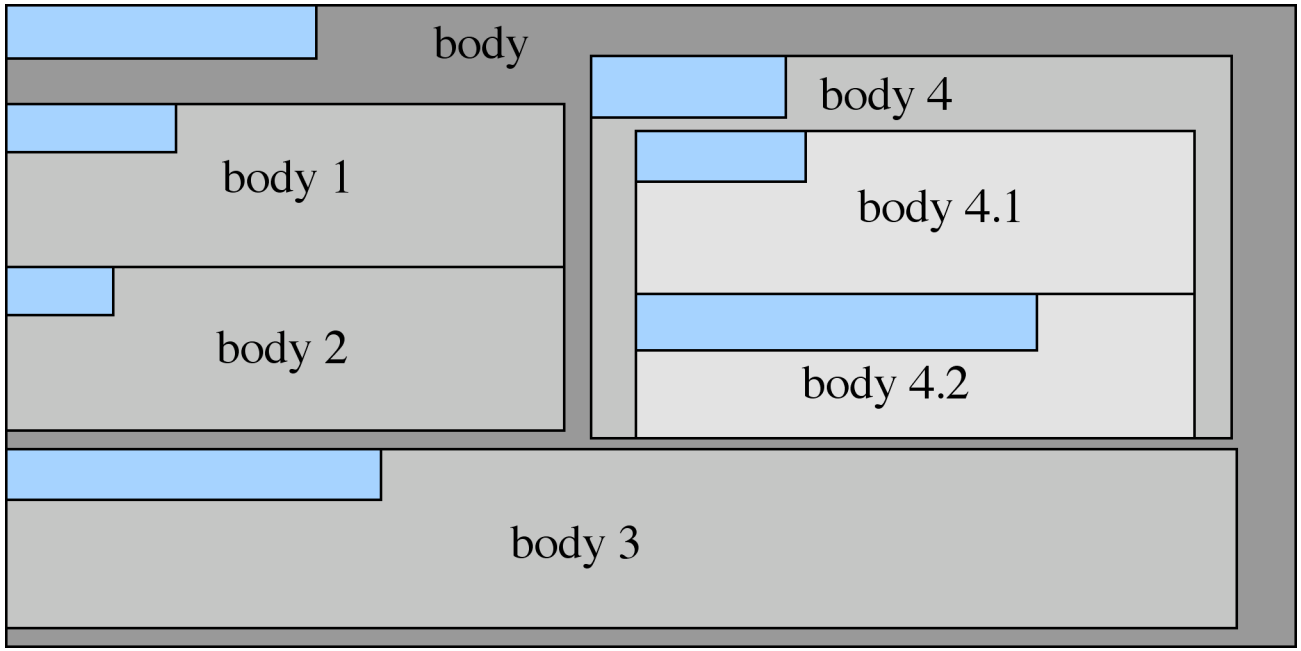
  => Server: nur Mail-Relay für bestimmte IP-Nummern-Bereiche

  => System: authenticated SMTP
- Anti-Spam
  - einfache Tests im empfangenden Mailserver
  - überprüft ob IP-Nummer zu From/Reply-to Domain passt
  - Black-List
  - lernende E-mail-Klienten - Inhaltsbewertung
- Absender-Verifikation: pgp/gpg-Signatur, Zertifikate
- Einschränkungen des Formates
  - nur ASCII-Zeichen
  - keine Zeile länger als 1000 Zeichen
  - begrenzte Nachrichtenlänge

6.4.2 MIME - Multipurpose Internet Mail Extensions

- Interent-Mails enthalten nur ASCII-Zeichen
- RFC 821 und RFC 822 spezifizieren Adressen und Übertragung
- RFC 1341
- Ziele:
  - mehrere Objekte in einer Datei
  - beliebige Zeilen- und Textlänge
  - ISO 8859-X Zeichensätze
  - Fonts
  - binäre Daten
  - Audio
  - Video
  - anwendungsspezifisch
- Kompatibel mit RFC 822
- Subset-Implementierung möglich
  - Minimaler Subset vorgeschrieben
- Neue Felder im RFC 822 Header

- body-part: header+body



- Neues Feld: Mime-Version

```
…
Mime-Version: 1.0
…
Date: Fri, 01 Nov 1996 12:05:56 +0100
To: frz@informatik.uni-ulm.de
…
```

- Neues Feld: Content-Type

  `Content-Type := type "/" subtype [";" parameter]`

  - Beispiele
    image/jpeg
    image/GIF
    audio/x-wav
    video/quicktime
    video/mpeg
  - 7 definierte Content-Types:
    Application, Audio, Image, Message, Multipart, Text, Video
  - X-TypeName
  - Registrierung neuer Content-Types
- Application
  - Application/Octet-Stream;<Name>;<Type>;<Conversions>; <Padding>
  - Application/ODA
  - Application/PostScript

- Multipart
  - /Mixed: serielle Präsentation
  - /Alternative: unterschiedliche Repräsentationen (Bsp: Sprachen)
  - /Parallel: gleichzeitige Präsentation
  - Teile getrennt durch 'boundaries': Parameter Boundary

```
…
Mime-Version: 1.0
Content-Type: multipart/mixed;
boundary="====================_846871556==_"
…
Präambel: to be ignored
--====================_846871556==
Content-type: text/plain; charset=us-ascii
Text explizit als ascii gekennzeichnet,
wie es sein sollte
--====================_846871556==
Text mit implizitem Typ
====================_846871556==--
Schluss, to be ignored
```

- Neues Feld: Content-Transfer-Encoding
  - RFC 821: 7 bit
  - Mechanismen zur Codierung von 8-bit
  - BASE64: 3 Byte => 4 6-bit Zeichen (24 =>24)
    64 Buchstaben: 00, …, 3F => A, B, …, 9,+,/
    ähnlich uuencode
  - Quoted-Printable: '=' als Escape-Zeichen
    M=9Fnchen
    nach 75 Zeichen: = CR
  - 7-bit, 8-bit: kurze Zeilen
  - binary: beliebige Zeilenlänge

```
Content-type: text/plain; charset=ISO-8859-1
Content-transfer-encoding: base64
```

- Neue Felder: Content-ID und Content-Description optional
- Message/External-Body
  - Referenz auf den echten Body

```
Content-type: message/External-Body; access-type=ANON-FTP;
    name=ernst.informatik.uni-ulm.de/usr/local/www/bild.gif
 Content-type: image/gif
```

6.5 WWW

- Dr. Tim Berners-Lee CERN '89
  - Zweck: Verknüpfung von Dokumentation der Hochenergiephysik
  - keine Bilder - textbasierte Klienten

5.5.1 Grundlagen

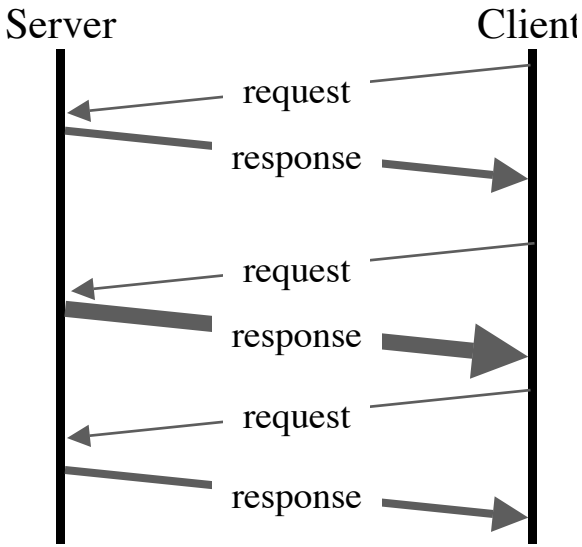
- Internetdienst wie eMail, FTP, gopher
  - Protokoll HTTP
  - eMail: **SMTP** <-> WWW: **HTTP**
  - Dokumentenformat html

5.5.1.1 URL: Uniform Resource Locator

- Namensraum für Objekte im WWW
- Aufgespannt durch Kombination mehrerer Namensräume
  - Protokoll
  - Hostadresse + Serverport
  - Pfadnamen (UNIX-style)
- Beispiel: http://frodo.informatik.uni-ulm.de:80/test/Beispiel1.txt
  identifiziert Datei: /usr/local/www/htdocs/test/Beispiel1.txt auf frodo

5.5.1.2 HTTP: Client-Server Modell

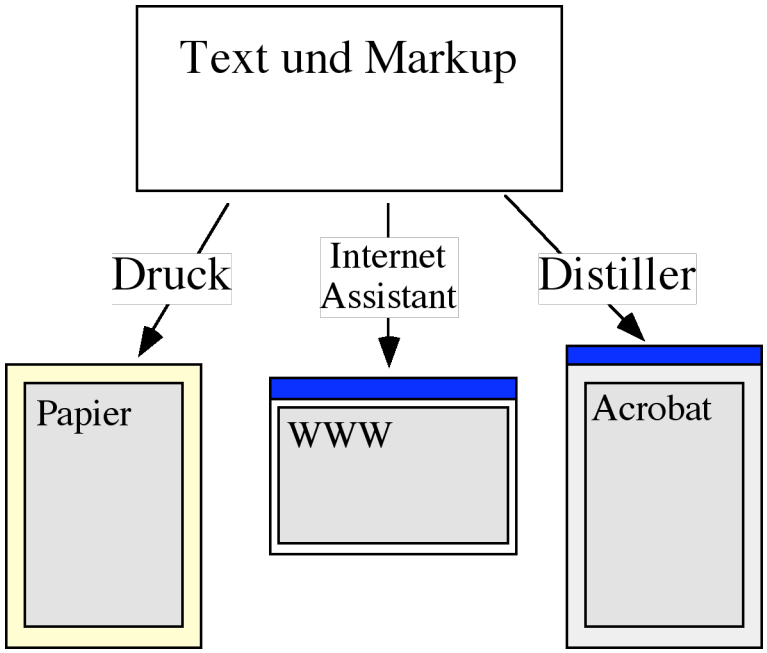
- Request-Response Mechanismus:
  - Request: Typ, Attribute (Header fields / Request fields), Objekt
  - Response: Typ, Attribute (Object Metainformation), Objekt
- Request-Typ (Methode)
  - **GET**, PUT (integrierte Dokumenterstellung)
  - POST, DELETE, ...
- Response-Typ / Code
  - **OK** 200 (2xx), Error 4xx, 5xx
  - No Response 204, Redirection 3xx, ...
- Objekt-Typ beim request unbekannt
  - MIME Typen: Bsp: text/plain, image/gif
  - im Response-Header als Attribut
  - Beispiel Response:




```
HTTP/1.0 200
Content-type: text/plain
Expires: Sun 26 Mar 95 17:50:36 GMT
Ein Beispieltext
```

6.5.1.3 HTML, die 'Sprache' des WWW

- Hypertext Markup Language [Berners-Lee 1989]
  - Verknüpfung von Dokumentation der Hochenergiephysik
  - keine Bilder - textbasierte Klienten
  - Standard Generalized Markup Language
  - Document Type Definition (DTD) von SGML
  - Hypertext-Referenzen: URLs eingebettet in das Dokument
  - http://www.w3.org/TR/REC-html40/
- Markup
  - logische Struktur für Text
  - Überschrift, normaler Paragraph, Zitat, …
  - Fußnote, Literaturverweis, Bildunterschrift, …
- Zuordnung der Attribute beim Satz
  - Autor produziert Inhalt und Struktur
  - Drucker setzt
  - Corporate Identity …
- HTML: ASCII-Text + <A>-Tag



- Beispieltext:

```
<HTML> <HEAD>
<TITLE> Ein HTML-Beispiel </TITLE>
</HEAD> <BODY>
<B>Dies</B> ist ein Hypertext Dokument.
<P>Mit einem Bild: <IMG SRC="bild.gif"> <BR> und einem
<A HREF="Beispiel1.txt"> Hyperlink </A> </P>
</BODY> </HTML>
```

Dies ist ein Hypertext Dokument.

Mit einem Bild: 

und einem Hyperlink

- Elemente:
  - Stile
  - Listen
  - Formatierung
  - Links

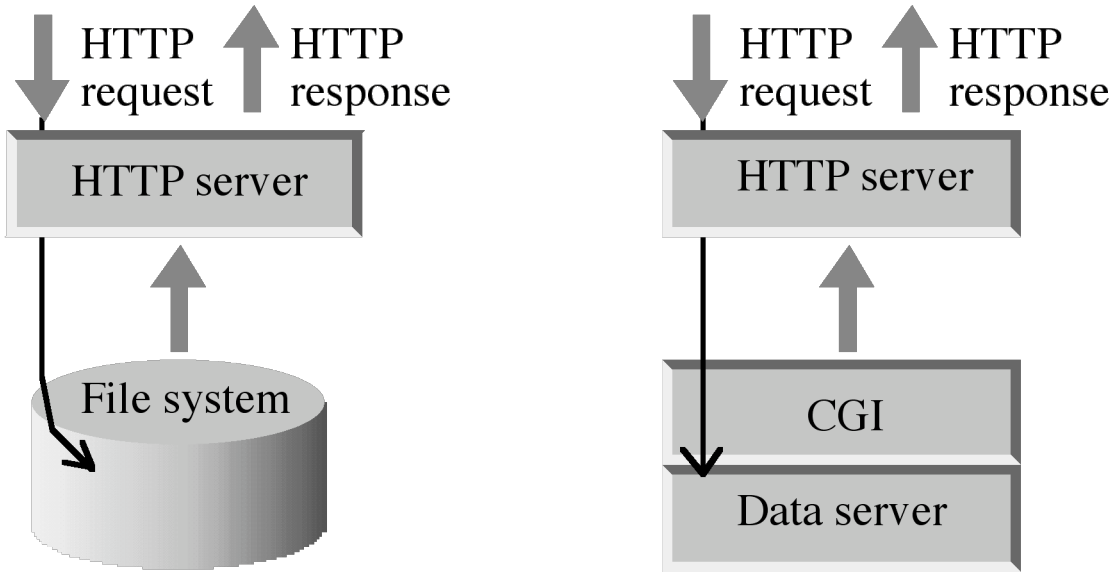
- Medienintegration
- Im Browser integrierte Viewer
  - HTML, GIF, JPEG
  - FTP-, gopher-Verzeichnisse
  - MPEG (a/v) ?
  - Streaming problematisch
- Externe Programme
  - Externe Viewer/Handler: MPEG, Audio, Postscript, uudecode
  - Präsentation von beliebigen Objekten
  - Zuordnung von Objekt und Viewer durch MIME-Typ
- Kennzeichnung mit MIME-Typen
  - text/html, image/gif, image/jpeg, video/quicktime, video/mpeg
  - application/rtf
  - Server-seitig konfiguriert bzw. abgeleitet
  - Unix, Windows: nach Namenserweiterung (.htm, .gif)
  - auch Heuristiken im Klienten: Namenserweiterung, Inhaltsanalyse
- XML, CSS, …

6.5.2 Details

6.4.2.1 Erweiterungen auf Serverseite

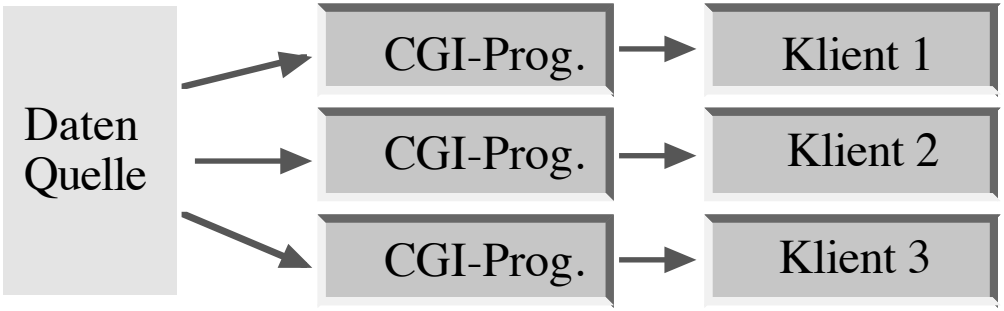
• CGI - Common Gateway Interface

- Erweiterung des Namensraums -> Programmenausgaben



- Realisierung als Kindprozess (Unix): stdout -> Server
  Applescript (MacOS): AEReply -> Server
- CGI-Typen:
  standard: Programmausgabe -> Datenteil der HTTP Response
  erweitert: volle Verbindungskontrolle: stdout=socket(TCP)

- Beispiel Imagemap:
  - Request: `GET /cgi/imagemap/Beispiel5.map?85,82`
  - Server: % `imagemap Beispiel5.map 85,82`
  - Beispiel5.map: `rect /staff/Wolf.html 6,58 147,113`
  - besser: Klientenseitige Auswertung (vgl. Forms)
- Andere Anwendungen für CGI:
  - Gateway zu: WAIS, Archie, Datenbanken, finger
  - kontinuierliche Medien: Text, Audio, Video
  - allgemein: Dateisystem: abgelegte Daten (Dateien)
    CGI: generierte Daten (Datenbankabfrage)



6.5.2.2 Andere Dienste im WWW

- Dienst identifiziert durch Dienstkennung im URL
  - URL = Dienstkennung : dienstspezifischer Teil

    http://frodo.informatik.uni-ulm.de:80/test/Beispiel1.txt
  - WWW-Klienten unterstützen mehrere Internetprotokolle

    FTP (RFC 959): ftp://134.60.77.7/pub/Res2PICT.hqx
    SMTP (RFC 822): mailto:wolf@informatik.uni-ulm.de
    Gopher (RFC 1436): gopher://cell-relay.indiana.edu/
    NNTP (RFC 977): news:comp.infosystems.www.announce
    Local : file:/home/wolf/.login
- Dienste integriert
  - WWW ist Kombination vieler Dienste
  - Beispiel: http://frodo.informatik.uni-ulm.de:80/soft/

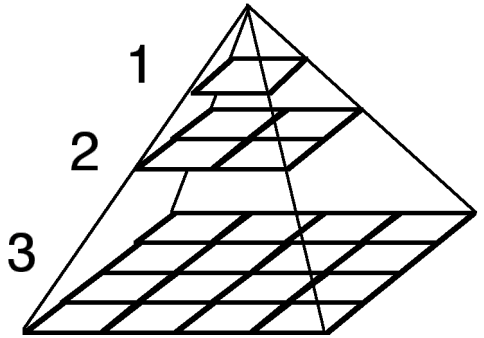
    ftp://user:pwd@www-vs.informatik.uni-ulm.de/pub/bsp.html

    news:ulm.test

    mailto: user@maschine.domain.de

6.5.2.3 Antwortzeit Optimierungen

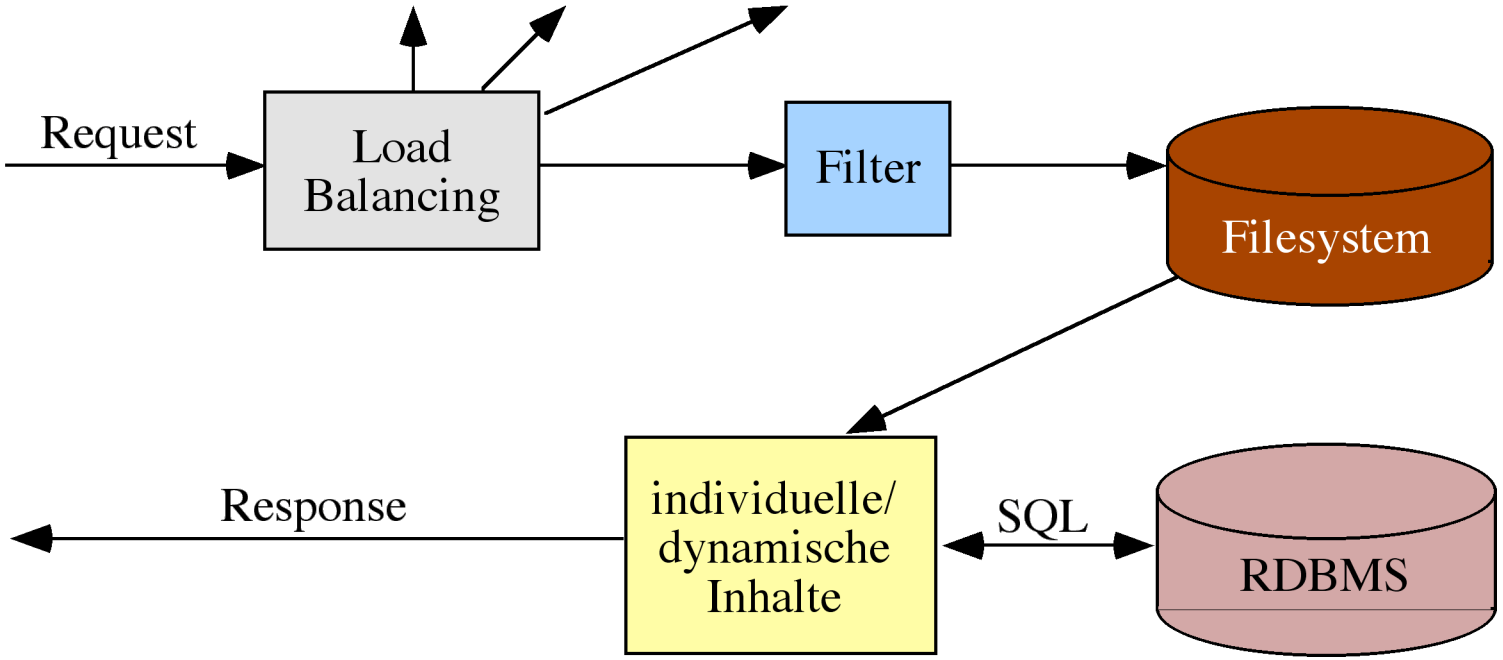
- Caching von Objekten (URLs)
  - lokal: Arbeitsspeicher, Massenspeicher
  - Proxy-Server: transparenter Mittler/Cache zwische Klient und Server
    auch mehrstufig: Fakultät, Universität, Provider, Kontinent ?
  - Expiration-Date für Cache-Management
- Hierarchisch organisierte Objekte:



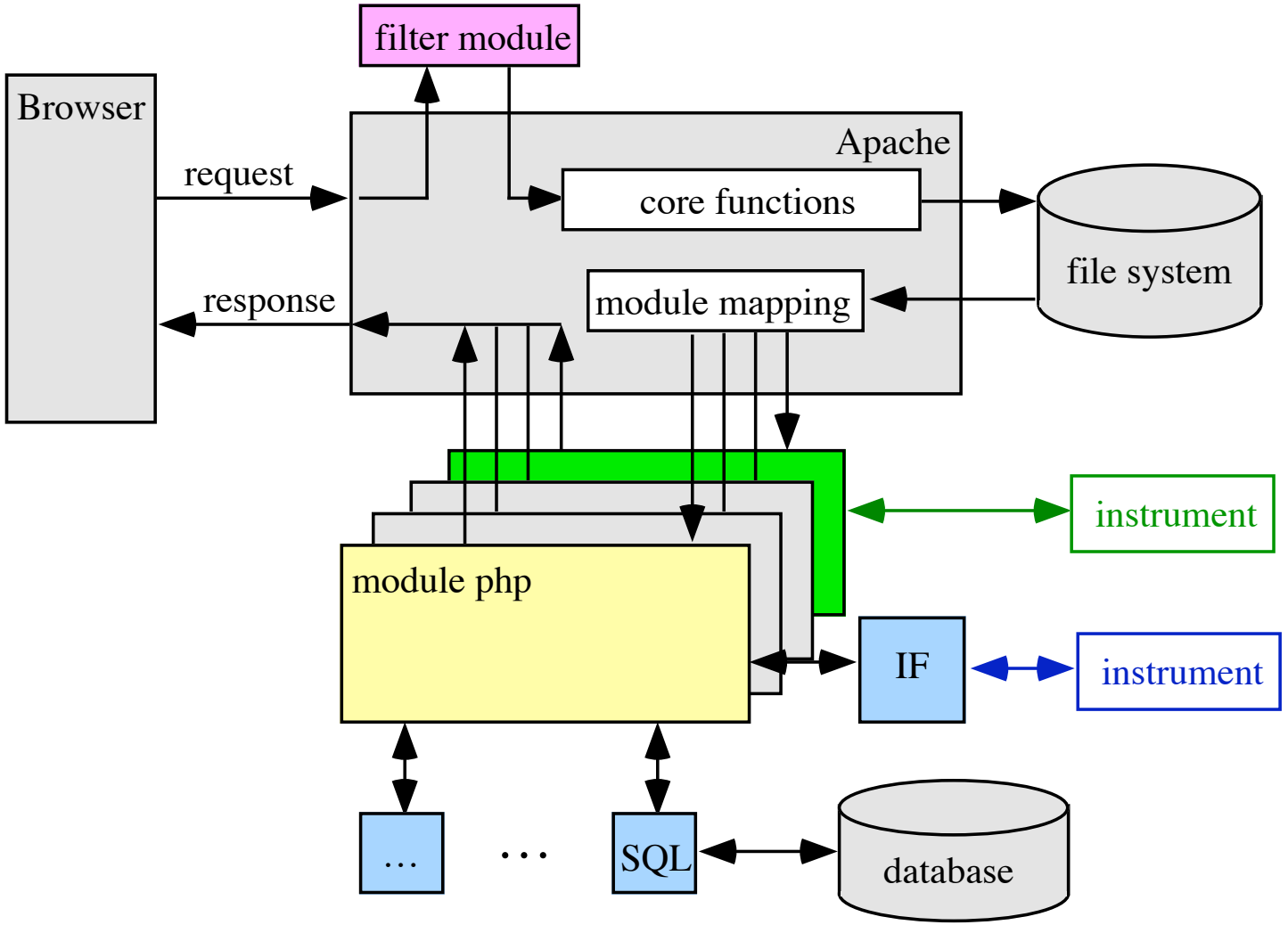
- HTTP-NG (2.0)
  - langlebige Transportsystem Verbindung
- Server-Antwortverhalten
  - Parallelität durch Threads statt fork
  - Hochleistungsserver als Cluster
- Content Delivery Networks: Akamai, …

6.5.3 Struktur eines Webservers

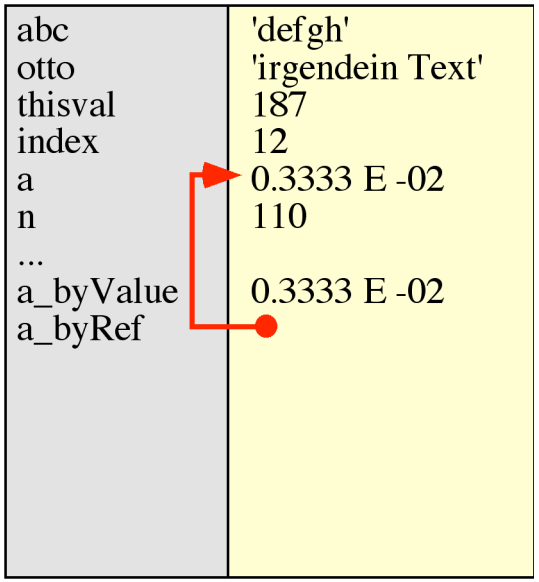
- Sites mit viel Verkehr
- Inhalte an individuellen Besucher anpassen
  - Shopping System
  - Abonnent, Suche, …
- Systematische Integration weiterer Datenquellen
  - Datenbanken
  - beliebige Programme



- Serverstruktur am Beispiel: Apache
  - Request-shaping (Filter)
  - Filesystem-Zugriff
  - Datei auf Modul mappen
  - Inhalt verändern



- Skriptsprachen
  - individuell generierte Seiten
  - z.B. php
- Datei im Filesystem enthält Vorlage (template)
  - Standardnavigation, Firmenlogo, etc.
  - Skripte in der Seite
  - besorgen dynamischer Inhaltskomponenten (Rückfrage)
  - erzeugen html-Stücke
- php (siehe http://www.php.net)
  - Syntax C-inspiriert
  - Datentypen: Boolean, Integer, Float, Strings, Array
  - Variablen: $<name>
  - Zuweisung 'by value' und 'by reference'
  - Kontrollstrukturen: if-else, do-while, for, switch, …
  - Funktionen: nichttypisierte Parameter, return-wert
  - Klassen und Objekte



- Primitives Beispiel

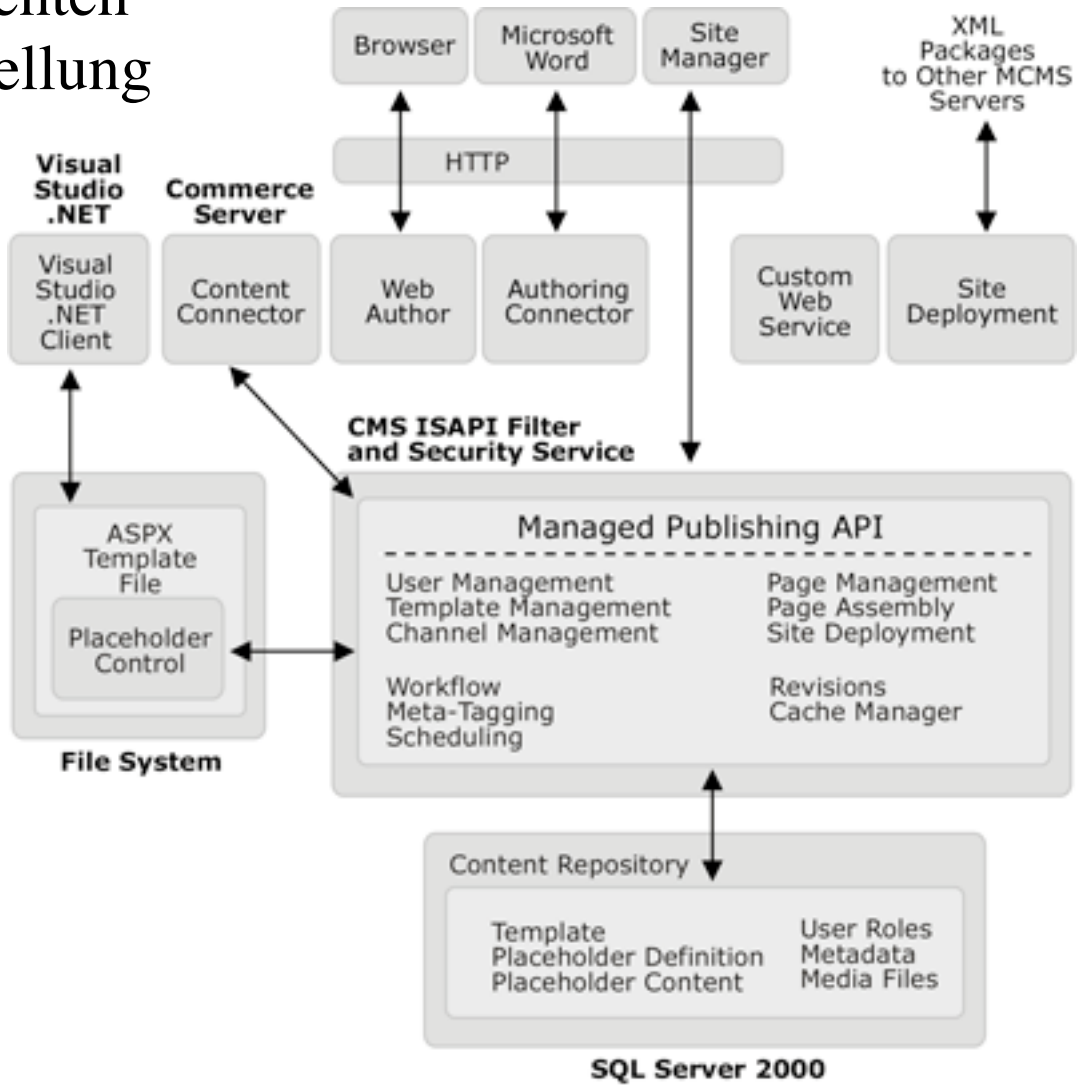
```
<html> <head> <title>Example</title> </head>
    <body>
        <?php
         if (strstr($_SERVER["HTTP_USER_AGENT"], "MSIE")) {
           echo "<p> You are using Internet Explorer </p>";}
        ?>
    </body>
</html>
```

  - php-Systemvariable `$_SERVER["HTTP_USER_AGENT"]`
  - Zugriff auf request: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
  - Funktion `strstr(string1, string2)` sucht string2 in string1
  - erzeugt Ausgabedatei:

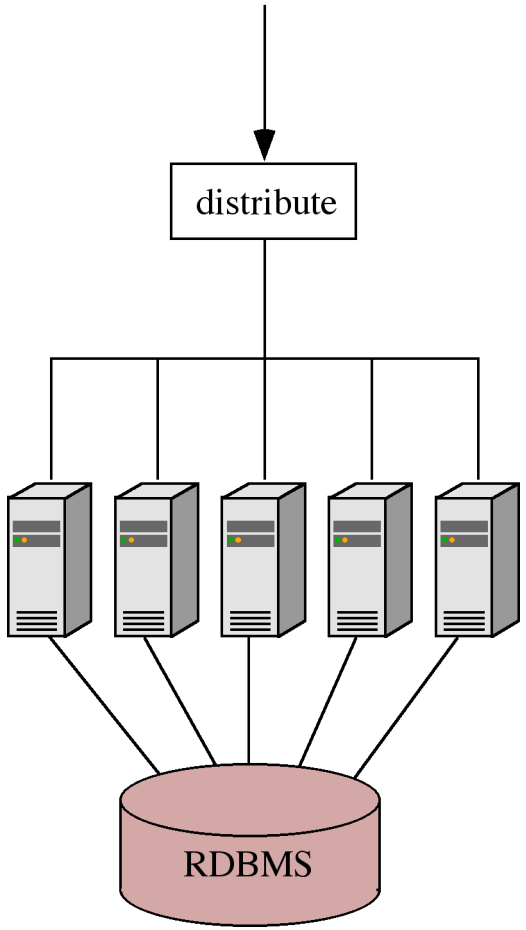
```
<html> <head> <title>Example</title> </head>
    <body> <p> You are using Internet Explorer </p> </body>
</html>
```

- Funktionen
  - sind API zu externen Programmen
  - in Gruppen zusammengefaßt
  -SQL, DBase, Cybercash, ftp, XML, zip, W32API

- Content Management
  - Templates für die Seitenstruktur
  - Inhalt = Komponenten
  - viele Autoren erstellen Komponenten
  - Workflow für Komponentenerstellung
  - Rollenkonzept
  - Einstellen von Text, Bildern, …
  - Versionen, Links
  - Microsoft ->
  - Umschalten des Inhalts
  - Staging Server



- Lastverteilung
  - mehrere/viele Server für eine Site
  - transparent für Nutzer (inklusive Bookmarks und Suchmaschinen)
  - Integration in Staging-Prozess
  - Persistenz: User kommuniziert während einer Session mit einem Server
- Option 1: DNS
  - www.xxx.yy -> 139.17.17.1 … 139.17.17.n
  - Probleme mit Caches etc.
- Option 2: Router
  - eine Virtual-IP-Adresse zu den Klienten
  - Routen an Pool von Servern mit echter IP
  - Cisco LocalDirector, F5 Networks's BIG-IP
- Option 3: MAC-Layer Switch
  - ähnlich Multicast
  - Server entscheiden dezentral
- Option 4: Eingangsserver
  - leitet Request an Inhaltsserver durch
  - oder Redirect
- Sessionsemantik für viele Anwendungen erforderlich:



  - Einkaufswagen
  - Authentisierung bei Abodiensten
  - Webseiten-Zirkulation: 'Distinct User', Page Impressions
  - aber: http ist zustandslos
  - evtl. Konflikt mit Load-Balancing
- Heuristik mit IP-Nummer und Zeit scheitert an NAT
- Cookies
  - gesetzt beim ersten Besuch
  - Server fragt Client wiederholt nach der ID
  - Nachfragen verursachen Netzlast
- Session-ID in fast allen Links auf einer Seite
  - vom Server dynamisch eingebettet
  - http://www.amazon.com/exec/obidos/ASIN/1558605967/qid=1052216005/sr=2-1/ref=sr_2_1/103-0347593-2203056
- Langlebige Seitenkomponenten
  - in besonderem Frame
  - z.B. kleines, unendlich langes GIF

6.5.4 JavaScript

- Programmfragmente in HTML
  - Verbesserung von HTML-Seiten auf der Klienten-Seite
  - von Netscape
  - Fenstergröße und -Gestaltung
  - Menus, Effekte, …
- Interpreter im Browser
- Eingebettet in HTML
  - script-Tag

```
<html><head><title>Test</title>
<script language="JavaScript">
<!--
 alert("Hallo Welt!");
//-->
</script>
</head><body>
</body></html>
```

- Oder in anderen HTML-Tags

```
<html>
   <head>
   <title>JavaScript-Test</title>
   <script language="JavaScript">
   <!--
    function Quadrat(Zahl)
     {Ergebnis = Zahl * Zahl;
      alert("Das Quadrat von " + Zahl + " = " + Ergebnis);
     }
   //-->
   </script> </head>
   <body> <form>
   <input type=button value="Quadrat von 6 errechnen"
        onClick="Quadrat(6)">
   </form> </body>
   </html>
```

- Eventhandler
  - Attribut in html-Tags
  - beschreiben Ausführungsbedingung
  - Aufruf einer JavaScript-Funktion
  - onLoad, onClick, onMouseover, …

- Sprache
  - Notation ähnlich Java
- Anweisungen
  - Zuweisungen

```
zahl = 0; zahl++; zahl+=1;
```

  - Bedingte Anweisungen und Schleifen

```
if (Zahl<0) zahl = 0;
while (idx<100) {…; idx++}
for(i = 1; i <= 100; i++)
      {…}
```

  - Funktionsaufrufe

```
alert("Und hier ein kleiner Hinweis");
```

  - Klammern mit {}

```
if (Ergebnis > 100)
{ Ergebnis =0; Neustart(); }
```

- Variablen
  - kein ordentliches Typenkonzept
  - Vereinbarung mit 'var'
  - Typ Zahl oder String
  - wird bei der ersten Zuweisung festgelegt

```
var Antwort = 42;
var Frage = "The Question for god …"
```

  - global oder in Funktion lokal
- Objekte
  - selbstdefiniert
  - vordefiniert in Umgebung
  - window, document, …

```
neuesFenster = new.window;
```

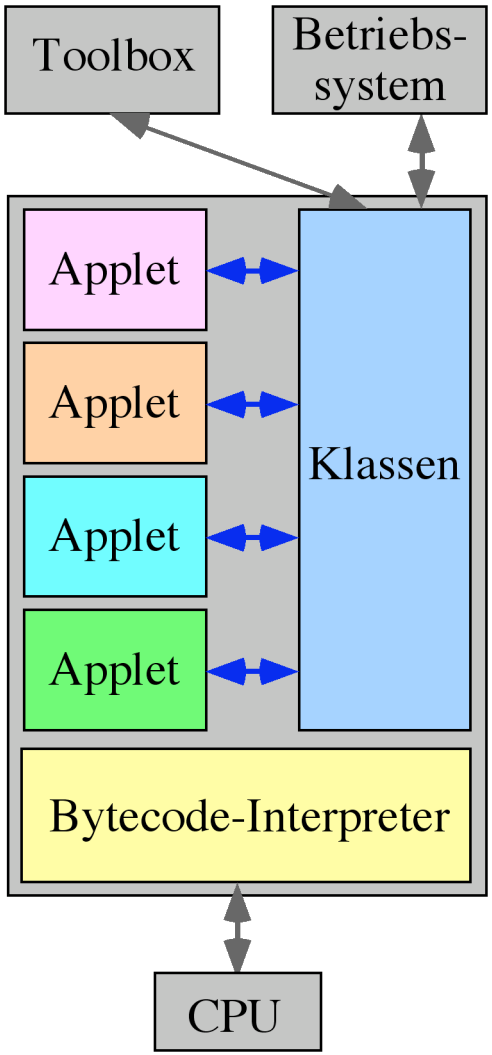
  - Methoden zur Manipulation

```
neuesFenster.open();
```

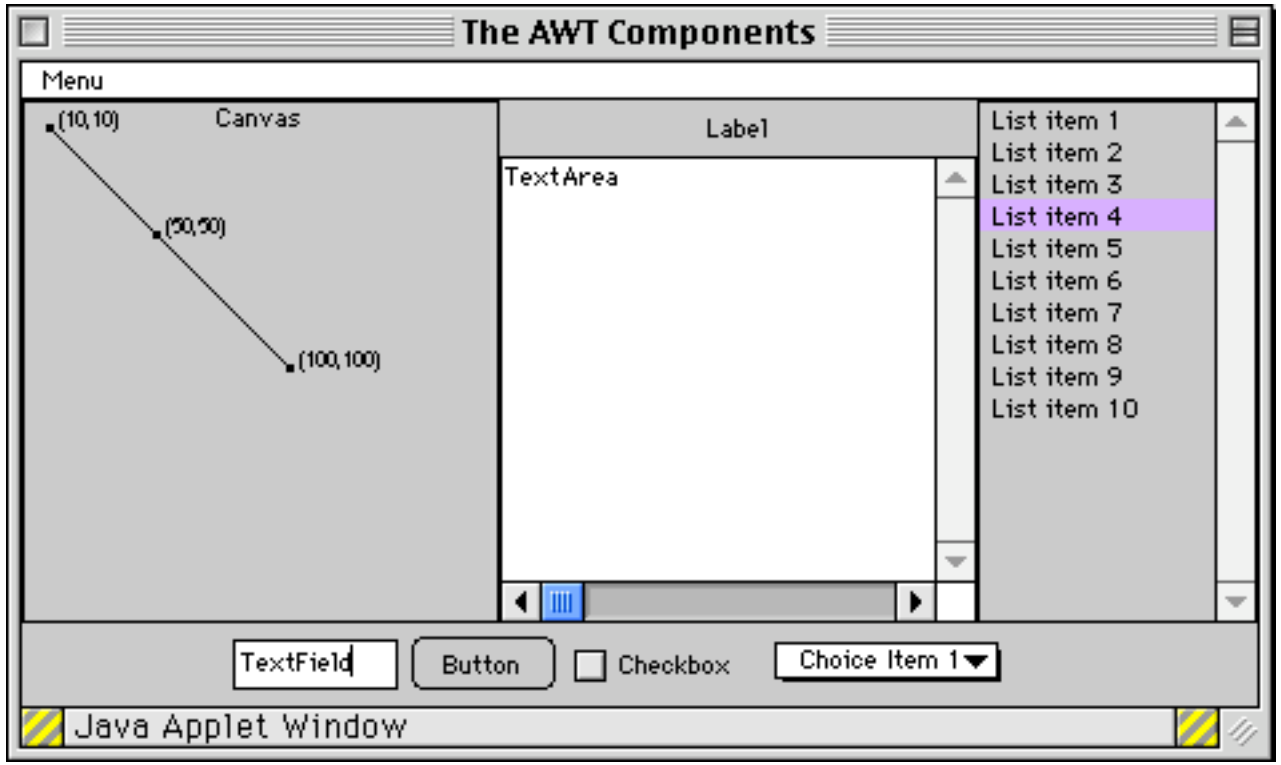
- Funktionen
  - Anweisungsblock mit Variablen
  - Aufruf aus anderen Funktionen und in Eventhandlern

6.5.5 Applets

- Programmiersprache Java
  - ordentliche Programmiersprache
  - Typensicherheit, Objekte, …
  - C-ähnlich
  - native-Compiler möglich
- Java und WWW
  - JavaScript
  - Java Applets
  - serverside Java
- Java Virtual Machine
  - YAVA
  - Virtueller Prozessor
  - Instruktionen: fmul, dmul, dload, ifeq, …
  - Threads
  - Compiler erzeugen i.d.R. Bytecode



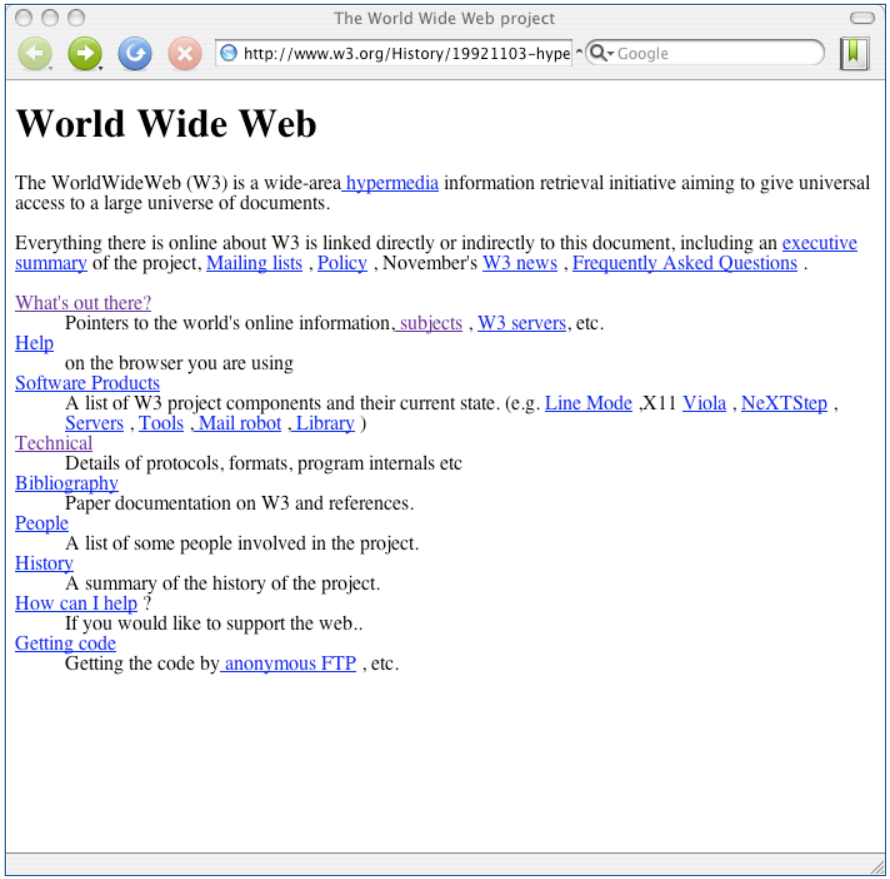
- Java Runtime Environment
  - Virtuelles Betriebssystem mit Toolbox
  - Klassen mit häufig gebrauchten Funktionen
  - Strings, Netzwerk, …
  - AWT: Buttons, Checkboxes, Choices, Lists, Menus, Text Fields
    Windows, Panels, Scroll Panes
    Grafische Objekte

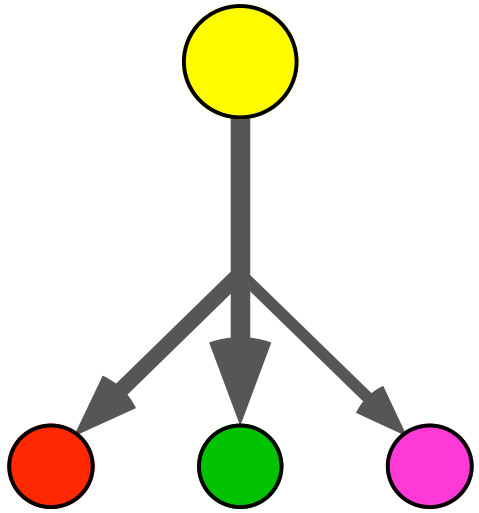


- Sicherheits-Einschränkungen
  - kein Dateizugriff
  - Netzwerkverkehr nur zur Quelle des Applets
- API's
  - JNDI (Naming and Directory), JIDL (CORBA), JDBC (Datenbank)
  - RMI, JTS (Transaktionen)
  - JavaMail, JavaMedia (2D, 3D, Sound, Speech, Telephony)
  - E-Commerce, JavaWallet
  - JINI (Gerätekontrolle)
  - …
- Übertragung von Applets
  - 30% ByteCode, 70% Namen (ASCII-Strings)
  - Klassen einzeln, jeweils eine TCP-Verbindung
  - JAR und komprimiertes JAR (ZIP)

6.6 Stichwort Web 2.0

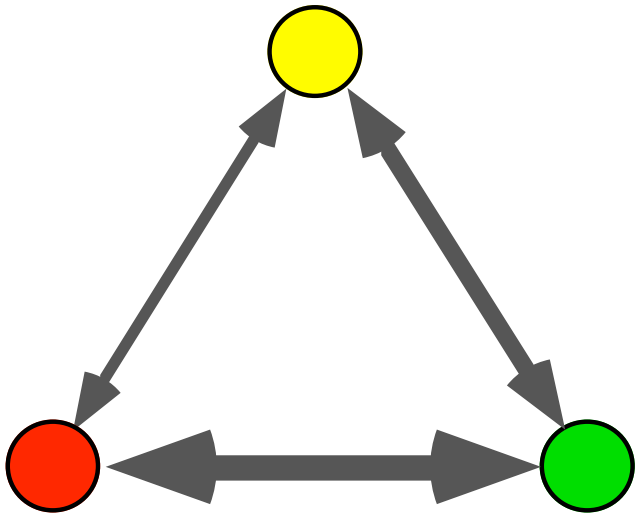
- Das WWW der Wissenschaftler
  - Publizieren
  - Links sammeln, Nützliches anbieten
- Web-Präsenzen
- Meta-Seiten
  - Yahoo's Linksammlung
  - Suchmaschienen (Altavista, Google, …)
- Statische Webseiten
  - request-response
  - makro-interaktiv
  - Inhalte ändern sich (sehr) langsam
- Datenbank-Webseiten ('Web 1.5')
  - Zeitungen
  - Warenwirtschaft -> Versandhandel
  - Amazon als Existenzbeweis
  - Content Management Systeme
- Produzent und Konsument

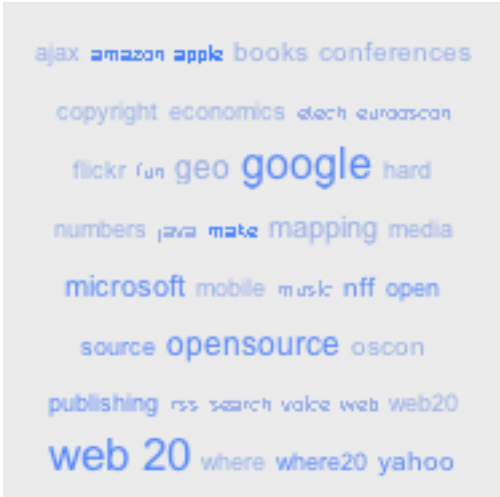




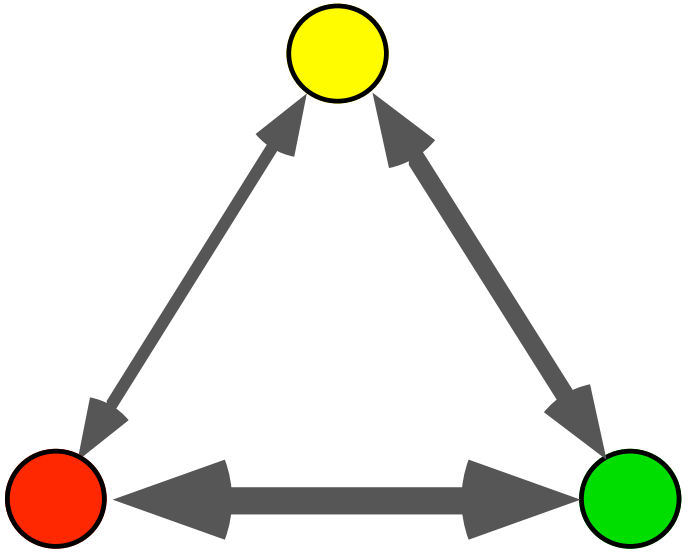
- Erfolgreiche kommerzielle Webdienste?
- Ebay
  - Transaktionssystem
  - Handelsplattform
  - Regeln
  - Kassieren
- Amazon
  - ISBN und Inhalt vom Buchhandel
  - Logistik- und Zahlungssystem
  - Handelsplattform
  - Benutzerkommentare
  - Auswertung des Kaufverhaltens
- Amazon Web Services
  - E-Commerce Service
  - Elastic Compute Cloud
  - Simple Storage Service
  - Amazon Mechanical Turk

- Erfolgreiche vorkommerzielle Webdienste?
- Sourceforge
- Blogs
  - Selbstgeschriebene Kommentare
  - Kommentare von Lesern (=Leserzuschriften)
- Wikipedia
  - Problem Qualitätskontrolle
- Medien Sharen
  - Napster, Kazaa, eMule
  - Flickr
  - YouTube
- Tagging (social bookmarking)
  - del.icio.us
  - Bookmarks sharen
  - Tag clouds



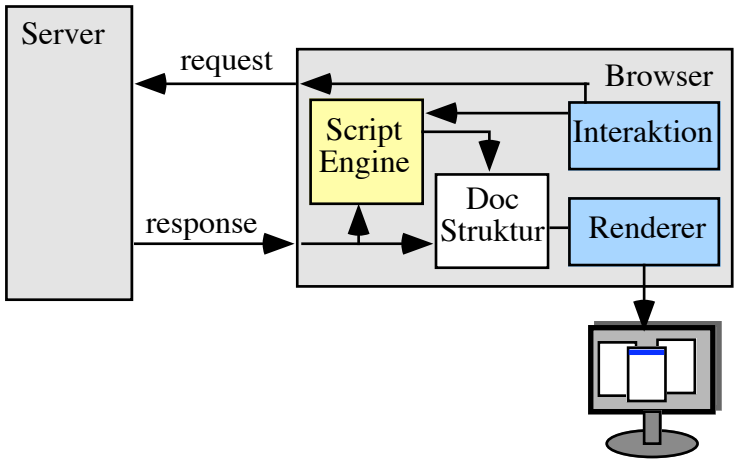


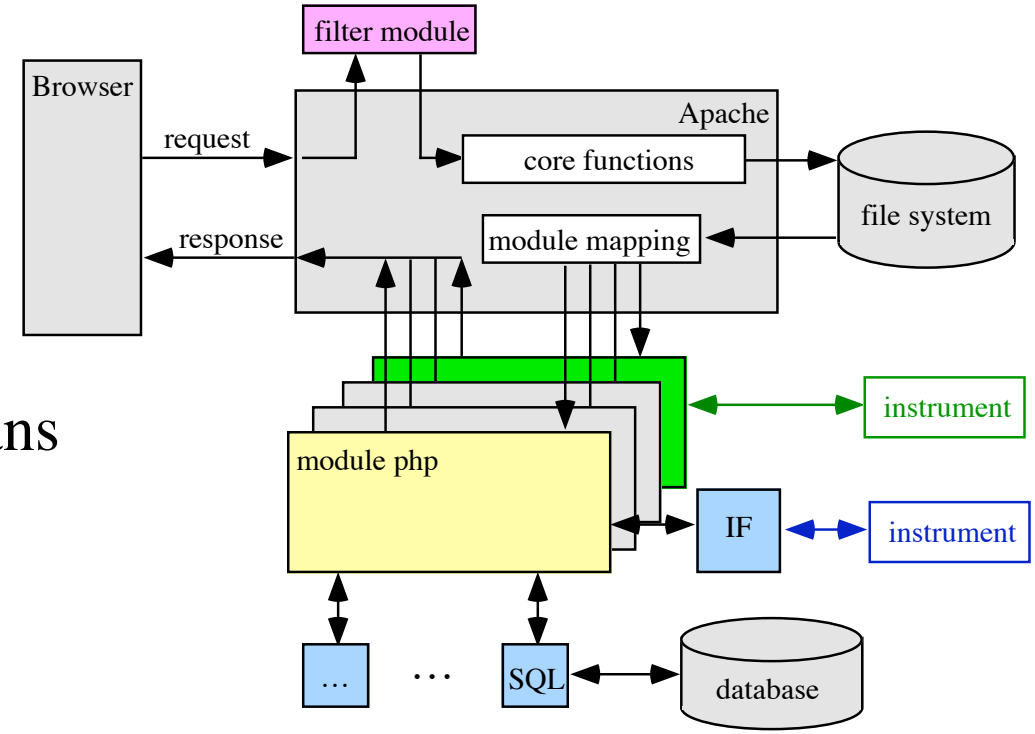
- Gemeinsamkeiten erfolgreicher Webdienste
- Infrastruktur für Benutzer
  - Benutzer erstellen den eigentlichen Inhalt
  - Benutzer schaffen die Werte
- Benutzer arbeiten lassen
  - Auktion erstellen
  - Artikel beschreiben
  - Rezensionen schreiben
  - bloggen und kommentieren
- Benuterbeiträge kanalisieren
  - Regeln erstellen und durchsetzen (Gesetze, 'Anstand', …)
  - Beiträge automatisch kontrollieren
  - Kontrolle durch 'Peers'



- 7 Thesen von Tim O'Reilly
- Einzigartige Daten
  - explizit und implizit vom Benutzer
  - Beispiel Amazon
- 'kollektive Intelligenz' 'bändigen'
  - Wikipedia, Google, Sourceforge, Blogs
  - tagging
- Services anstelle von Softwarepaketen
  - Google statt Netscape
  - Web-Plattformen: Amazon, Google, …
- Das Ende der Pakete und Versionen
- Leichtgewichtige Entwicklung
  - User als Co-Entwickler
  - inkrementell
- Plattformunabhängigkeit: Hardware, OS
- 'Rich User Experience' im Web

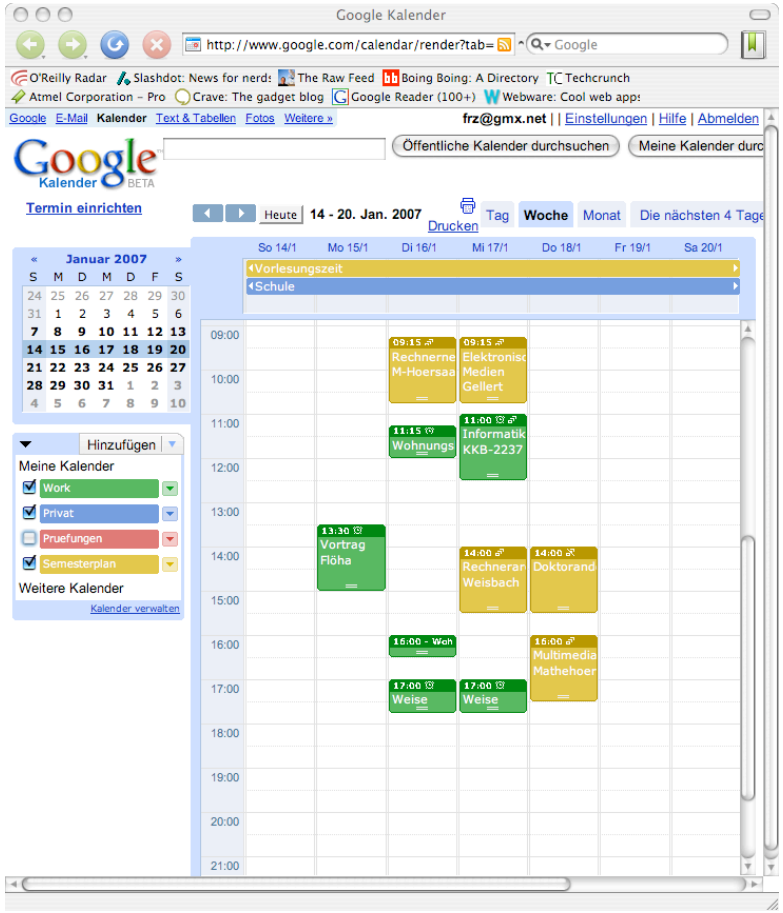
- Webapplikationen
  - Mikro-interaktiv
- Benutzeroberfläche
  - Browser
  - html mit Grafiken als Substrat
  - individuelles Verhalten: JavaScript
- Server
  - Datenbank
  - Geräte
  - Apache und Apache-Module
- Scripting
  - client-side: Javascript, Java Applets
  - server-side: ASP, php, Java und Beans
- Beispiele
  - rr.informatik.tu-freiberg.de
  - Shopping-Seiten



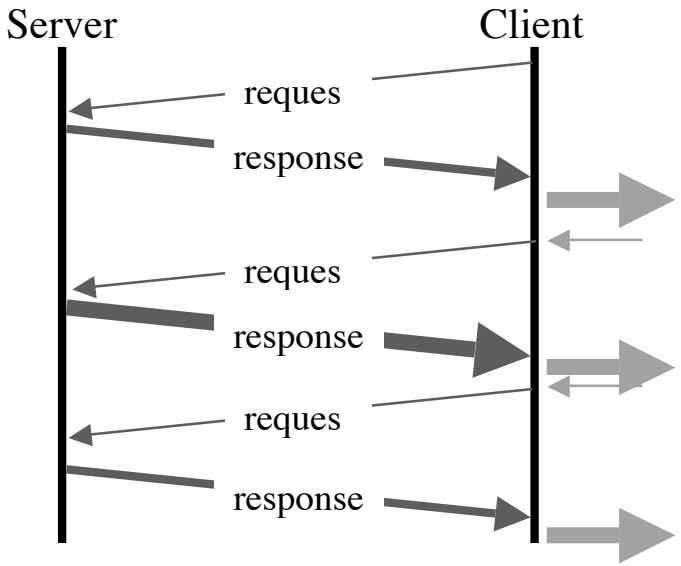


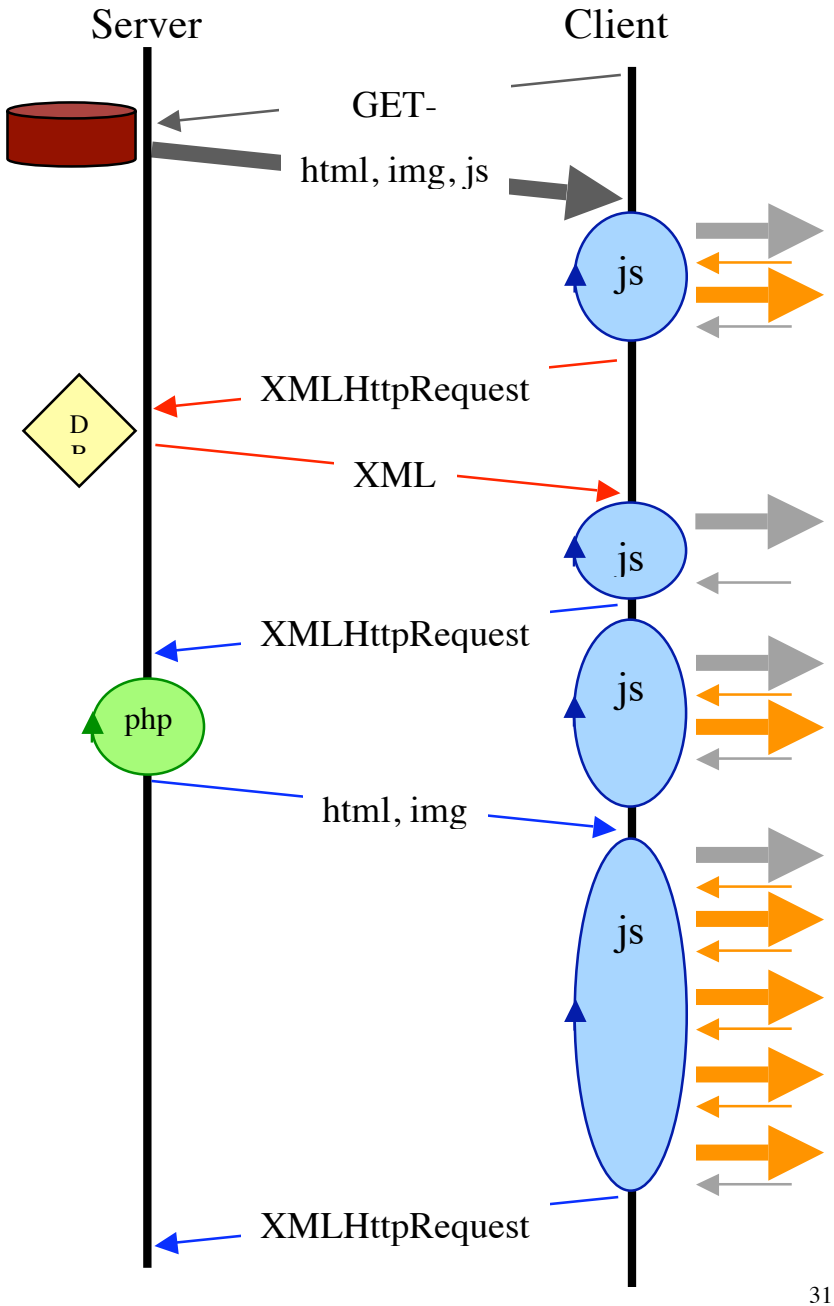
- Programmierparadigma

- Infrastruktur
  - Server:(Apache+php|JavaBeans)
  - Server:Datenbank
  - Client:(InternetExplorer|Firefox|Safari)
- Krümelware
  - kleine, vorgefertigte Bauteile (-> Objekte)
  - abgeleitete (erweiterte) Objekt
  - inkrementelles Programmieren
  - Einfügen und Erweitern statt Bauen
  - Gesamtkonzept? Architektur?
  - Datenfluß designen
- AJAX
  - Asynchronous JavaScript And XML
  - maps.google.com
  - flickr.com
  - docs.google.com, google calendar



- Architektur von AJAX-Applikationen
- Programm im Browser(Client)
  - JavaScript
  - AJAX-Libraries
- Server
  - Webserver, Datenbank
  - Server-side Scripte
- Leichtgewichtige Kommunikation
  - XMLHttpRequest
  - synchron und asynchron





- XMLhttpRequest
- JavaScript Klasse
  - erstmals in IE 5
  - Interface für Http
  - ohne Benutzerinteraktion
  - synchron und asynchron

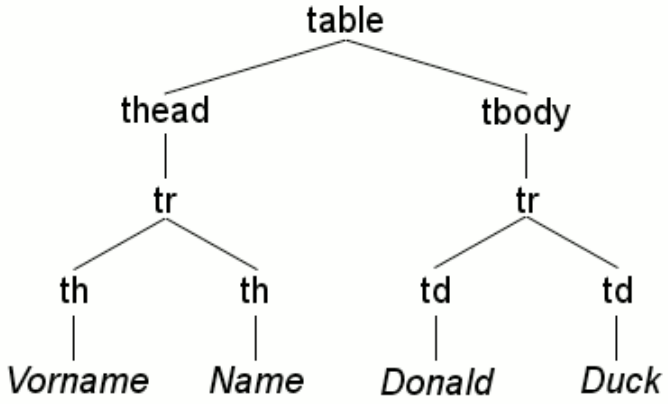
```
function createXMLHttpRequest() {
 try {return new ActiveXObject("Msxml2.XMLHTTP"); } catch(e) {}
 try {return new ActiveXObject("Microsoft.XMLHTTP"); } catch(e) {}
 try {return new XMLHttpRequest(); } catch(e) {}
 alert("XMLHttpRequest not supported");
 return null;}
```

- Properties
  - readystate, status
  - onreadystatechange
  - responseText
- Methoden
  - open
  - send

```
var xhReq = createXMLHttpRequest();
xhReq.open("get", "sumget.phtml?num1=10&num2=20", true);
xhReq.onreadystatechange = function() {
   if (xhReq.readyState != 4)  { return; }
   var serverResponse = xhReq.responseText;
   ...};
xhReq.send(null);
```

- Das X in AJAX
- Markup
  - Markup: Trennung Struktur - Inhalt
  - logische Struktur der Seite
  - Bsp: Überschriften, Absätze, Zitate, …
- XML: eXtensible Markup Language
  - Syntax für Markup
  - Semantik in XSL oder CSS
- Document Object Model DOM
  - baumartige Struktur der Dokumente
  - Zugriff auf Dokumenteninhalt (=Objekte)
  - Inhalt, Struktur, Stil
- AJAX
  - XML als ein Transfer-Format für Inhalt
  - Manipuliert DOM-Knoten
  - Einfügen, Löschen, Ändern
  - Browser 'rendert' Dokument

```
<table>
   <thead>
     <tr>
       <th>Vorname</th>
       <th>Name</th>
     </tr>
   </thead>
   <tbody>
     <tr>
       <td>Donald</td>
       <td>Duck</td>
     </tr>
   </tbody>
```

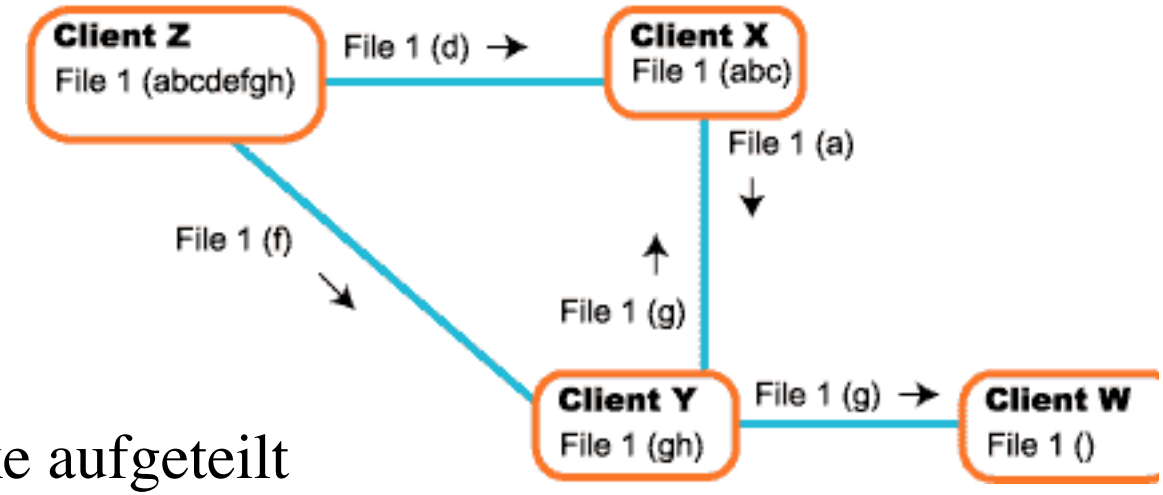


Quelle: de.wikipedia.de/wiki/Document_Object_Model

6.7 Peer to Peer Networks (www.openp2p.com,O'Reilly)

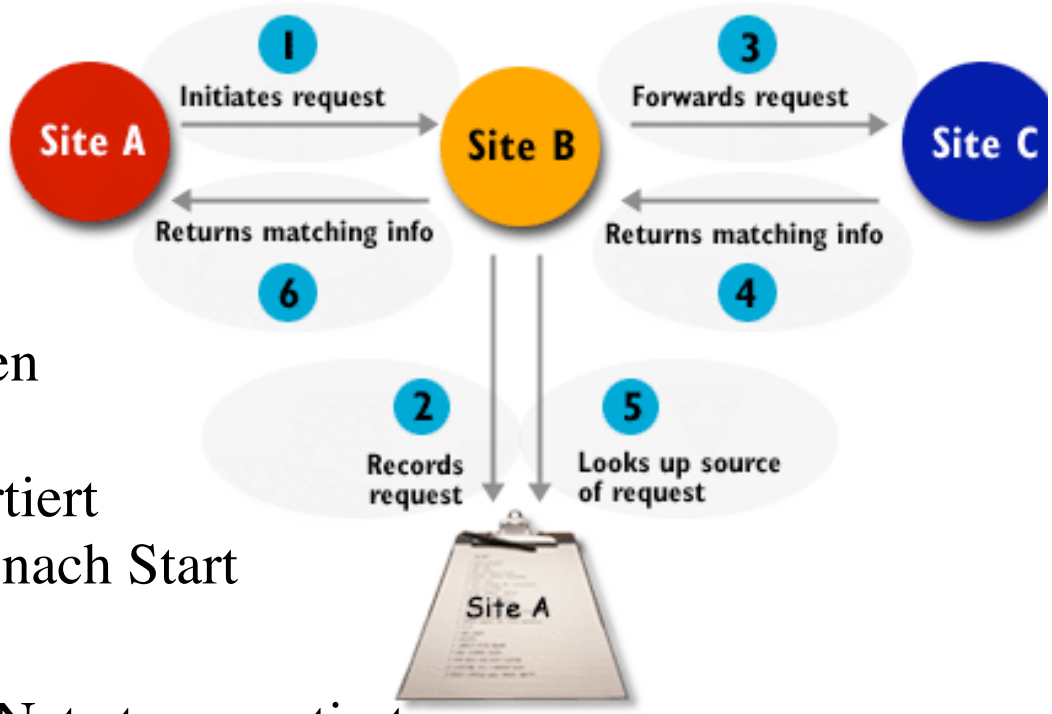
- Idee: Dezentrale Speicherung und Verteilung größter Datenmengen
  - Summe der PC-Festplatten
  - jeder PC kann auch Fileserver sein
  - Problem: Dateien *finden*
  - 'Decentralized Resource Discovery'
- Zentraler Verzeichnisdienst
  - Peers registrieren Dateien und Metadaten
  - Überlastproblem und Fehlertoleranz lösbar (-> Google)
  - juristisch verwundbar (-> Napster)
- Dezentrale Verzeichnisse
  - in jedem Peer suchen skaliert schlecht
  - Untermengen bilden und dort suchen
  - Mehrschichtarchitekturen (Superserver-Server-Peers)
  - Wie findet man Schicht-1-Server: well-known oder google
  - Schicht1-Server juristisch angreifbar?

- Daten übertragen (verteilen)
  - asynchron über einen Server
  - synchron zwischen Peers
  - sequentiell zusammenhängend (Filetransfer)
  - stückweise (chunks)



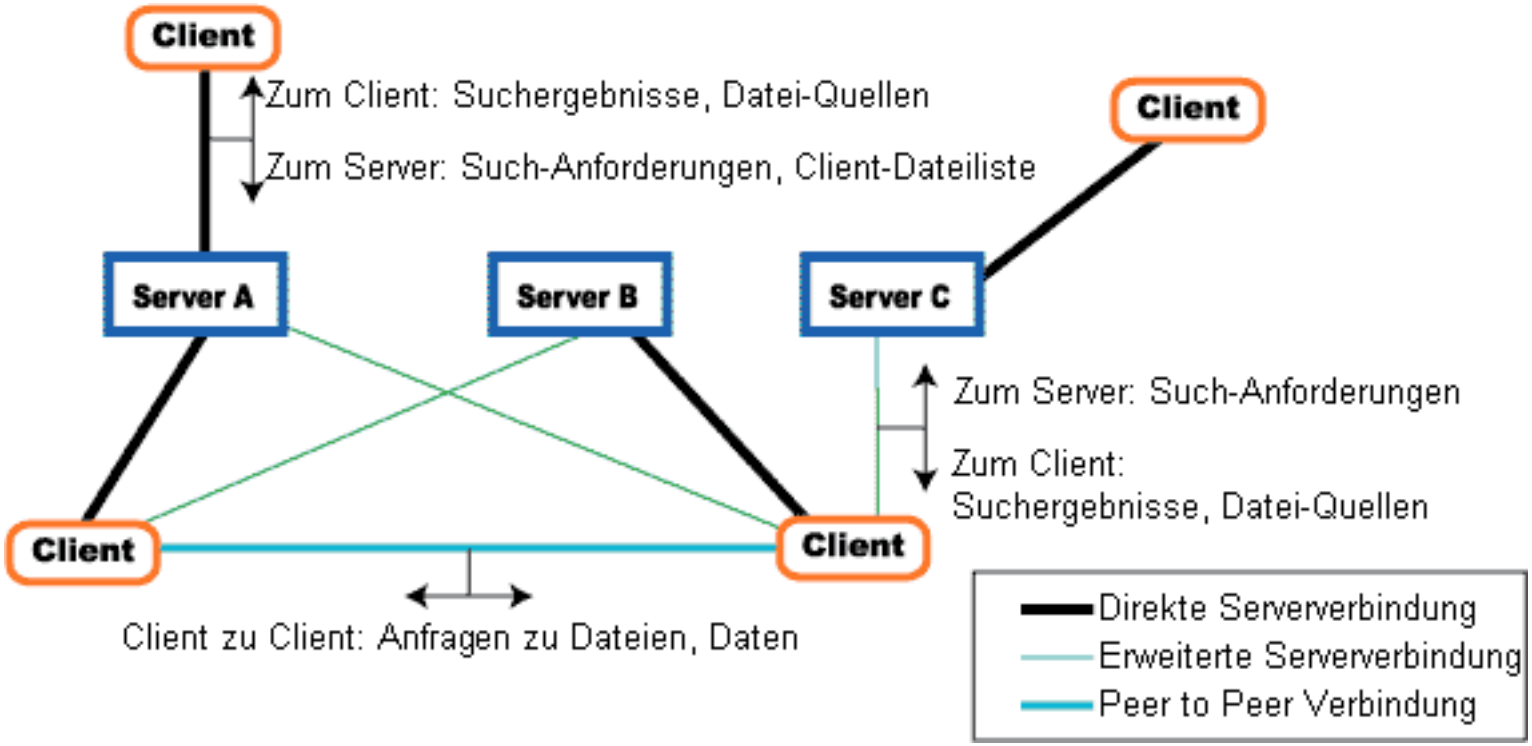
- Chunks
  - Datei wird in viele kleine Stücke aufgeteilt
  - paralleler Download verschiedener Stücke
  - sequentieller Download nichtzusammenhängender Stücke
  - Software setzt Stücke zusammen
  - gleicht heterogene Netzwerksturkturen aus
  - Fehlertoleranz durch breite Download-Basis
  - funktioniert nur bei identischen Quellen -> Hash pro Datei
  - auch bei unvollständigen Quellen …

- Gnutella
  - Peer = Client + Server (servent)
  - http-basiertes Protokoll
  - Rekursive Suche nach Peers und Dateien
  - MultiCast: 'Schneeballsystem'
  - auch Inhalt (=Dateien) wird so transportiert
  - Hostcache zum finden der ersten Peers nach Start
- Freenet [Ian Clarke, UoEdinburgh]
  - Dateien werden immer durch das Peer-Netz transportiert
  - Dateien migrieren beim Request
  - Point-to-Point-Topologie verlängert Suche
- FastTrack
  - kein zentraler Server
  - 'Supernodes': Peers mit guter Netzwerkanbindung und Hardware
  - Supernodes halten Index und Suchen
  - zentraler Server um Supernodes zu finden
  - Suche relativ schnell



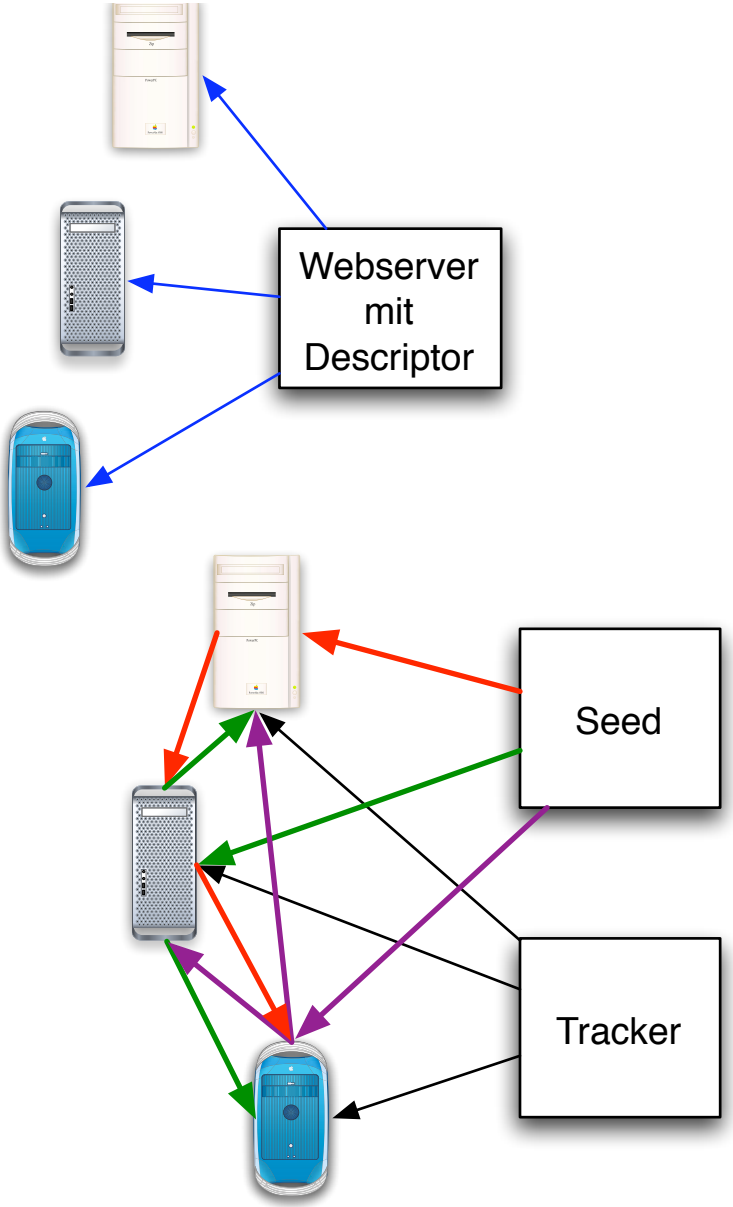
- eDonkey 2000
  - 2-Schicht Netzwerk: Server (dserver, Lugdunum) und Peers
  - Peers registrieren Files (Metadaten, Hash) bei Servern
  - Peers suchen im Servernetz nach Files
  - search-request(Attribute) – {(filename, hash, …)}
  - source-request(hash) – (Anzahl, {(IP,port)})
  - WebServer mit ed2k-Links: ed2k://

ed2k://|file|Suse.Linux.7.3.Professional.cd1.ShareReactor.bin|772636704|322c08442589c09fd2885a69980ff442|

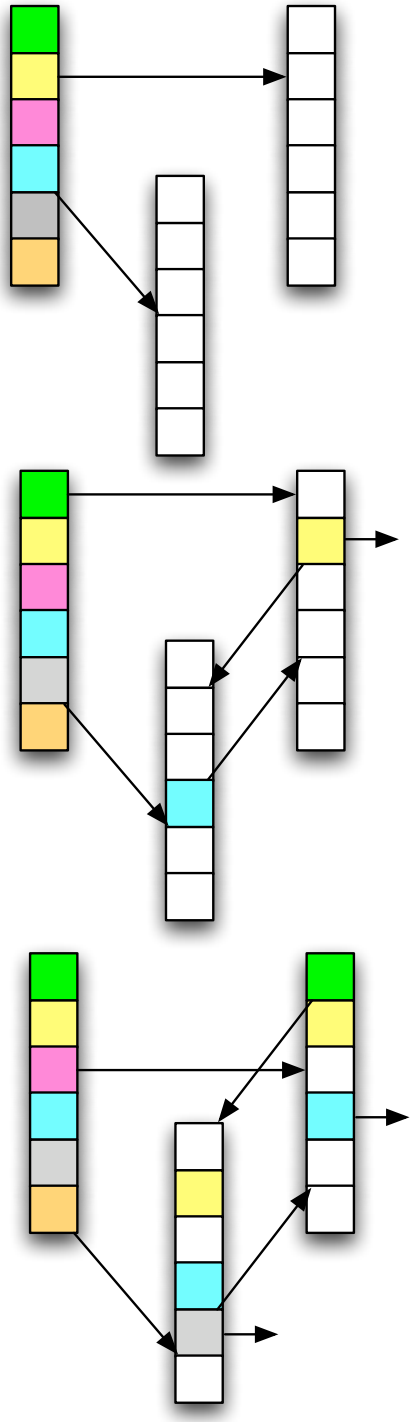


Quelle: thedonkeynetwork.com

- BitTorrent [Cohen, 2001]
  - dezentraler Verteildienst für Dateien
  - Klienten werden Teil des Fileservice
  - 'Übergabe der Datei an den Schwarm'
  - Dateien werden gestückelt transportiert
- Infrastruktur
  - Initial Seed, Seeds
  - Webserver mit Torrent Descriptor
  - Tracker
  - Clients
- Torrent Descriptor
  - Metainformation
  - Adresses eines Trackers
  - piece length
  - Liste mit Hashes für Dateistücke
  - vom 'Herausgeber' erstellt



- Tracker
  - Get enthält hash und eigene peer id
  - Antwort mit peers
  - mehrere (peer, IP/hostname, Port)
- Peer
  - sucht nach Torrent-Descriptor
  - fragt Tracker nach Peer/Seed
  - TCP-Verbindungen Peer to Peer
  - zufällig Stücke holen (index)
  - Stückes selbst bereithalten
  - geladene Stücke bekanntgeben
  - piece requests auf Vorrat
- Probleme
  - tracker pro Torrent zentral
  - Distributed Hash Tables als Ausweg
  - Verteilung beliebter Files skaliert gut
  - selten verlangte Files verlieren seeds
  - seed promotion Problem



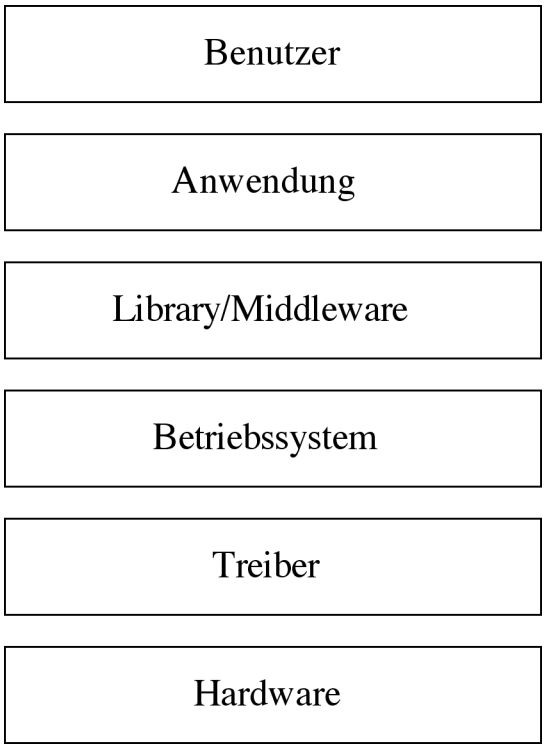
- Peer-Software sorgt für Fairness
  - Warteschlangen für Requests
  - Bewertungsfaktor(Anzahl eigene Files, Ressourceinvestition, …)
  - beeinflusst Warteschlangenposition
  - Freeloader
- Network <> Peer-Software
  - eDonkey – eDonkey, eMule, …
  - Gnutellea – LimeWire, Morpheus (?), BearShare
  - FastTrack – (Morpheus,) KaZaA, Grokster
  - Freenet

	aktive Peers?	Verzeichnis	Dateihaltung	Übertragung
Webserver	-	zentral	zentral	Datei
Napster	zentral implizit	zentral	verteilt	Datei
Gnutella		verteilt	verteilt	Datei
eDonkey	verteilte Server	verteilt	verteilt	Chunk
FastTrack	zentraler Server	SuperNodes	verteilt	Datei/Chunk
Freenet	verteilt	verteilt	transient verteilt	forwarding
BitTorrent	zentral/verteilt	verteilt, offen	transient verteilt	Chunk

7. Verteilte Systeme

Ein verteiltes System Ist eine Menge unabhängiger Computer, die den Benutzern als ein Einzelcomputer erscheinen. [Tanenbaum]

- Entwurf Verteilter Systeme
  - Komunikationsmodelle
  - Gruppenkommunikation
  - Integrationsniveau
  - Algorithmen
- Programmierung Verteilter Systeme
  - Synchronisation (Zeit?)
  - Koordination (Semaphore, Mutex)
  - Auswahlverfahren
  - Zuverlässigkeit
  - Programmiermodell, ACE

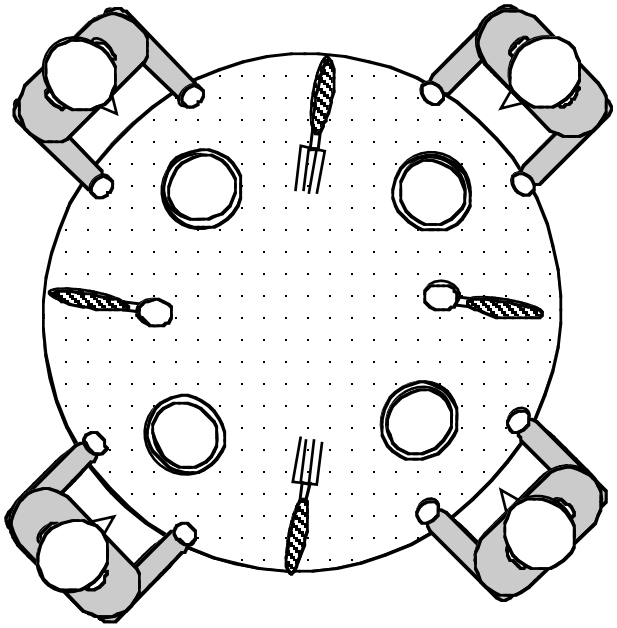


A distributed system is one on which I cannot get any work done because some machine I never heard of has crashed. [Leslie Lamport]

- Das zentrale Problem aller verteilten Systeme: Nebenläufigkeit
  - Concurrency

> Concurrency occurs when two or more execution flows are able to run simultaneously [Dijkstra]

  - Ereignisse beeinflussen sich *nicht* gegenseitig
  - d.h. nicht kausal abhängig
  - oft nicht gegeben
- Gleichzeitige Ausführung
  - viele Ausführungspfade
  - unterschiedlicher oder gleicher Code
  - konkurrieren um Ressourcen
  - Datenbanksätze
  - Hardware
- 'Synchronisation'
  - Koordination abhängiger Ereignisse
  - mutual Exclusion
  - neue Probleme: Deadlocks



- Beispiel: Sequentielles Programm A:
  - ergibt 30 als Resultat in der Variablen "c"

Statements

`a:= 0;`
`b:= 0;`
`a:= 10;`
`b:= 20;`
`c:= a+ b;`

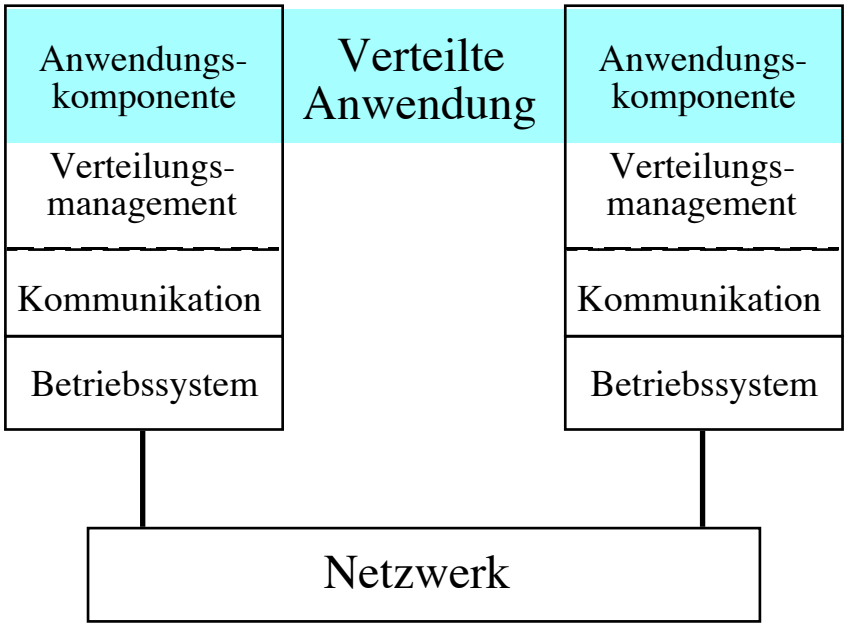
- Nebenläufige Ausführung ohne Synchronisierung
  - verschiedene Resultate möglich für c
  - 120 mögliche Permutationen (5*4*3*2)
  - zum Beispiel für drei Prozessoren sei c = { 0, 10, 20, 30 ... }

CPU #1	CPU #2	CPU #3
a:=0;	*b:=0;*	*a:=10;*
b:=20;	*c:=a+b;*	

7.1 Infrastruktur Verteilter Systeme

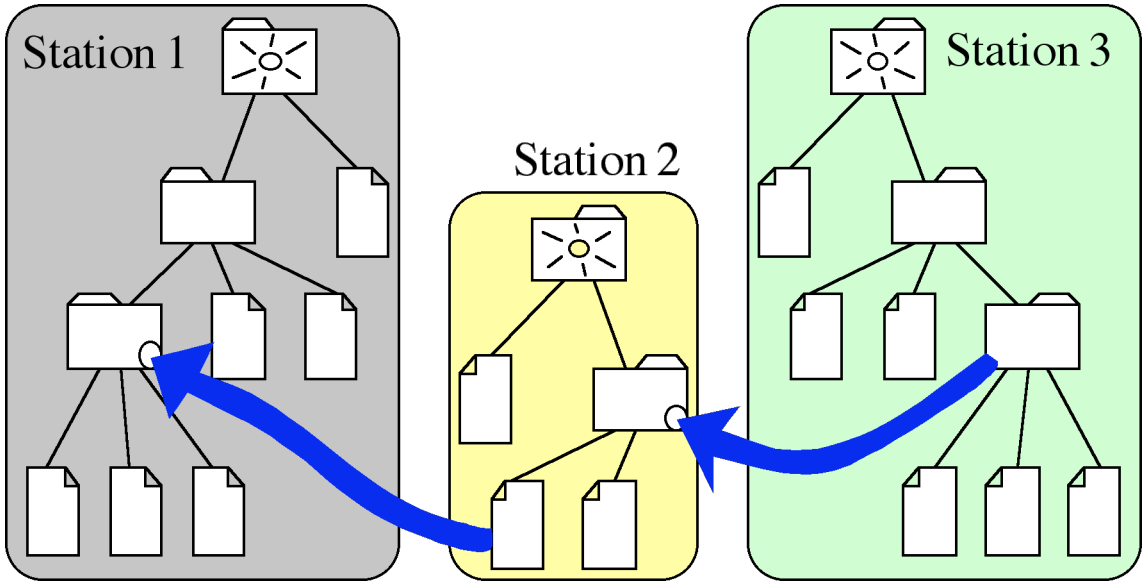
7.1.1 Do it Yourself

- Rechnernetzbasiert
  - insbesondere Internet
- Kopiersemantik (FTP, HTTP ...)
- Dienstfragmente (QoS, Naming, ..)
- Suboptimale Nutzung
  - Verbindungen …
- ALF: Application Layer Framing
  - gute Anpassung an das Problem
- Programmierung schwierig
  - Konsistenz, Nebenläufigkeit
  - viel Handarbeit



7.1.2 Verteilte Dateisysteme

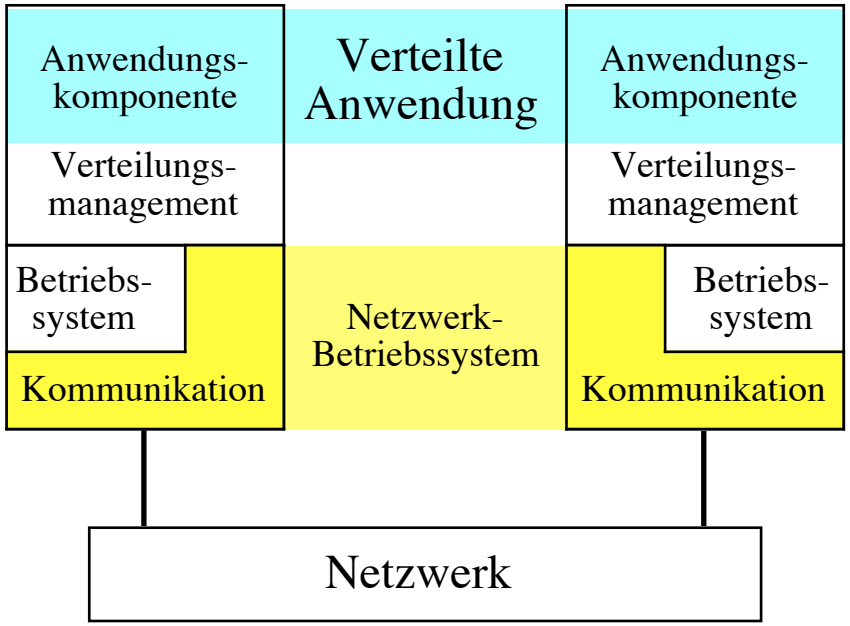
- Remote Disk
- Remote Files
- Zugriff auf entfernte Volumes
- Zugriff auf nicht-lokale Dateien
- Allgemein montierbare Verzeichnisbäume



- Replikation von Dateien und Verzeichnissen

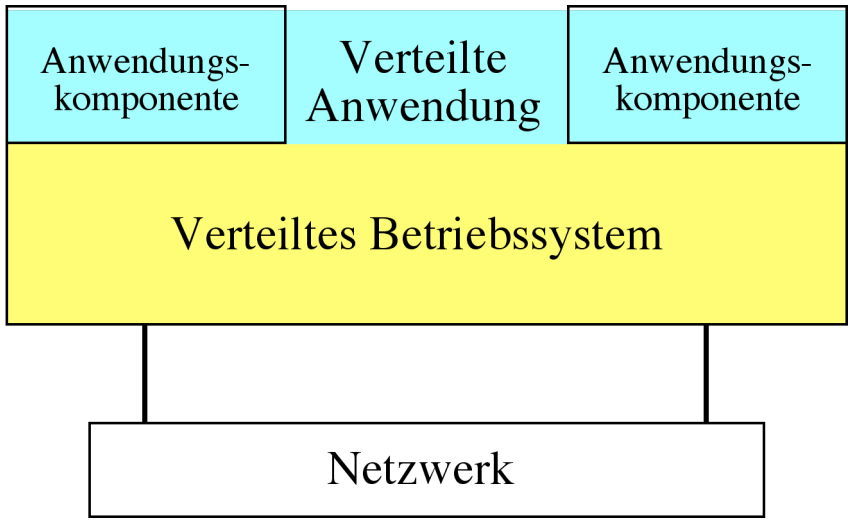
7.1.3 Netzwerkbetriebssysteme

- Weitgehend autonome Stationen
  - eigene Bedienerschnittstellen
  - eigene Betriebssystemkopie
  - ohne Netz arbeitsfähig
- Substitution lokaler Dienste
  - diskless Workstations
  - Remote Login
- Lokalisierbare Dienste im Netz
  - drucken und speichern
  - Rechner hochfahren
  - Namensdienste
  - Mail-Server
- Beispiele:
  - Novell Netware, Windows NT, Solaris, …
- Programmierung einfacher, Parallelisierungsprobleme ungelöst
- Betriebssystem nicht einfach, Semantikverlust



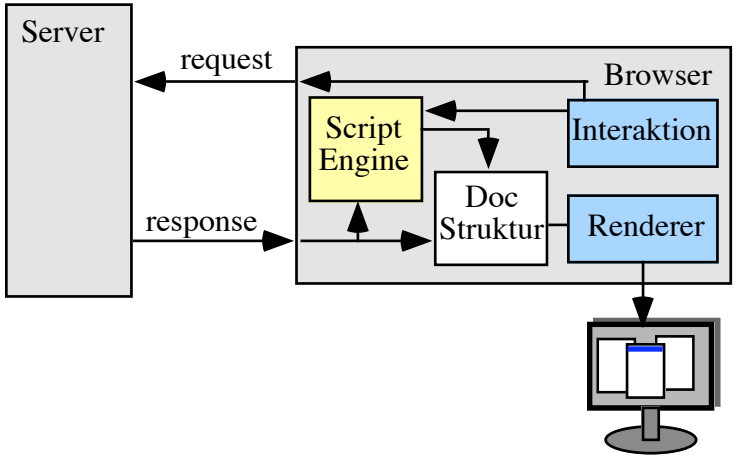
7.1.4. Verteilte Betriebssysteme

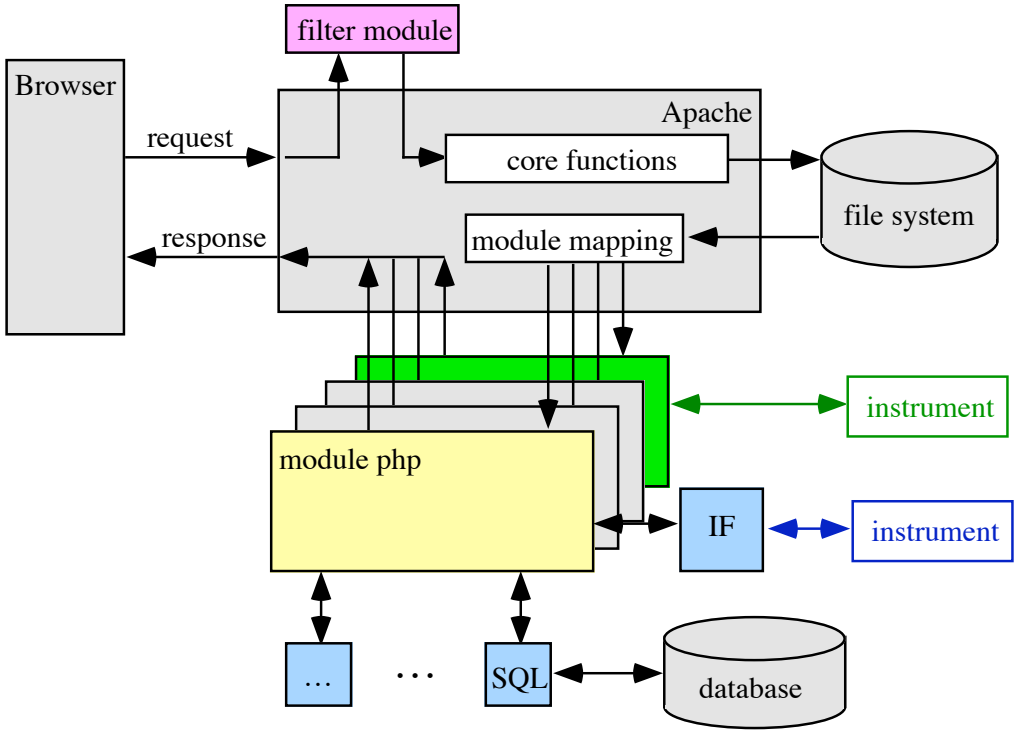
- Explizite Kommunikation
  - Datagramm-Nachrichten
  - Sockets und Streams
  - Entfernte Prozeduraufrufe
  - Verteilte Objekte & Methoden (RMI, DCOM)
  - Corba, Java Spaces (Jini) ...
- Implizite Kommunikation
  - transparente Prozeduraufrufe
  - transparente Variablenzugriffe
  - identisches API - lokal & im Netz
  - softwaregestützt (Compiler & Run-Time)
  - hardwaregestützt (MMU, Segmente, ...)



7.1.5 Web-Applikationen

- Client
  - Web-Browser
  - Webseiten mit aktivem Inhalt
  - Skripte und Plugins
- Server
  - z.B. Apache mit Plugins
  - Datenbank
  - Skripte (php, Ruby on Rails, …)
  - Beans-Server: Java, asp, .net
- Nebenläufigkeitsprobleme
  - meist durch Datenbank gelöst
  - evtl. Unterstützung im Server



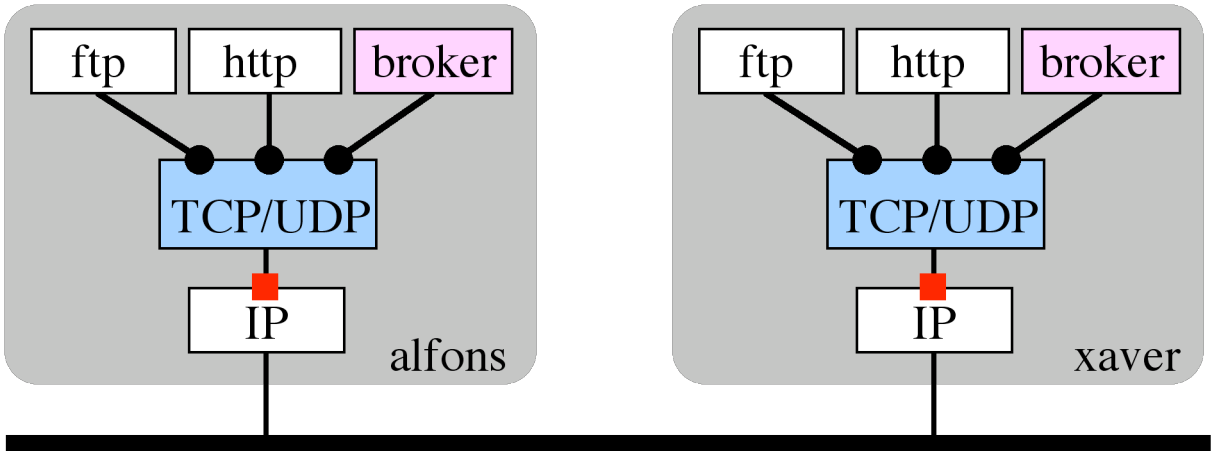


7.1.6 Anforderungen und Probleme

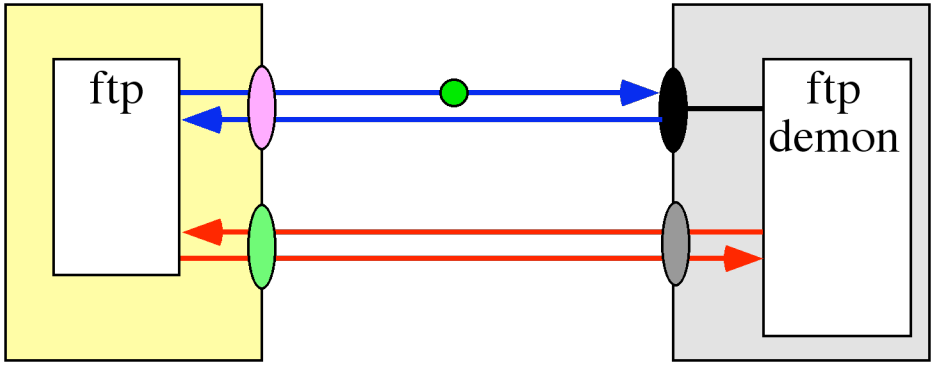
- Zugriffstransparenz
  - Einheitliches API
  - Single-System image
  - Logische Namen (nicht physische)
  - Yellow-Pages
- Ortstransparenz
  - Dateien und Verzeichnisse
  - Terminalzugang, Arbeitsumgebung
  - Rechenleistung
  - Hauptspeicher
  - Prozesse
- Dienstetransparenz
  - Drucker und Server
  - Namensdienst
  - Netzwerk-Browser ...
- Skalierbarkeit
  - Lastverteilung
  - Nutzung von Gerätepools
  - inkrementelle Erweiterung
  - additive Resourcenkumulation
- Fehlertoleranz
  - graduelle Leistungsabnahme bei Defekten
  - Partitionierung und Fusion
  - Transaktionssicherheit
  - Replikation
  - Migration
- Plattformintegration …
  - Formate und Byteordnung
  - Betriebssysteme
  - Hardware

7.2 Kommunikationsmodelle

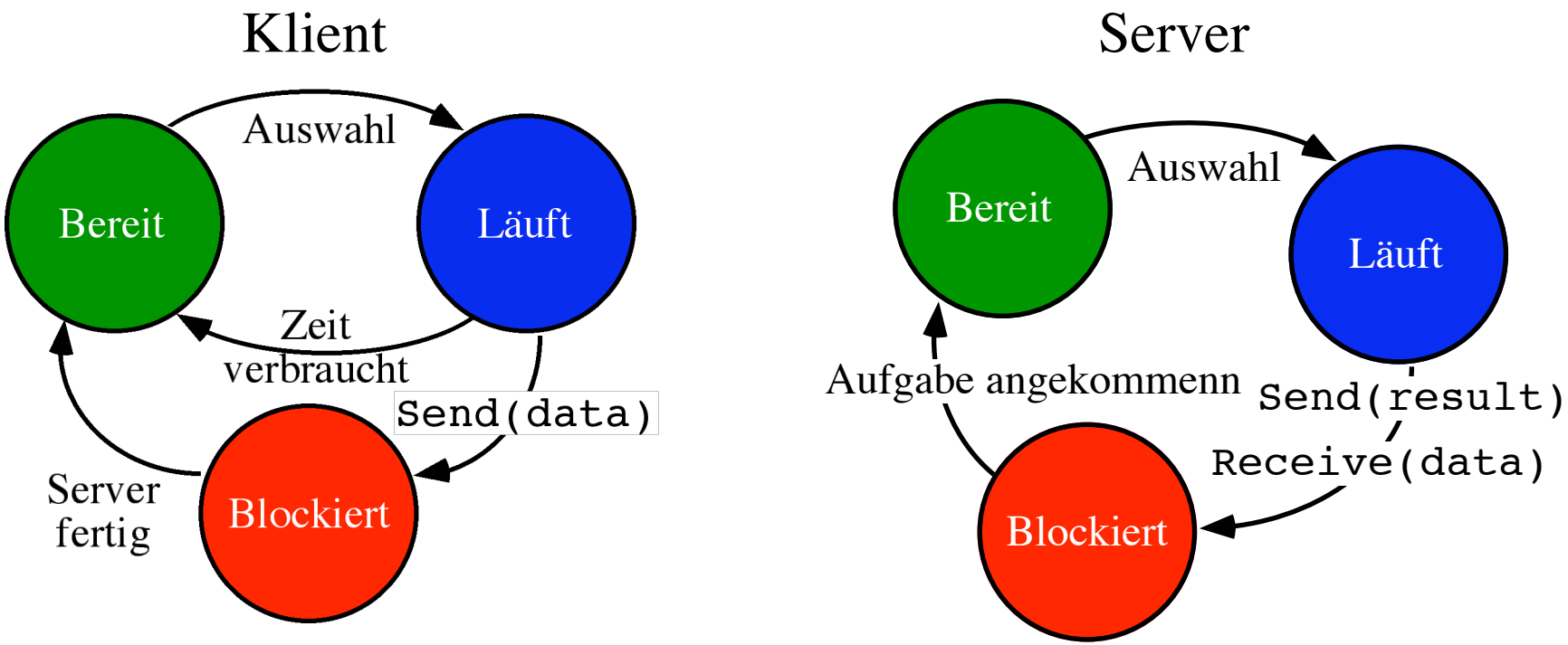
- Adressierung
  - -> Vorlesung Rechnernetze: Adressen
  - -> Vorlesung Rechnernetze: Ports



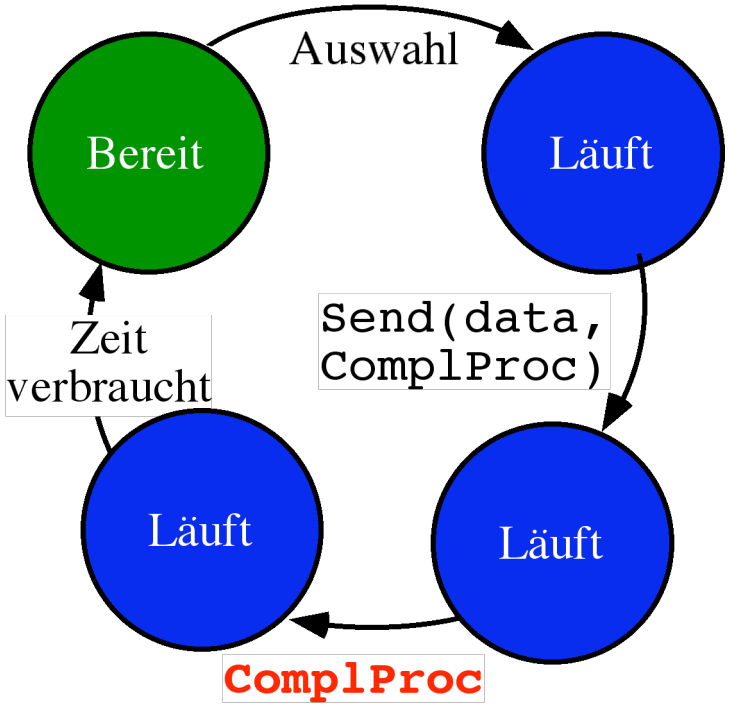
- Port-Auskunftssysteme: Broker
  - Teil von Corba, Jini, …



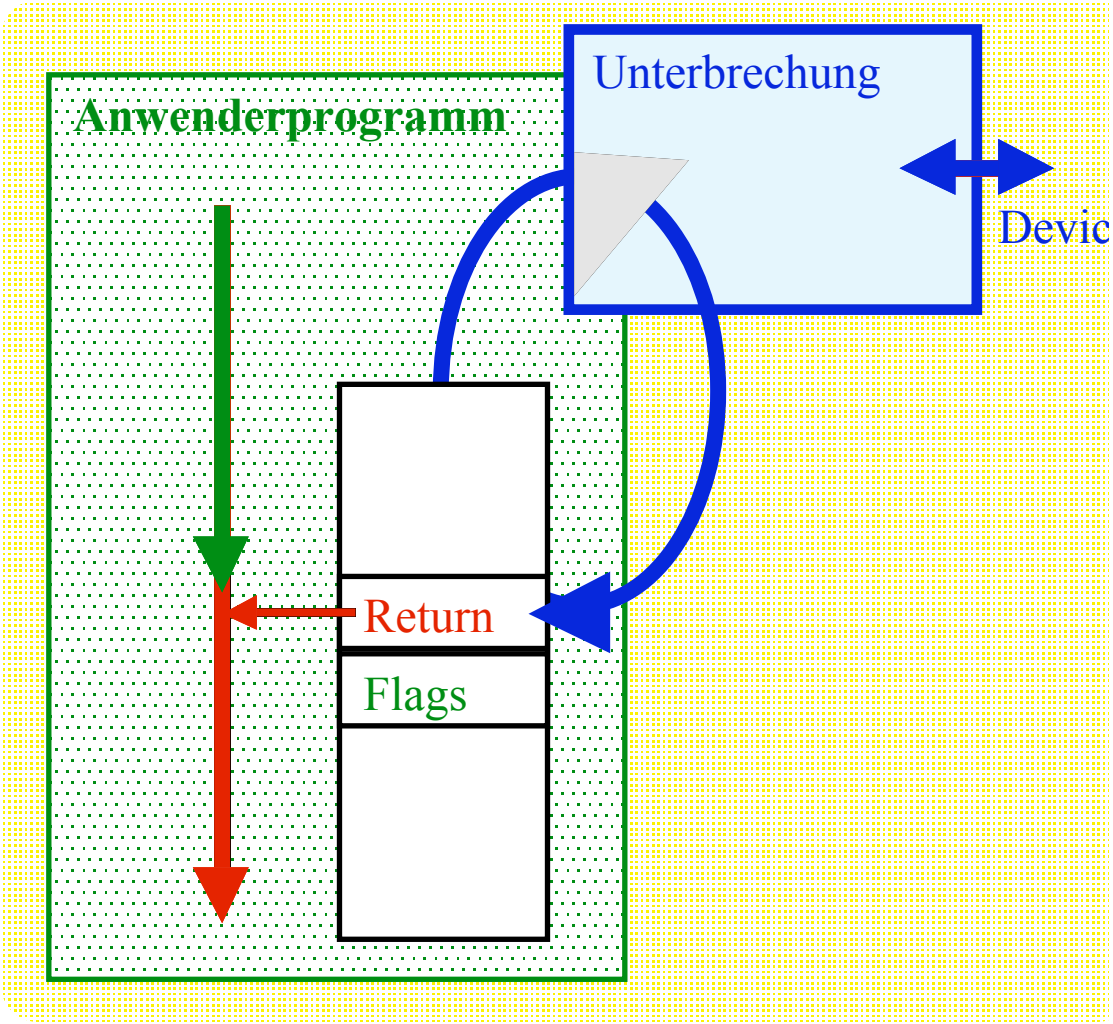
- Blockierung
  - Semantik der Diensteanforderung
- Serverprozess wartet mit `receive(data)`
- Synchrone Kommunikation
  - Prozess ruft Dienst mit `send(data)`
  - Prozess wartet bis Aufgabe erledigt
  - Prozess läuft weiter



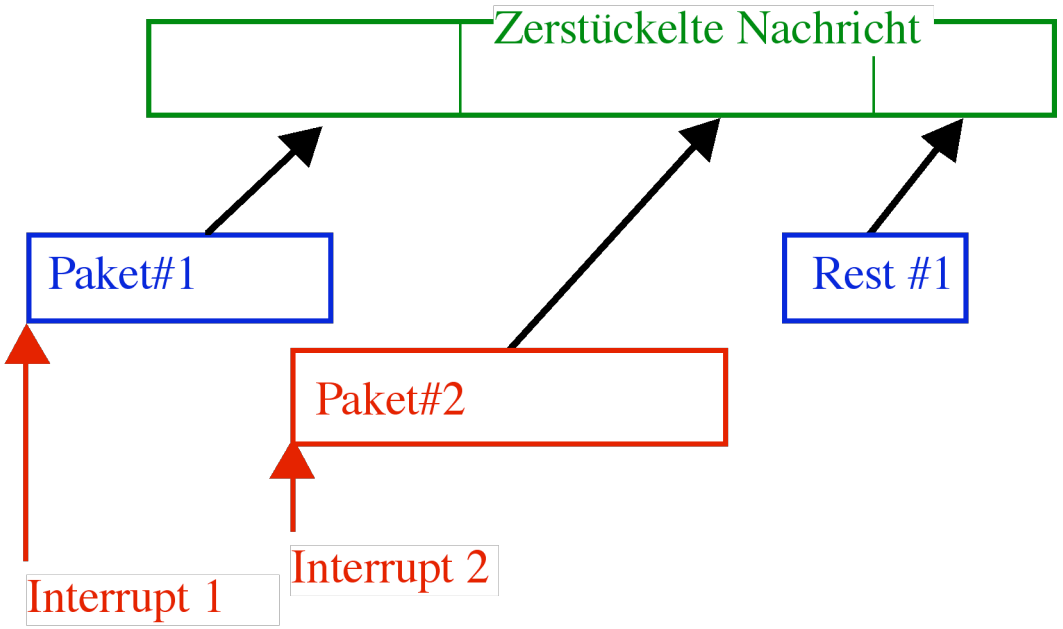
- Asynchrone Kommunikation
  - Prozess setzt Empfänger (Interrupt Server etc) auf
  - Prozess ruft Dienst mit `send(data,ComplProc)`
  - Prozess wartet bis `send` zurückkehrt
  - Prozess arbeitet weiter
  - Completion Prozedur wird gerufen im Interrupt

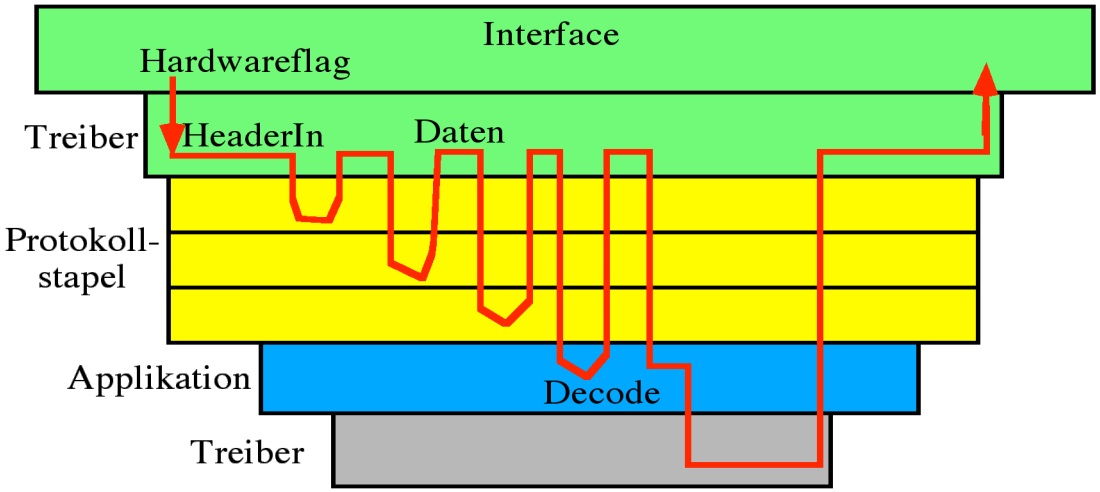


- Interrupt
- Programm unterbrochen
  - Aufgabe mit höherer Priorität
  - Zustand des unterbrochenen Programmes aufzeichnen
  - Zustand nach Abschluß der Unterbrechung restaurieren
  - Fortsetzung an Unterbrechungsstelle
- Behandlung von Fehlersituationen
  - Speicherschutzverletzung
  - Ungültige Maschineninstruktion
  - Paritätsfehler im Hauptspeicher
  - Arithmetische Fehler
  - Ausgelagerte Speicherseite
  - Debugging ...



- Externe Gerätepuffer[1]
  - Tastatur, serielle Datenleitung
  - Festplattenkontroller
  - Netzwerkkarte, …
- Softwareinterrupts
  - Schnittstelle zum BIOS/OS
- Interrupts maskieren
  - automatisch im Interrupt
  - Freigabe mit EOI
  - manuell (CLI und STI)
- Heraufarbeiten in Schichten





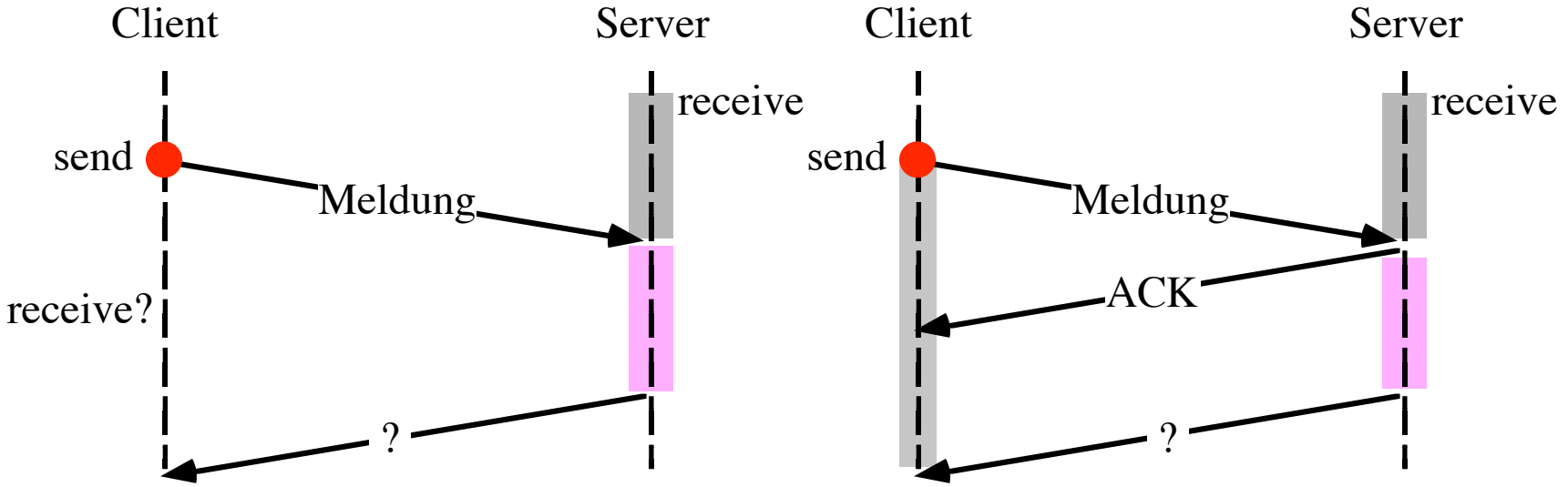
[1]

- Vorteile - Nachteile

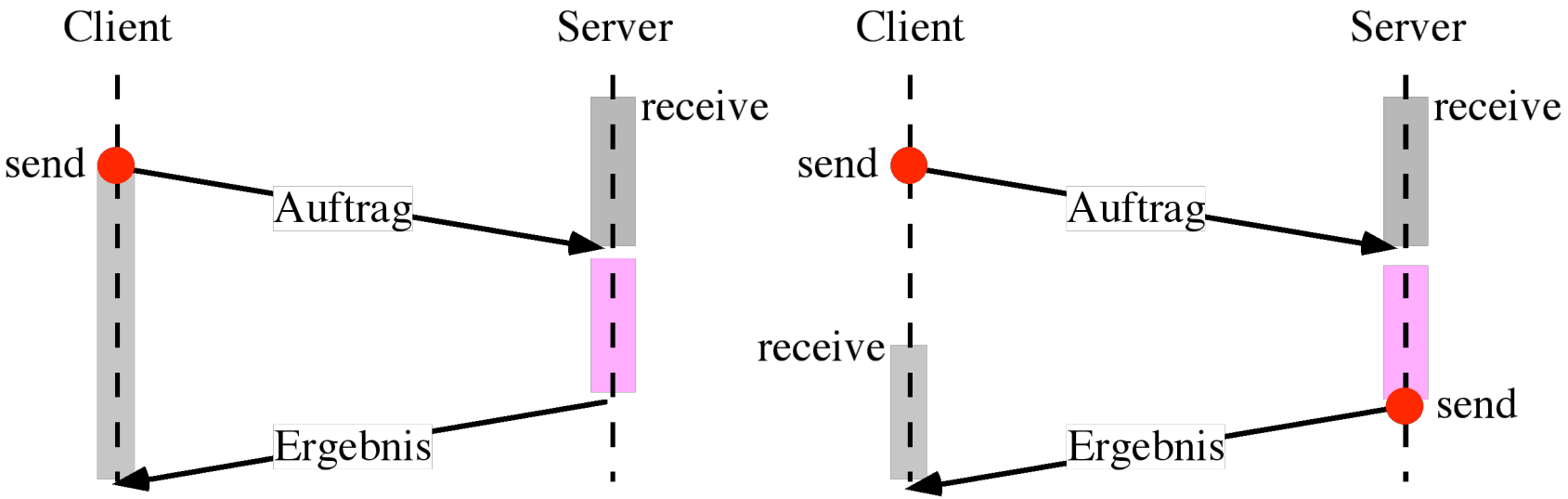
	Vorteile	Nachteile
Synchron	Synchronisation automatisch	Rechenzeitverschwendung
	keine Puffer	eigentlich sequentiell
	einfaches Programmiermodell	Verklemmung nicht ohne fremde Hilfe behebbar
		=> timeout - Interrupt
		=> asynchrone Routine
		System 'blockiert'
Asynchron	Entkopplung	Puffer nötig
		Pufferverwaltung …
	parallele Verarbeitung	komplexe Eventverschachtelung
		Programmieren schwerer
		Locking

	asynchron	synchron
meldungsbasiert	Datagramm	rendevous
auftragsbasiert	ARSI: async. Remote Service Invocation	SRSI: sync. Remote Service Invocation

- Meldungsorientiert
  - Request in Meldung schicken
  - auf Antwort warten
  - Parameter selber verpacken



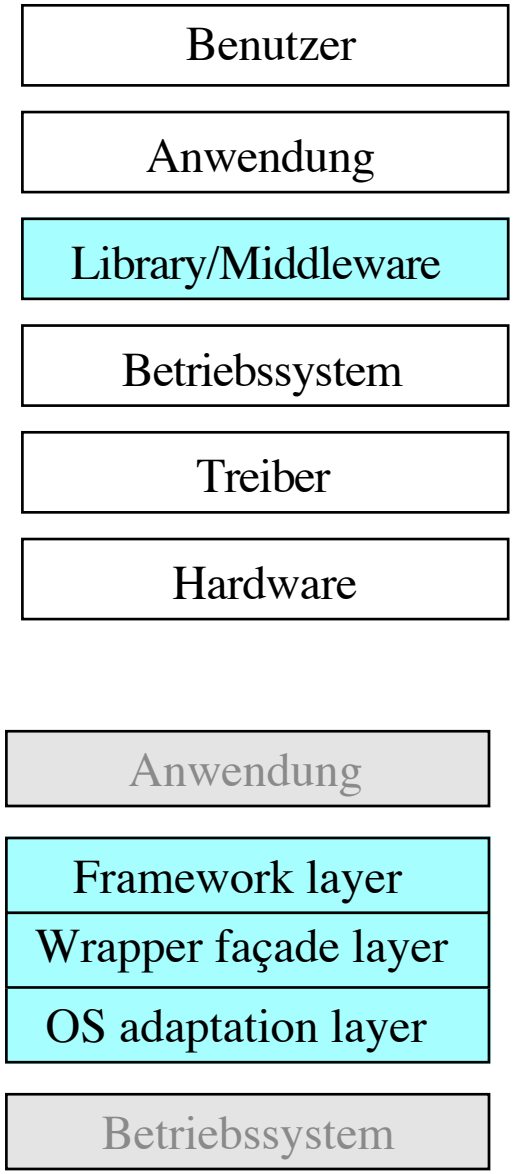
- Auftragsorientiert
  - ähnlich Prozeduraufruf
  - normal oder 'nebenläufig'
  - Parameterverpackung oft implizit
  - Versand oft implizit



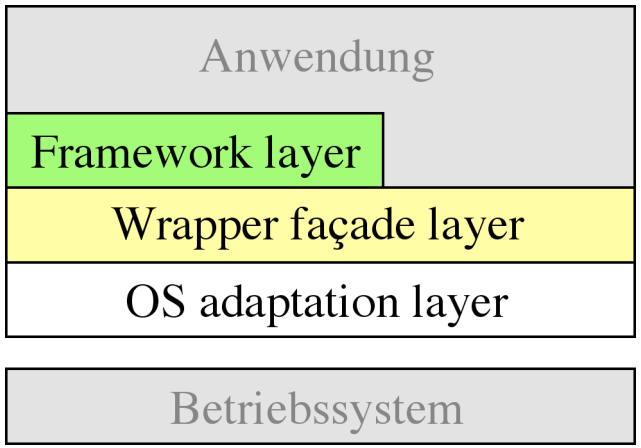
- Empfindlich gegen Fehler

Beispiel: Toolkit für Verteilte Systeme

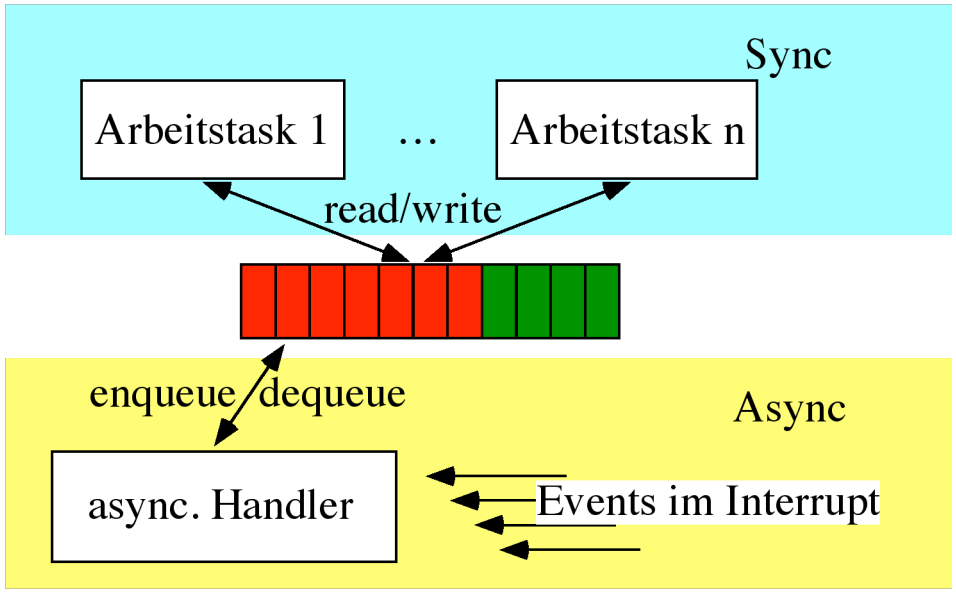
- ACE: Adaptive Communication Environment [Schmidt]
  - Netzwerkprogrammier-Library
  - Plattformabstraktion, Portabilität
  - 'Wrapper'
  - Framework-Metapher
  - Patterns
- Unterstützung für nebenläufiges Programmieren
  - Threads und Thread-Management
  - Reactor, Proactor
  - Pattern zur Synchronisation
- Wrapper für Socket API
  - Socket API in fast allen OS zentrales Netzwerk-API
  - betriebssytemspezifische Varianten
  - viele Konzepte in einem API
  - objektorientierter ACE-Wrapper für Socket API



- Weitere Wrapper 'Façades'
  - Event Demultiplex
  - Process, Threading
  - Synchronization: Guard, Mutex, Locks, Semaphor
- Events: Anreize für Programm
  - Bsp: Paket, Mouse-Click, Request
  - klassische Programme haben 'Eventloop'
  - Eventloop verteilt Events an Service-Funktionen
  - Eventloop auch ein Pattern
- Framework (nach Schmidt)
  - erhält Events
  - sucht und ruft Handler (dispatch)
  - Programm wird Handler-Menge

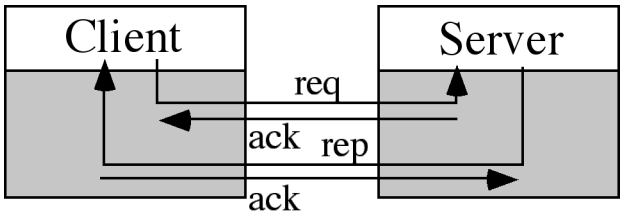


- Design-Patterns
  - Entwurfsmuster, Standard-Vorgehensweise
  - Schmidt et al. haben Patterns erarbeitet, in ACE implementiert
- Reactor
  - register_handler(handler: ACE_Event_Handler *, mask: …):int
  - Programm übergibt Kontrolle: handle_events(): synchron
- Proactor
  - asynchroner I/O
  - ACE hilft beim dispatch des Resultats
- Acceptor/Connector
  - acceptor: passiver Verbindungsaufbau
  - connector: aktiver Verbindungsaufbau
  - synchron und asynchron benutzbar
  - Timer möglich
- Task-Framework
  - half-sync/half-async
  - 'active object'

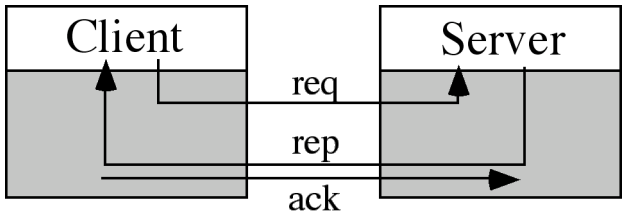


Client-Server

- Server
  1. wartet (synchron oder asynchron) auf Request (evtl. gepuffert)
  2. Request bearbeiten
  3. Ergebnis versenden
  4. weiter mit 1



- Client
  - sendet Request
  - erwartet Response
- Datenübertragung gesichert und verbindungsorientiert (TCP)
- Datagramme: ungesichert (UDP)
  - unzuverlässig
  - Request-Ack-Reply-Ack
  - Request-Reply(-Ack)



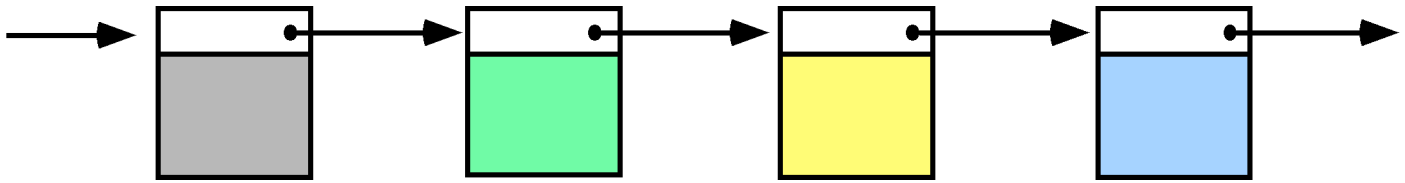
- Parameter meist selber verpacken
  - Standards für Format und Protokollfelder

• Implementierungsoptionen

Aufgabe			
Adressierung	Anschlußpunkt-adresse	Netzwerkadresse	Kommunikations-Adresse
Blockierung	Blockierend	asynchron mit Kernunterstützung	asynchron mit Interrupt
Puffern	ungepuffert	impliziter Puffer	verwalteter Puffer
Zuverlässigkeit	unzuverlässig	request-ack-reply-ack	request-reply-ack

• Protokollelemente
  - Request, Reply
  - Ack (,Nack), try again
  - Are you alive?, Alive
  - address unknown

- Daten in Nachrichten verpacken
  - Austauschformat
  - nur Konvertierung im Zielsystem
  - Verhandlung
- Homogener Fall
  - Elementare Datentypen und Records als Speicherabbild
  - Pointer ungültig
  - Dynamische Datenstrukturen linearisieren
- Lineare Listen
  - durchlaufen und elementweise verpacken
  - Zeiger werden nicht übertragen

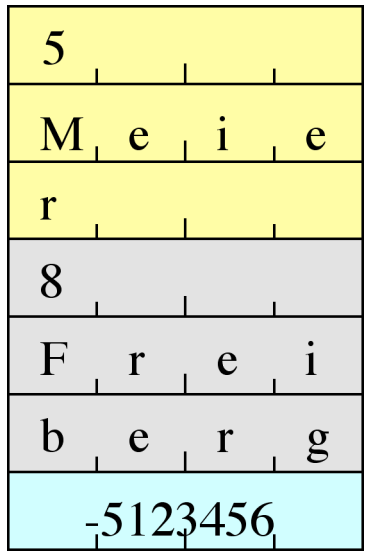


- Bäume
  - durchlaufen und elementweise verpacken
  - in-order, post-oder, pre-order?
  - allgemeine Graphen?

- Sun XDR (eXternal Data Representation, [RFC 1832])
  - für normale Datentypen
  - keine Typinformation in der Nachricht
  - ASCII, Integer, U…, …
  - Arrays und Strings mit Längenfeld
- CORBA
  - CDR: Common Data Representation
  - Interface Definition Language: Code generieren
- Java Object Serialization
- Mach: Matchmaker mit type-tags
- Xerox Courier

```
Person : TYPE = RECORD [
    name, wohnort : SEQUENCE OF CHAR;
    kontostand : CARDINAL                    ]
```

- Abstract Syntaxt Notation ASN.1 [CCITT, 1988]
  - Definition von Datentypen in der Nachricht
  - implizite Typen oder type-tags
  - siehe Vorlesung Kommunikationsdienste



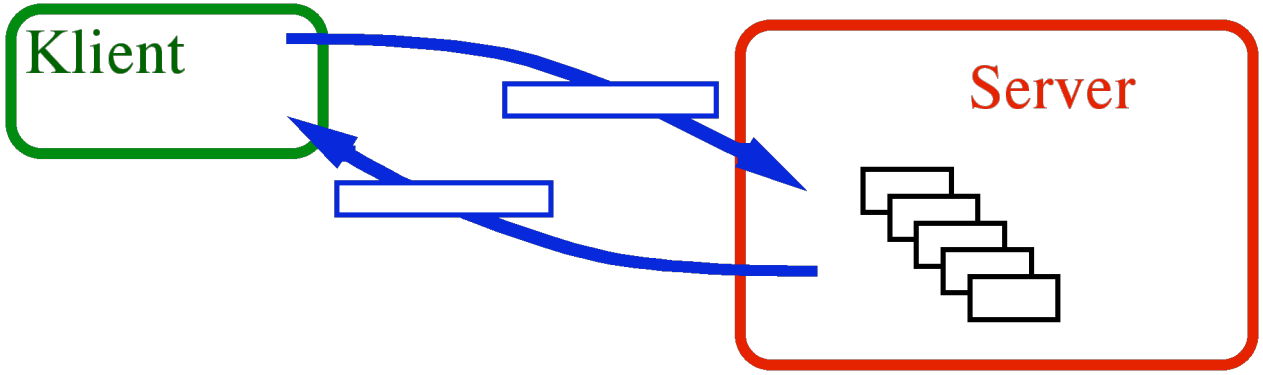
- Marshalling
  - Vorgang des Ein- und Auspackens
  - in der verteilten Anwendung

```
char *name = "Meier", *wohnort = "Freiberg";
char kontostand = -5123456;
sprintf(message,"%d %s %d %s %d",
    strlen(name),name,strlen(wohnort),wohnort,kontostand);
```

  -> Ausgabe: 5 Meier 8 Freiberg -5123456
- Automatische Generierung des Interfacecodes
  - Beschreibungssprache
  - Präpozessor
- Aufwand oft beträchtlich

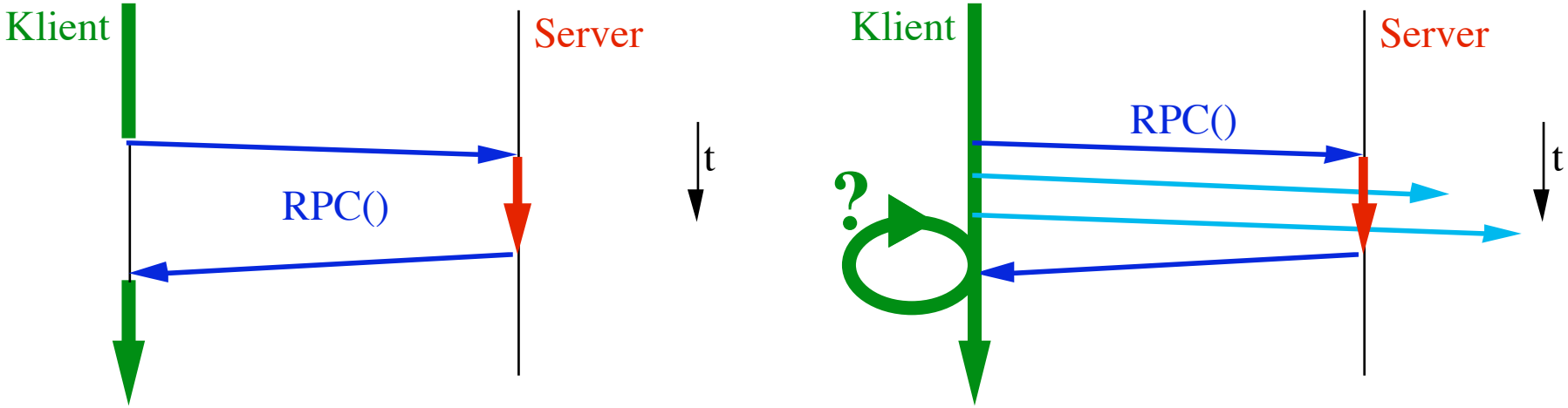
Remote Procedure Call

- Nachrichtenbasierte Kommunikation zwischen Knoten im Netz



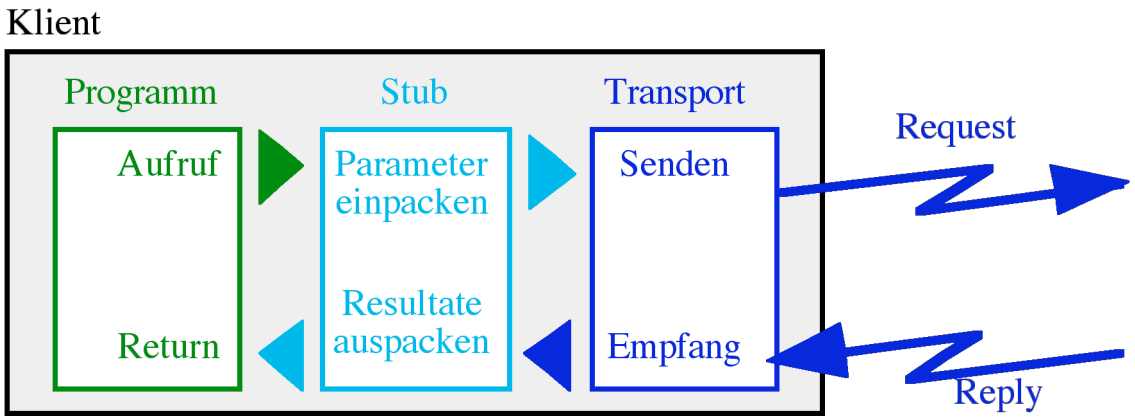
- Teilweise vergleichbar einem lokalen Prozeduraufruf
  - Parameter an die ferne Prozedur übergeben
  - Resultate an Klienten zurückliefern (Return)
  - eventuell warten
- Probleme im Klienten-Programm
  - globale Variablen
  - äußere Prozeduren
  - Zeiger auf Datenobjekte im Heap
  - komplexe Datenstrukturen schwierig

- Synchroner RPC
  - blockierend
  - kein explizites Warten auf Antwort
  - sicherheitshalber Time-Out bei Netzstörung ...



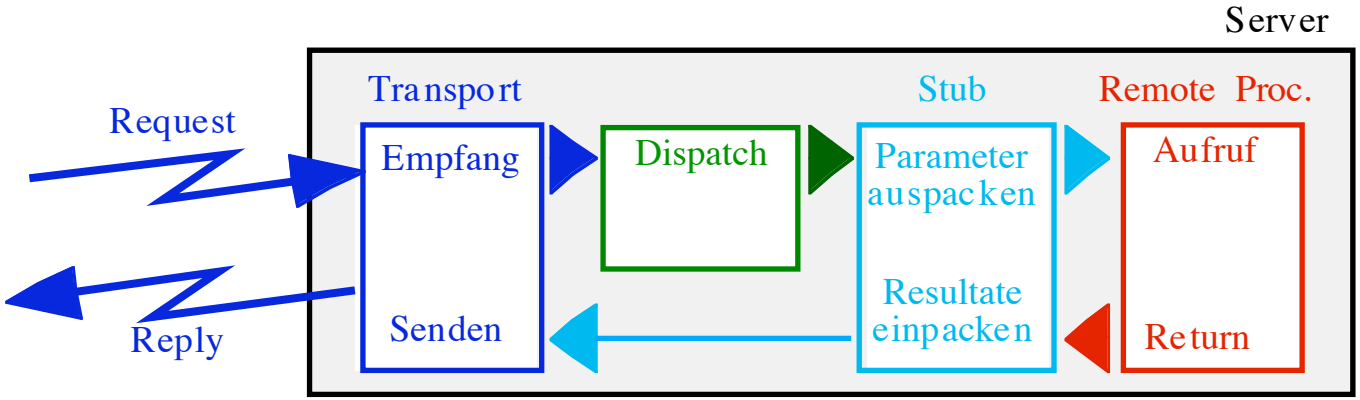
- Asynchroner RPC:
  - Klient läuft weiter
  - kann weitere RPCs absetzen
  - explizites Abfragen, ob Antwort schon da
  - oder "Completion routine" oder HW-Interrupt ...

- Besondere Programmiersprachen
  - integrierter RPC
  - Cedar, Argus, …
- Interface Description Languages
  - Präprozessor
  - XDR, ANSA Testbench, Matchmaker
  - Signatur für Prozeduren
- Software im Klienten-Rechner
  - "Stub" als Rumpfprozedur und lokaler Platzhalter



  - Verpacken der Daten für den Versand

- Software in der Server-Maschine
  - Dispatcher: Identifikation der gewünschten Prozedur
  - lokale Datendarstellung der Parameter (auspacken)

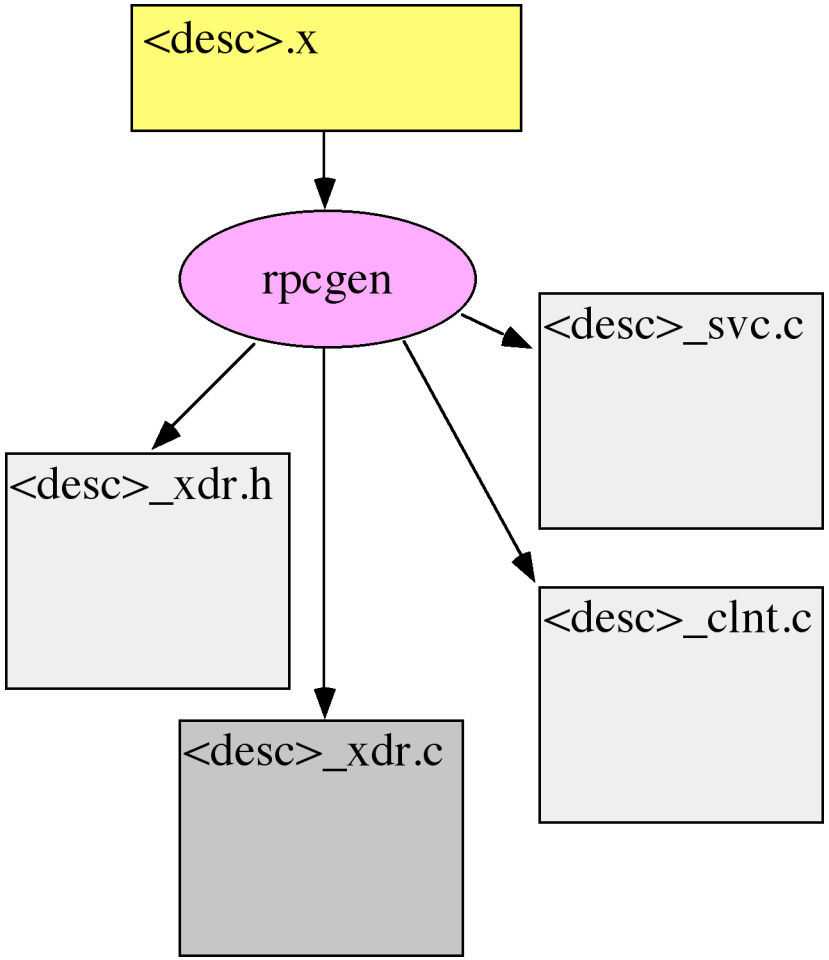


- Rückgabe der Resultatparameter
  - netzkonforme Darstellung herstellen (einpacken)
  - an Transportsystem übergeben
- Zuverlässigkeit

retransmit request	duplicate filtering	Reaktion	Semantik
Nein	-	-	maybe
Ja	Nein	re-execute	at-least-once
Ja	Ja	re-transmit	at-most-once

- RPC/XDR-Beispiel
- Schnittstellenbeschreibung: add.x

```
struct i_result {
    int x
    };

struct i_param {
    int i1;
    int i2;
    };

program ADD_PROG {
    version ADD_VERS {
        i_result ADDINT(i_param) = 1;
        } = 1;
    } = 21111111;
```



- Generiertes Headerfile: add.h
  - in Server und Klient-Programm einbinden

```
typedef struct {
    int x
    } i_result;

bool_t xdr_i_result ();

typedef struct {
    int i1;
    int i2;
    } i_param;

bool_t xdr_i_param ();

#define ADD_PROG ((u_long) 21111111)
#define ADD_VERS ((u_long) 1)
#define ADDINT   ((u_long) 1)

extern i_result *addint_1();
```

• XDR-Konvertierungsroutinen: add_xdr.c (3)

```
#include <rpc/rpc.h>
#include "add.h"

bool_t xdr_i_result (xdrs, i_resultp)
    XDR *xdrs;
    i_result *i_resultp;
{
    if (!xdr_int (xdrs, &i_resultp)) return (FALSE);
    return (TRUE);
}

bool_t xdr_i_param (xdrs, i_paramp)
    XDR *xdrs;
    i_param *i_paramp;
{
    if (!xdr_int (xdrs, &i_paramp->i1)) return (FALSE);
    if (!xdr_int (xdrs, &i_paramp->i2)) return (FALSE);
    return (TRUE);
}
```

- Client-Stub. add_clnt.c (2)

```
/* Please do not edit this file.
 * It was generated using rpcgen */

#include <rpc/rpc.h>
#include <sys/time.h>
#include "add.h"

/* Default timeout can be changed using clnt_control() */
static struct timeval TIMEOUT = {25, 0};

i_result *addint_1(i_param *argp, CLIENT *clnt) {
   static i_result res;

   bzero ((char *)&res, sizeof(res));
   if (clnt_call (clnt,
                  ADDINT,
                  xdr_i_param, argp,
                  xdr_i_result, &res,
                  TIMEOUT)   != RPC_SUCCESS) {
       return (NULL);
       }
   return (&res);                                  }
```

- Server-Schleife: add_svc.c (4)

```
#include <stdio.h>
#include <rpc/rpc.h>
#include "add.h"
static void add_prog_1();

main() {
    SVCXPRT *transp;

    (void) pmap_unset (ADD_PROG, ADD_VERS);
    transp = svcudp_create(RPC_ANYSOCK, 0, 0);
    if (transp == NULL) {
       (void) fprintf (stderr, "cannot create udp service.\n");
       exit (1);
       }
    if (!svc_register (transp, ADD_PROG, ADD_VERS,
         add_prog_1, IPROTO_UDP)) {
       (void) fprintf (stderr,"unable to register.\n");
       exit (1);
       }
    svc_run();
    exit(1); /* not reached */                }
```

```
void add_prog_1 (struct svc_req *rqstp, SVCXPRT *transp) { (5)
   union {
      i_param addint_1_arg;
   } argument;
   char *result;
   bool_t (*xdr_argument)(), (*xdr_result)();
   char *(*local)();

   switch (rqstp->rq_proc) {
      ...
      case ADDINT:
         xdr_argument = xdr_i_param;
         xdr_result = xdr_i_result;
         local = (char *(*)()) addint_1;
         break;
      ...
      }
   bzero ((char *)&argument, sizeof(argument));
   svc_getargs (transp, xdr_argument, &argument);
   result = (*local)(&argument, rqstp);
   svc_sendreply (transp, xdr_result, result);
   }
```

- Server-Prozedur (selbst zu programmieren) (6)

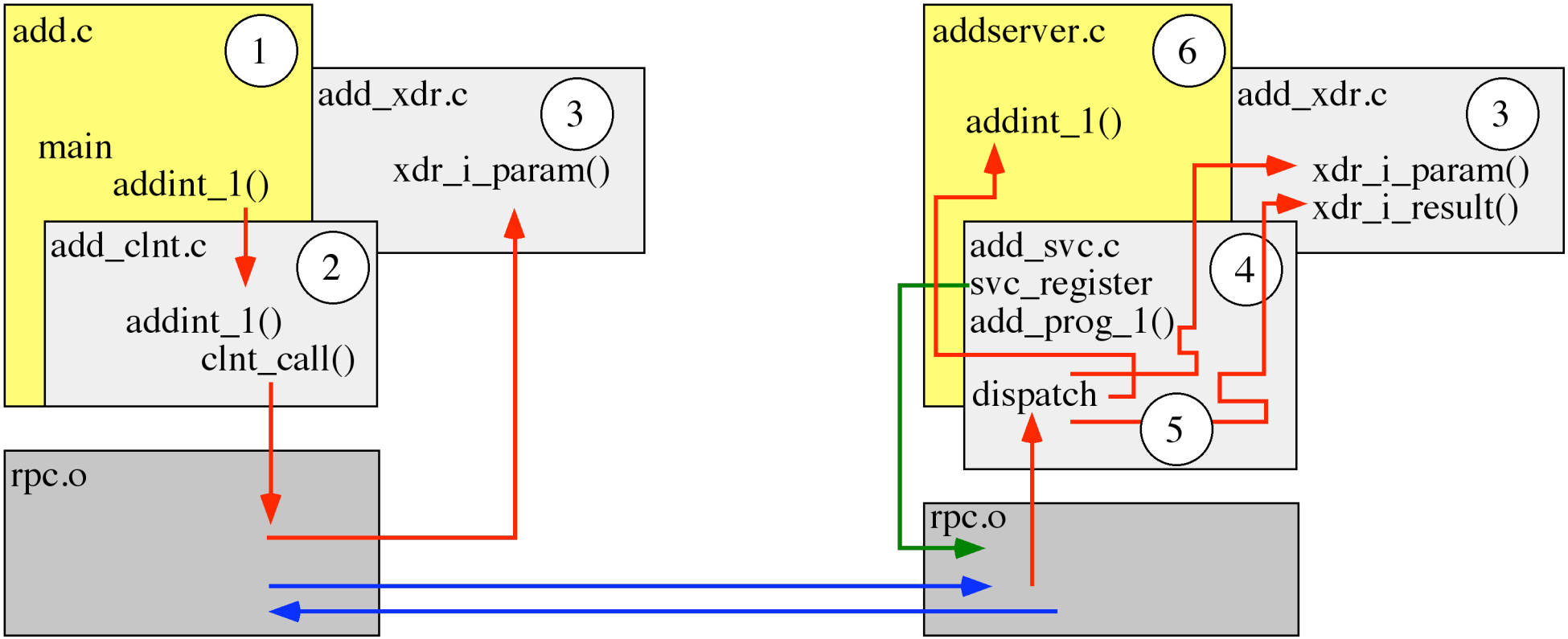
```
#include <rpc/rpc.h>
#include "add.h"
i_result *addint_1(i_param *p) {
    static i_result result;
    result.x = p->i1 + p->i2;
    return (&result);
}
```

- Client-Hauptprogramm (1)

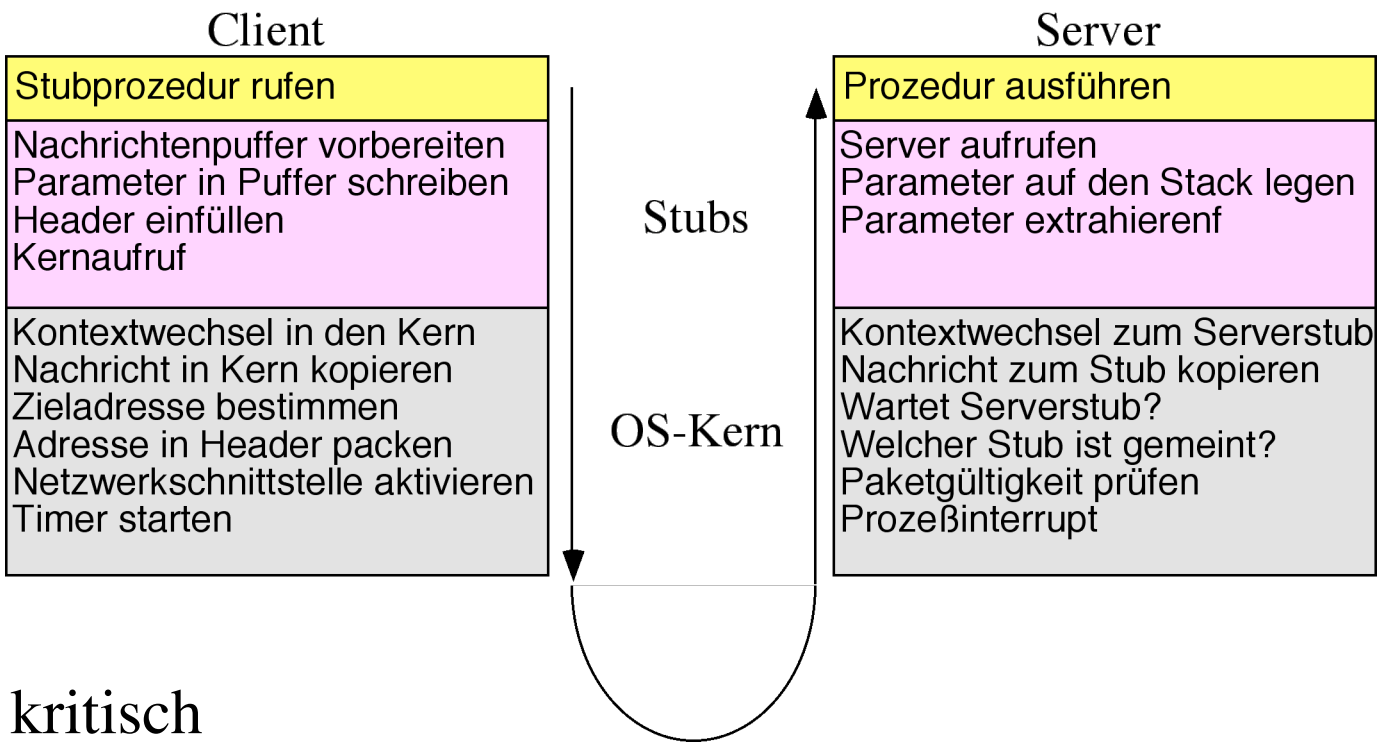
```
main (argc, argv)
    int argc; char *argv[];
{   CLIENT *cl;
    char *server;
    i_param parameter;
    i_result *ergebnis;

    server = argv[1]; /* Serveradressierung durch 1. Parameter */
    cl = clnt_create (server, ADD_PROG, ADD_VERS, "udp");
        /* Fehlerbehandlung? */
    parameter.i1 = 12; parameter.i2 = 30;
    ergebnis = addint_1 (&parameter, cl); /* transparency? */
    printf ("Summe ist %d\n", ergebnis->x);                }
```

- Ein Ausschnitt aus xdr.h

```
enum xdr_op {
  XDR_ENCODE = 0,
  XDR_DECODE = 1,
  XDR_FREE = 2
};
typedef struct XDR XDR;
struct XDR
{ enum xdr_op x_op;    /* operation; fast additional param */
  struct xdr_ops
  { bool_t (*x_getlong) (XDR *_xdrs, long *_lp);
    /* get a long from underlying stream */
    bool_t (*x_putlong) (XDR *_xdrs, _const long *_lp);
    /* put a long to " */
    bool_t (*x_getbytes) (XDR *_xdrs, caddr_t _addr, u_int _len);
    /* get some bytes from " */
    bool_t (*x_putbytes) (XDR *_xdrs, _const char *_addr, u_int
_len);
  /* more functions ... */
  } *x_ops;
/* ... */
};
```

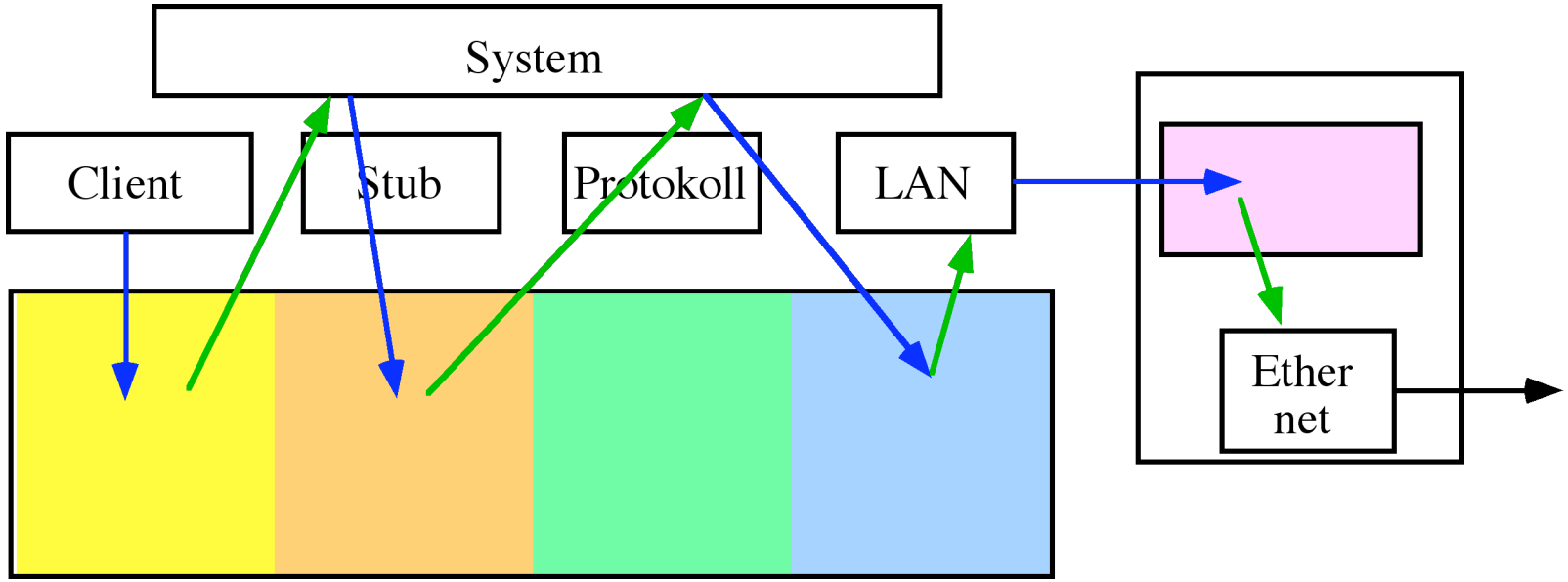

add.c
1
main
addint_1()
add_xdr.c
3
xdr_i_param()
add_clnt.c
2
addint_1()
clnt_call()
rpc.o
addserver.c
6
addint_1()
add_xdr.c
3
xdr_i_param()
xdr_i_result()
add_svc.c
svc_register
add_prog_1()
4
dispatch
5
rpc.o

- Aufwand
  - Marshalling
  - Paketisierung
  - Protokoll
  - Dispatch
  - Prozeßwechsel

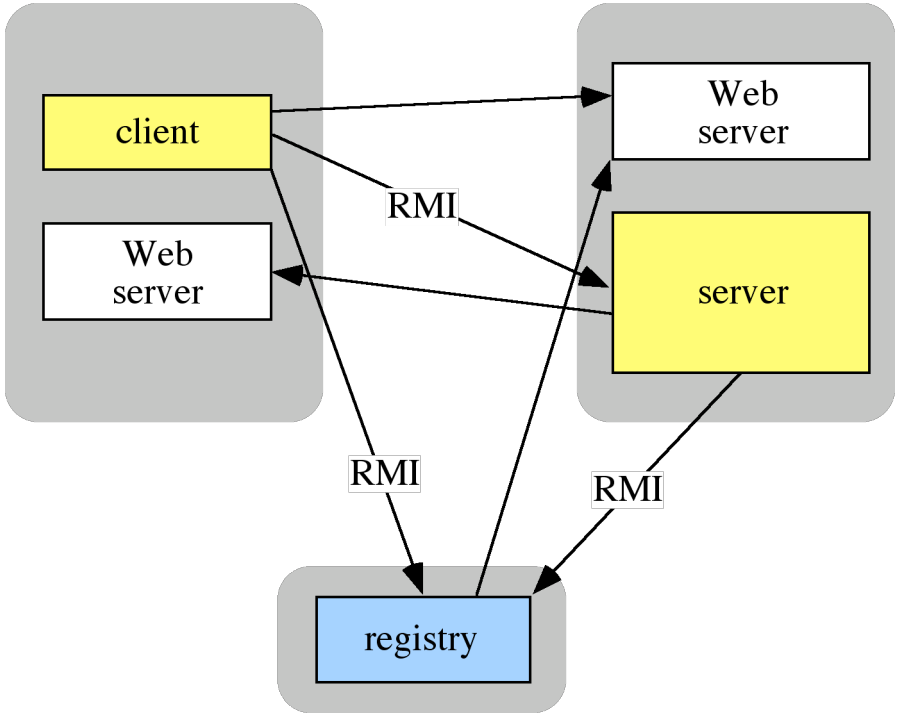


- Anzahl Datenkopien kritisch

- Ausführungsfluß im Schichtenmodell
  - Applikation
  - Stub
  - Systemdienst: Socket-Layer
  - Protokoll
  - Systemdienst: BlockCopy/Gather
  - Ethernet (Segmentation)
  - System: Scheduler

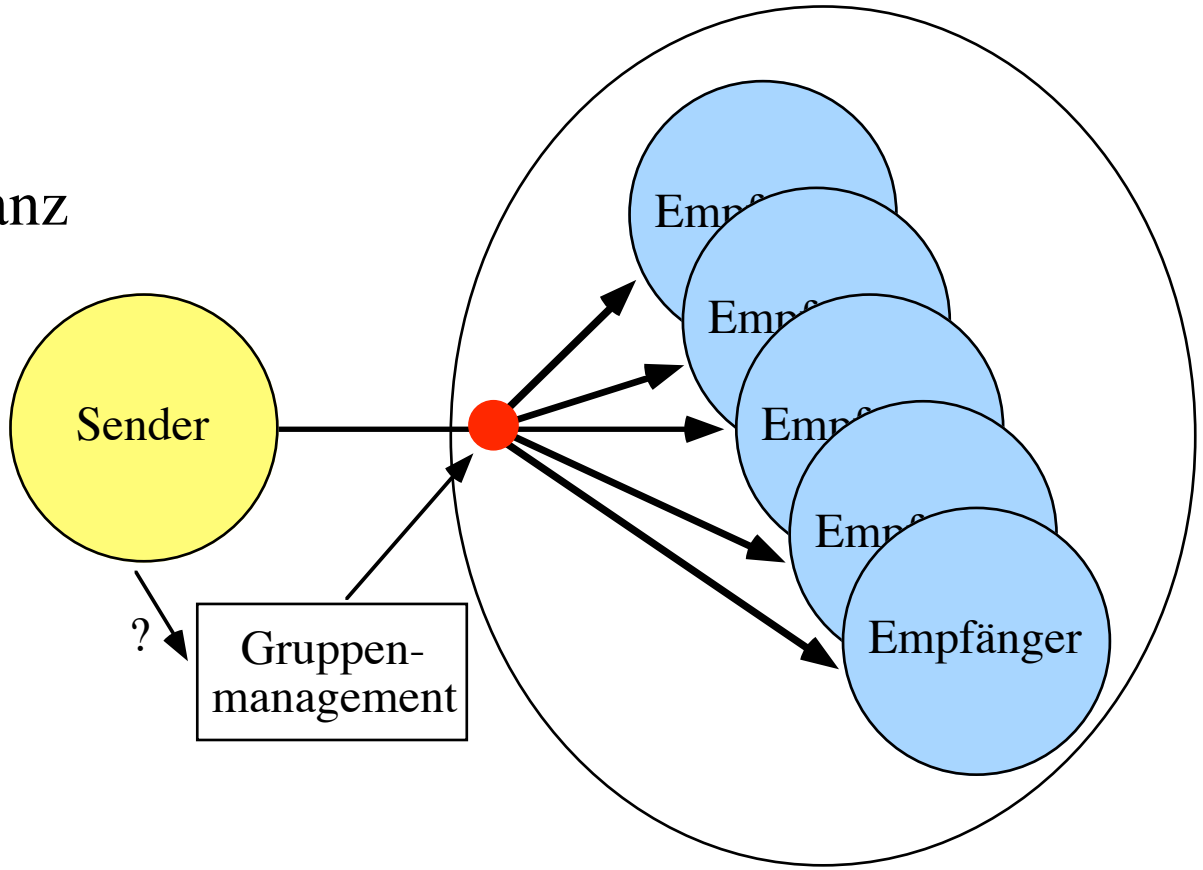


- Sonderfall objektorientiertes Programmieren
  - Selektor an entferntes Objekt schicken
  - remote method invocation: RMI
  - Serializable Java object
  - keine IDL
- Distributed Object Application
  - RMIRegistry
  - WWW-Transfer des Bytecodes
- Remote Interface
  - macht Objekte benutzbar
  - **`java.rmi.Remote`** erweitern
  - Stub wird übergeben statt Objekt

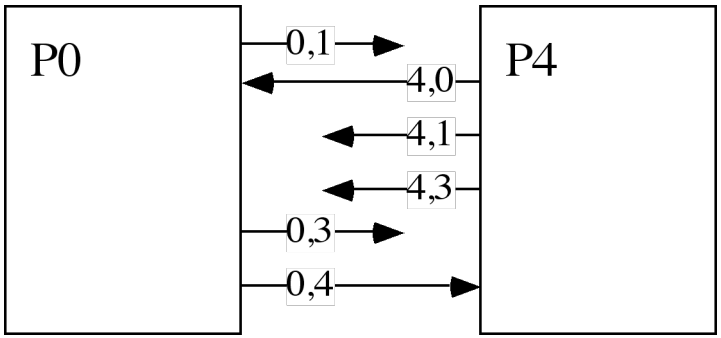


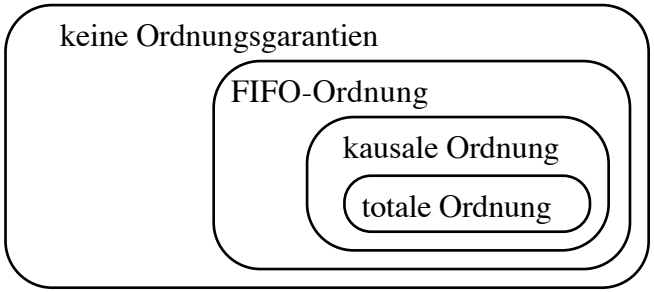
7.2.2 Gruppenkommunikation

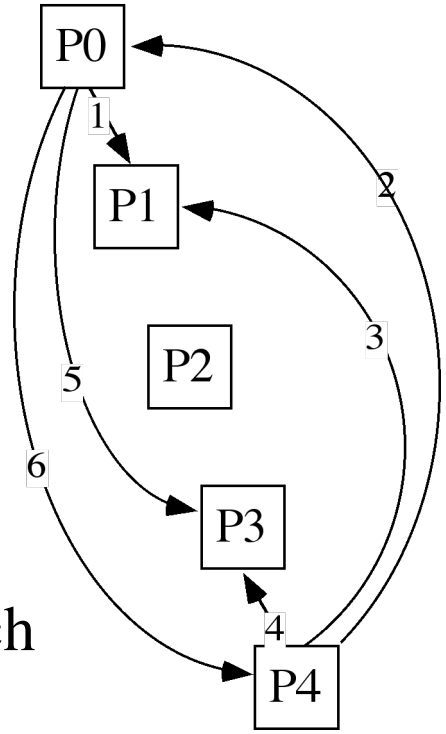
- 1:m Kommunikation
  - Replikation und Fehlertoleranz
  - Lookup (Objekte, Dienste)
  - Konferenzsysteme
  - …
- Gruppenzugehörigkeit
  - statisch oder dynamisch
  - join, leave, kick, …
  - create, discard, …
  - deterministisch?
- Zugangsregeln
  - offen oder geschlossen
  - Rechte: read, write, …
- Rollen
  - Hierarchie: Entscheidung ohne Abstimmung
  - Peers: bessere Ausfallsicherheit



- Zuverlässigkeit in Gruppe mit n Teilnehmern
  - keine Garantie
  - k-zuverlässig, k<n
  - atomar: n empfangen oder keiner
- Ankunft der Nachrichten
  - Reihenfolge des Sendens
  - Reihenfolge der Ankunft
  - eventuell Konsistenzprobleme
- Ordnung
  - ungeordnet
  - FIFO-geordnet
  - kausal geordnet
  - total geordnet
- Kausale Ordnung
  - $e_1 <_k e_2$ <=> e_1 kann Auswirkung auf e_2 haben
  - kausale Unabhängigkeit: keine Auswirkungen möglich
  - Transitivität: $e_1 <_k e_2$, $e_2 <_k e_3$ => $e_1 <_k e_3$







7.2.3 Zeit in verteilten Systemen

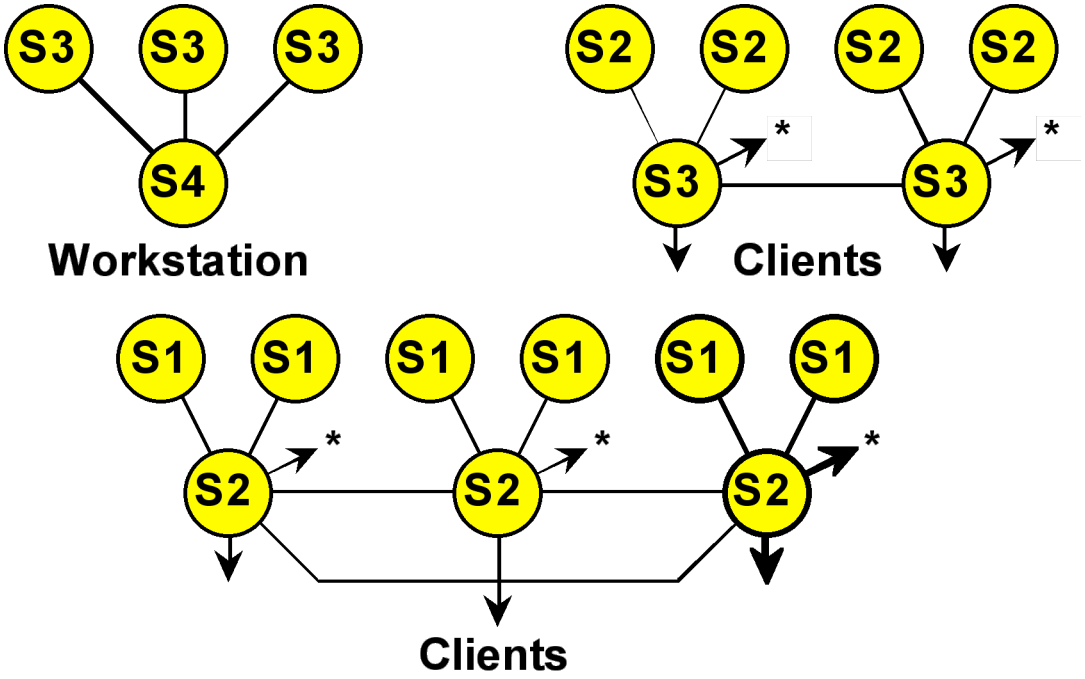
- Synchronisation
  - extern: Gleichzeitigkeit
  - intern: Intervallmessung
- Zeitstempel
  - Börse: Order-management und Zeitstempel für Transaktionen
  - RPC at-most-once Transaktionen
  - Maßnahmen gegen replay-Angriffe
  - Experimente: Messungen und Kontrolle
  - Kryptographie: key management
- Probleme
  - Laufzeit und Laufzeitunderschiede
  - clock-drift (1 Sekunde in ca. 11 Tagen)
  - Interrupts gehen verloren

- Astronomische Zeit (GMT)
  - Intervall zwischen 2 Sonnenhöchstständen
  - Sekunde = 1/86400 mittlerer 'Solartag'
- Internationale Atom-Zeit (TAI):
  - 9.192.631.770 Caesium 133 Zustandsübergänge
- WWV, DCF77
  - Auflösung 1 Sekunde, Genauigkeit << 1μsec
  - Laufzeit des Signales vom Sender zum Empfänger bekannt
  - Atmosphärische Störungen etc. => Genauigkeit ± 10 msec
- NAVSTAR GPS
  - Positionsermittlung in mobilen Einheiten
  - 22 Satelliten senden Zeit und Satelliten-Position
  - Endgerät ermittelt eigene Position aus Laufzeitunterschieden
  - Unabhängig von der Position des Endgerätes
  - Zeitfehler < 363 ns

- Uhrenanpassung
  - nachgehende Uhr wird vorgestellt
  - vorausgehende Uhr wird gebremst
  - zurückstellen erzeugt Probleme mit Zeitstempeln
- Zentraler Zeitgeber mit Richt-Zeit
  - Zeit-Nachricht hat Verzögerung T_{trans}
  - T_{trans} ist variabel
  - Laufzeitmessung
- Algorithmus von Christian [1989]
  - Zeitanfrage vom Klienten zum Server
  - round-trip delay messen
  - $t_d = (t_e - t_s)/2$
  - $t_{UTC} = t_{xmit} + t_d$
  - Antwortzeit des Servers berücksichtigen
  - t_d mit statistischen Methoden bewerten
  - Laufzeiten symmetrisch?

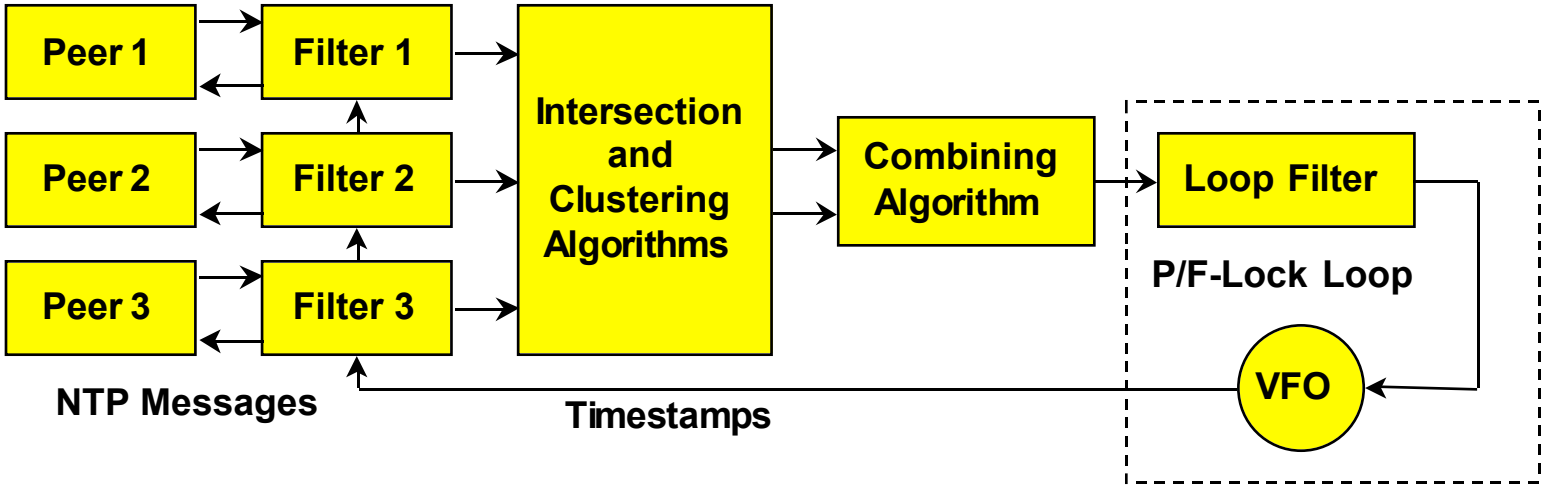
- Berkeley-Zeit-Algorithmus
  - System ohne 'gute' Zeit => Abstimmung über die Zeit
  - zentraler time-deamon fragt alle Geräte nach der Zeit
  - round-trip delay messen und berücksichtigen
  - Zeit wird gemittelt
  - jeder Klient erhält Korrekturinformation
  - Fehlertoleranzmaßnahmen
- Network Time Protocol NTP [Mills, 1992]
  - RFC 1059, 1119 und 1305
  - UTC-Synchronisation
  - 1.000.000+ Prozesse im Internet
  - UDP
  - skalierbar => hierarchisches Serversystem
  - Authentifizierung und Fehlertoleranz
  - Millisekunden im WAN
  - < 1 msec im LAN
  - < 1 μsec mit GPS-support

- NTP: Hierarchisches System
  - Primärserver mit UTC-Emfängern
  - Sekundärserver werden synchronisiert
  - auf mehreren Ebenen (stratum)
  - Klienten auf der untersten Ebene



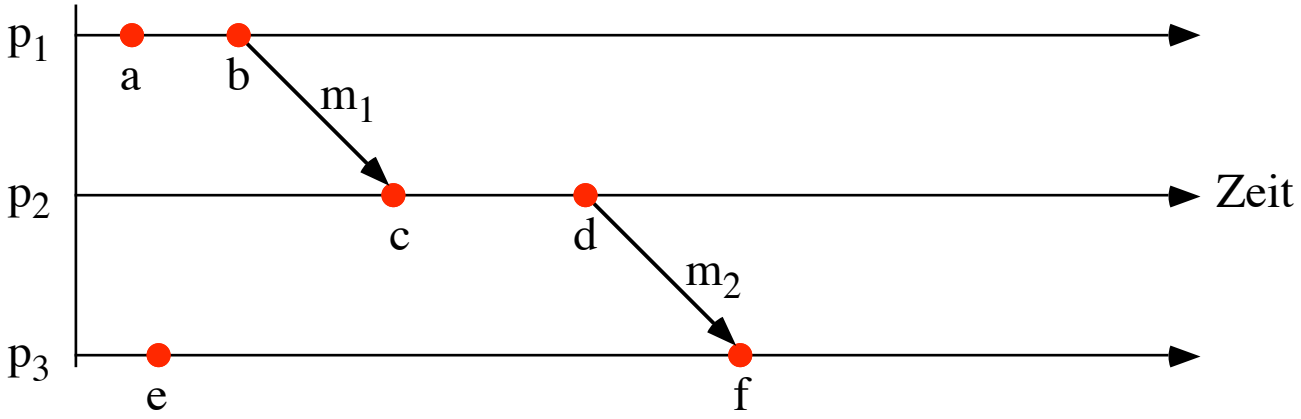
- NTP Kommunikationsmodi
  - multicast mode im LAN
  - symmetric mode synchronisiert Server-Uhren
  - procedure call mode ähnlich Christian's protocol

- Verfahren im Teilnehmer



- Mehrere Quellen für Zeit
  - Uhr-Filter wählt aus bis zu 8 Offsets aus
  - Clustering Algorithmen wählen Server-Untermenge
  - Genauigkeit und Fehlertoleranz
  - Gewichteter Durchschnitt der Offsets
- Phase/frequency-lock feedback loop kontrolliert lokale Uhr
  - Genauigkeit
  - Stabilität

- Logische Zeit
  - physikalische Zeit nicht immer nötig
  - relative Ordnung von Ereignissen
- Lamport-Zeit
  - happened-before Relation
  - im Prozess: a vor b: a→b
  - in verschiedenen Prozessen: send(m)→rcv(m)
  - transitiv
  - alle 'concurrent'



- Lamport Algorithmus
  - jeder Prozess hat eigene Uhr = Zähler
  - lokale Ereignisse:

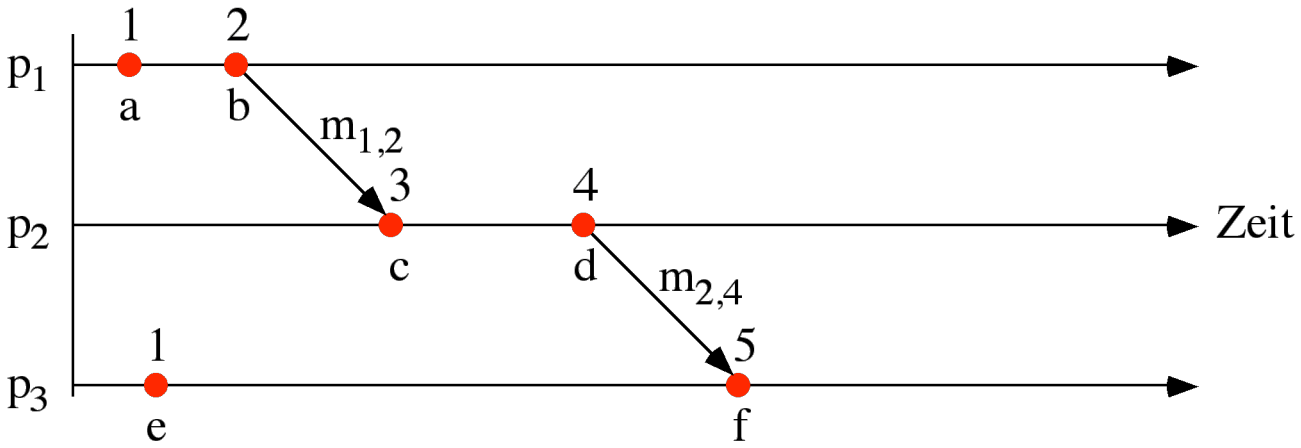
```
c_local++;
```

  - Sendeereignis:

```
c_local++; send(message,c_local);
```

  - Empfangsereignis:

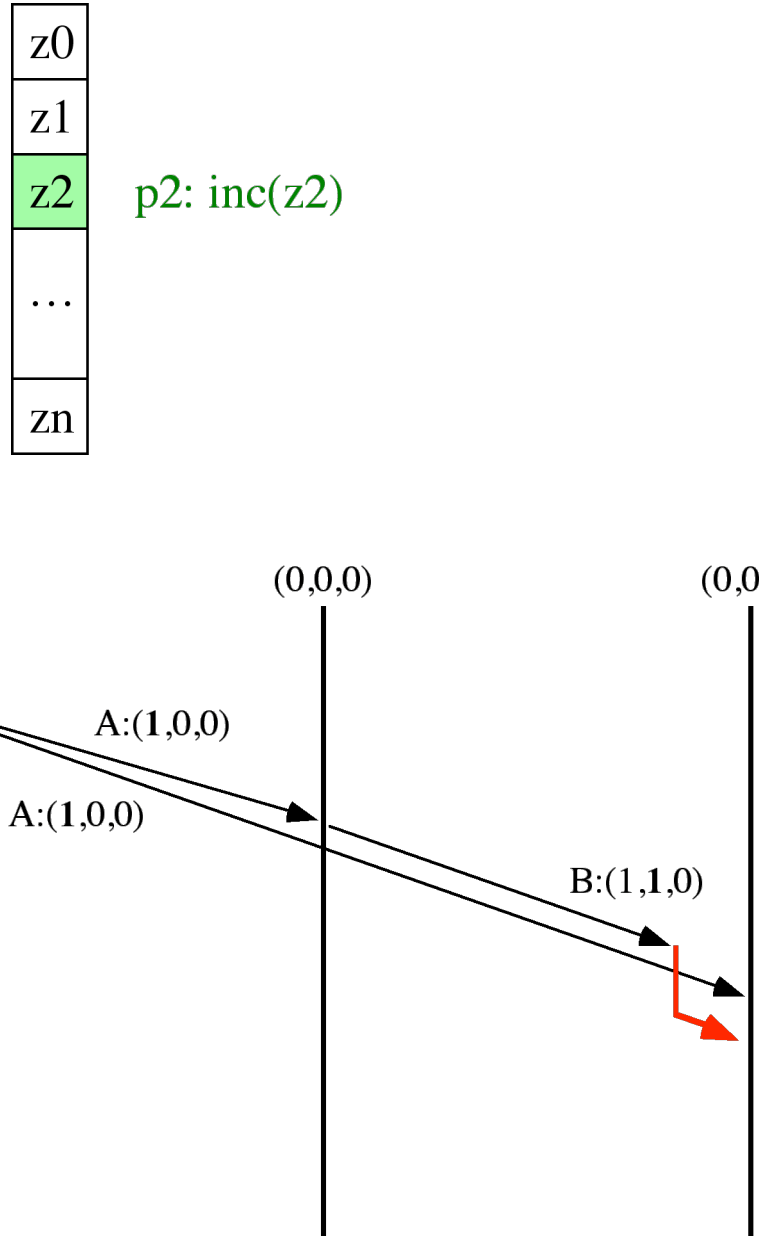
```
receive(message,c_remote);
c_local = max(c_local,c_remote)+1;
```



- C(a)<C(b): a→b oder allb
  - keine totale Ordnung

- Zeitstempel
  - jeder Prozess p_i hat *logische* Zeit z_i
  - bei Nachrichtenversand inkrementieren
- Vektorzeit
  - Zeitstempel-Vektoren
  - jede Station hält lokalen Vektor
  - Nachrichten enthalten Vektoren

  ->Vektor der Quelle beim Senden
  - Empfang: Vergleich mit lokalem Vektor
  - Empfänger verzögert evtl. Nachrichten



- Implementierung der Vektorzeit
  - Sender k packt Zeitstempel-Vektor in Nachrichten
  - Empfänger vergleicht Stempel-Vektor V mit lokalem Vektor L
  - accept if $(V_k = L_k + 1)$ and $(V_i \leq L_i \; \forall \; i \neq k)$

p0	p1	p2	p3	p4	p5
4	3	3	3	2	3
6	7	5	7	6	7
8	8	8	8	8	8
2	2	2	3	2	3
1	1	1	1	1	1
5	5	5	5	5	5

p0	p1	p2	p3	p4	p5
4	4	3	4	2	4
6	7	5	7	6	7
8	8	8	8	8	8
2	2	2	3	2	3
1	1	1	1	1	1
5	5	5	5	5	5

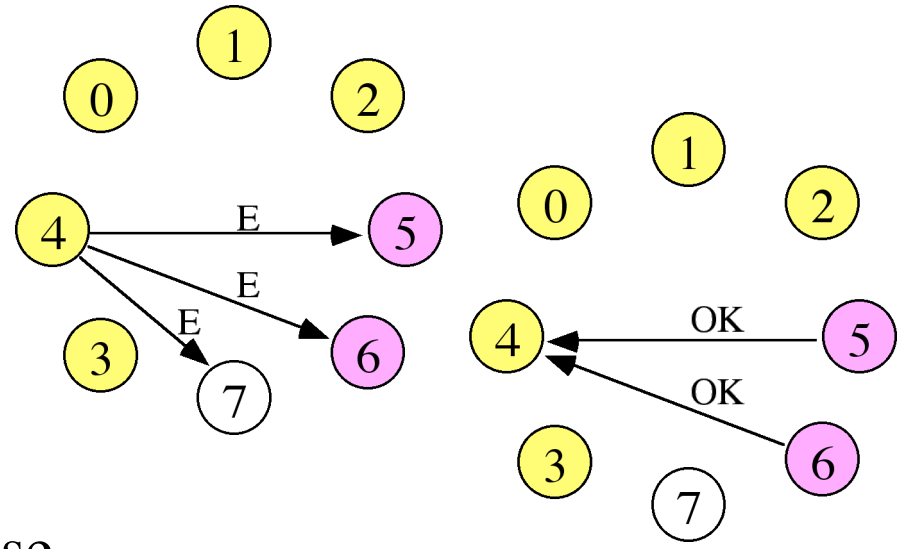
Nachricht von p0 an alle

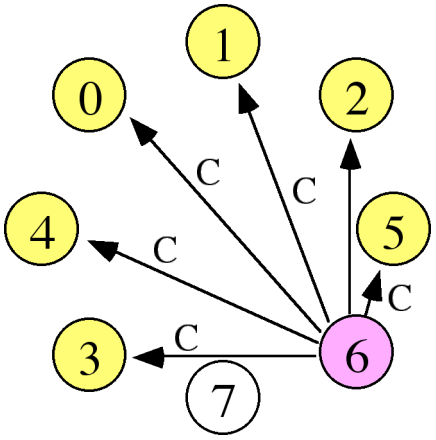
4	6	8	2	1	5	Daten

p0	p1	p2	p3	p4	p5
4	3	3	3	2	3
6	7	5	7	6	7
8	8	8	8	8	8
2	2	2	3	2	3
1	1	1	1	1	1
5	5	5	5	5	5
	accept	delay	accept	delay	accept

7.2.4 Wahlverfahren

- Zentrale Leitung verteilt 'wählen'
  - alle Prozesse haben Nummer
  - alle Prozesse sind einander bekannt
  - kein Wissen über Aktivität
- Bully-Algorithmus (bully = Tyrann)
  - Station P merkt, daß Koordinator weg
  - P sendet Election(P) an 'höhere' Prozesse
  - Anwort 'OK' => Abbruch
  - keine Antwort => P neuer Koordinator
  - Coordinator(P) an alle
- Q empfängt Election(P)
  - tolerieren oder
  - OK an P + Election(Q) an alle höheren
- Falls ein Koordinator X wieder aufwacht:
  - Coordinator(X) an alle
- Ringalgorithmus auch möglich



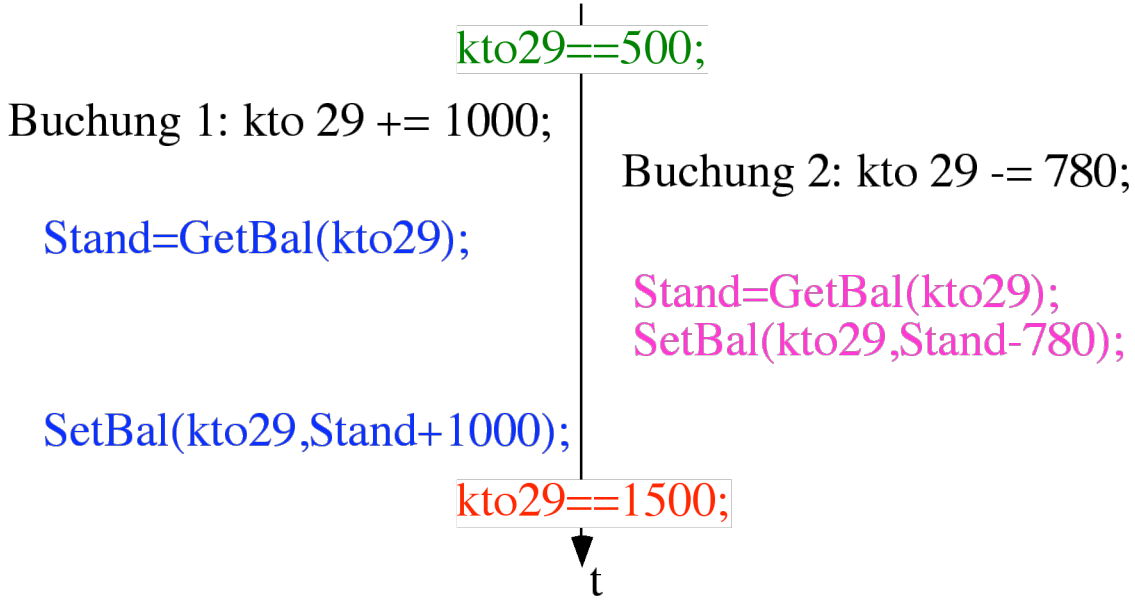


7.2.5 Transaktionen

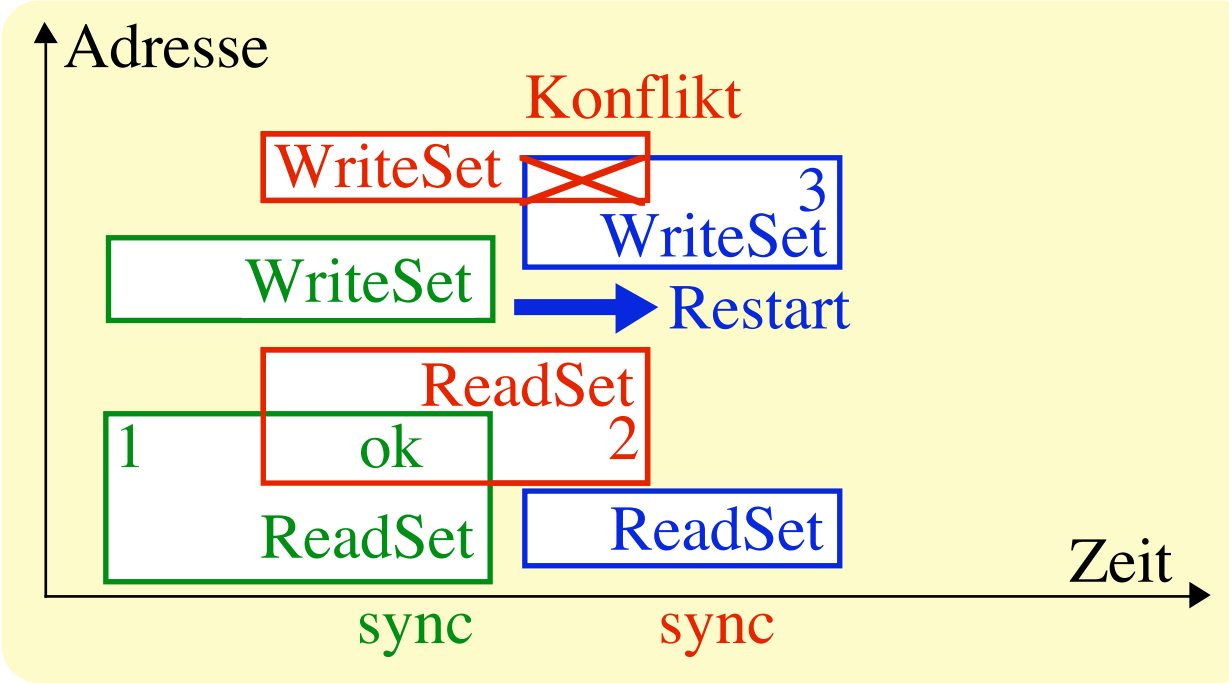
- Verteilte Dateisysteme, Datenbankoperationen
  - Lesen + Schreiben = Update
  - Überweisung = Abheben(kontoX) + Einzahlen(kontoY)

```
void Einzahlen(konto_nummer,betrag) {
    kontostand = getbalance(konto_nummer);
    putbalance(konto_nummer, kontostand + betrag; }
```

- Probleme bei Parallelität



- Atomare Operationen
  - Transaktion
  - ACID
- Atomicity
  - entweder alle Teile oder keiner ausgeführt
  - commit oder abort
- Consistency
  - Zustandsüberführung konsistent -> konsistent
- Isolation
  - Serialisierbarkeit
  - erlaubt Nebenläufigkeit
- Durability
  - Persitenz des Resultates => Speicherung

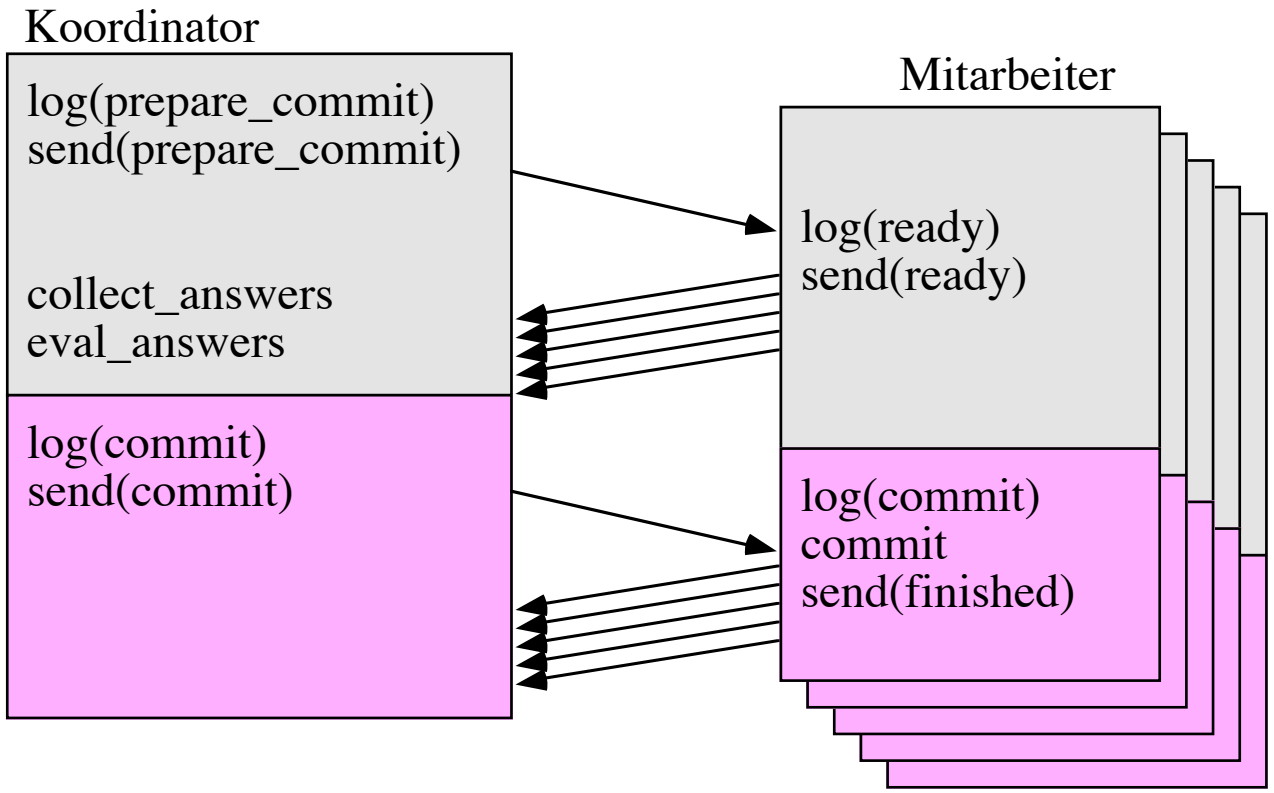



```
- tid beginTransaction
- result endTransaction(tid)
- val get(tid,attribut)
- set(tid,attribut,val)
- abort(tid)
```

- Fehlerverhalten des Servers
  - recovery nach dem Restart des Servers
  - roll-back nach Klienten-Absturz
  - time-outs bis zum Abschluß einer Transaktionen
- Klienten-Verhalten
  - Operation läuft in time-out
  - Operation kommt mit Fehler zurück nach Server-restart
  - Rückfrage beim Benutzer?
- Private Arbeitskopie
  - Operationen auf shadow copy
  - commit: atomares Zurückschreiben
  - Vergleich Orginal mit Anfangs-Shadow?

- Intention-list
  - Server zeichnet Operationen einer Transaktion auf
- Optimistisch
  - Undo-Liste
  - Lesen einfach
  - Commit: Undo-Liste löschen
  - Abort: Rückabwickeln (rollback)
  - Lesen für andere Prozesse schwer
- Pessimistisch:
  - Do-Liste sammeln
  - Lesen komplex: Orginal und Do-Liste mergen
  - Commit: atomar Ausführen
  - Abort: Do-Liste löschen
  - Lesen für andere Prozesse einfach

- 2-Phasen Commit Protocol
  - Koordinator sendet 'Prepare' vor dem Commit
  - Mitarbeiter antworten Ready, wenn Commit OK
  - Stimmen zählen
  - Commit durchführen

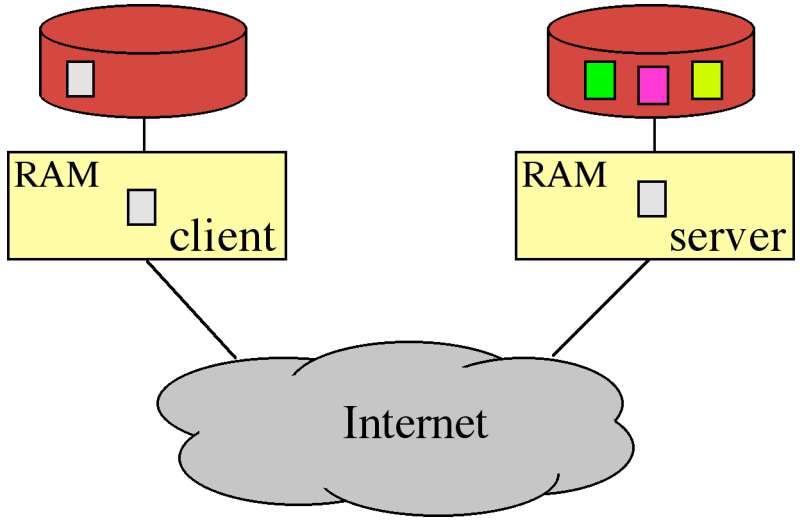


7.2.5 Effizienz beim verteilten Zugriff

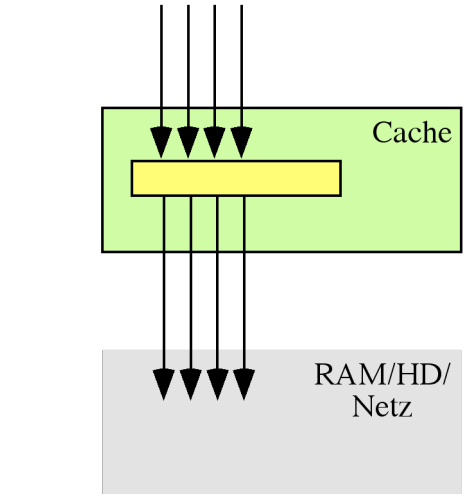
- Daten gemeinsam nutzen
  - Koordination: Semaphore, Locking
  - Konsistenz: Datenbanken, Transaktionen
  - Effizienz: Caching, verteilte Kopien
- Entfernter Zugriff kostet Zeit
  - Transport im Netzwerk
  - TCP-Verbindung, …
  - Lesen und besonders Schreiben
  - zeichenweiser Zugriff auf Strings …
- Caching
  - lokale Kopie -> schneller Zugriff
  - dynamisch, heuristisch
  - Ersetzungsalgorithmen
- Replikation: Verfügbarkeit und Fehlertoleranz
- Problembereiche: Konsistenz und Transparenz

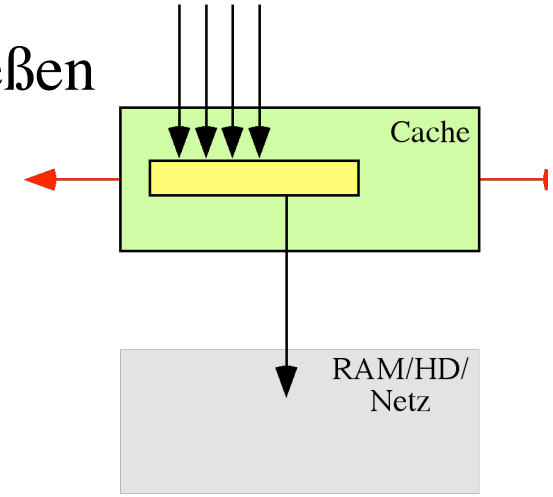
7.2.5.1 Caching

- Objekte 'in der Nähe' halten
  - Speicherzellen, Seiten, Sektoren, Tracks, Files, …
  - Cache-RAM, RAM, Controller, …
  - Zugriffskosten reduzieren
- Sonderfall verteiltes Dateisystem
  - Server: RAM
  - Klient: Festplatte, RAM
  - Prozess, OS, Caching-Prozess
- Aufgaben
  - Einheit des Cachings: Blöcke oder Files?
  - Cache kleiner als Speicher: was soll 'gecacht' werden?
  - Prefetch?
- Least recently used
  - Element das am längsten nicht mehr benutzt wurde ersetzen
  - File-Cache-Zugriff selten vgl. CPU-Cache
  - verkettete Liste als Verwaltungsstruktur



- Write-through
  - Schreiben im Cache
  - Schreiben auf Orginal
  - Validierung späterer Zugriffe: Zeitstempel,Prüfsummen
- Delayed write
  - Netzwerk-Nachricht bei jedem Schreiben aufwendig
  - Sammeln der Updates und Burst
  - unklare Semantik
- Write on close
  - Session-Semantik
  - Öffnen - lokale Updates - komplett-Update beim Schließen
  - Locking bzw. Transaktionssemantik
- Write-invalidate: andere caches konsistent halten
- Zentrale Kontrolle
  - alle Zugriffe anmelden: read, write, update
  - Remove-from-Cache bei konkurrierenden Zugriff
  - unsolicited-message

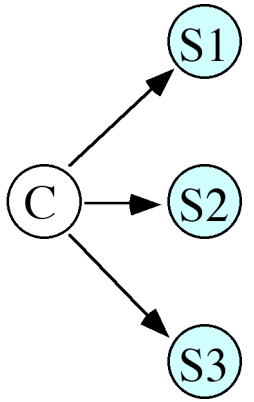


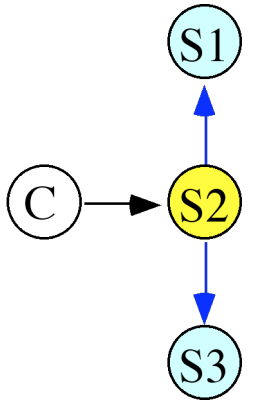


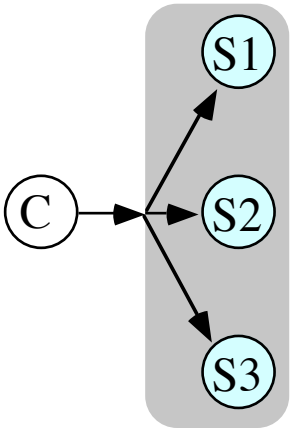
7.2.5.2 Replikation

- Client/Server, Peer-to-peer
  - Nachrichtenaustausch
  - Lesen Regelfall, Schreiben Ausnahme
- Verteilter gemeinsamer Speicher (DSM)
- Replikationsmanagement
  - Konsistenzmodell
  - Verhältnis Lesen-Schreiben
- Asynchrones Replikationsmanagement
  - Lesen und Schreiben lokal
  - Update anderer Kopien periodisch
  - inkohärent bis zur Synchronisation
  - entkoppelt und einfach
- Synchrones Replikationsmanagement
  - alle Replikate atomar geändert
  - volle Konsistenz
  - hoher Aufwand

- Erzeugung von Replikaten
  - nicht alles muß repliziert werden
  - Auswirkungen auf Konsistenzerhaltung
- Explizite Replikation
  - klientengesteuert: Anforderung an mehrere Server
  - Zugriff auf ein Replikat
  - Konsistenzmanagement beim Klienten
- Lazy Replikation
  - Objekterzeugung durch Server
  - Server legt Replikate bei anderen Servern an
  - Konsistenzmanagement durch Server
- Servergruppe
  - Gruppenkommunikation
  - Objektanforderung bewirkt n Replikate
  - Schreibzugriff an Gruppe
  - Konsistenz abhängig von Gruppensemantik







- Primärkopien
  - 'Orginal' im Primärserver
  - Replikate auf Sekundärserver
  - Lesen überall, Schreiben nur auf Orginal
  - Update der Replikate in der Servergruppe
  - Primärserver-Ersatz siehe Election
- Abstimmung
  - N Server
  - N/2+1 Replikate beschreiben mit Zeitstempel
  - N/2+1 Replikate lesen
  - Verallgemeinerung: Quota
  - $N_l+N_s > N$
  - N_l und N_s entsprechend Verhältnis Lesen/Schreiben optimieren
- Lazy Update
  - Server sammeln Updates
  - gossip-Nachrichten
  - schwache Konsistenz
- Peermodell als Variante