Verteilte Systeme
und
Kommunikationsdienste

Prof. Dr. Konrad Froitzheim

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

Inhaltsübersicht

Literatur

- Coulouris: Distributed Systems - Concepts and Design, 1994
- Flueckiger, F.: Understanding Networked Multimedia, 1995.
- Halsall, F.: Data Communications, Computer Networks and Open Systems, 1995.
- Schmidt, D., Huston, S.: C++ Network Programming; 2002.
- Stallings, W.: Networking Standards, 1993.
- A. Tanenbaum: Distributed Operating Systems, 1995
- M. Weber: Verteilte Systeme, Skript Universität Ulm, 1996

Formales

Termine:

Vorlesung:	Mittwoch	18:00 - 19:30
Übung:	tbd	

Dramatis Personae:

Konrad Froitzheim
Professur Betriebsysteme und Kommunikationstechnologien
mailto:frz@tu-freiberg.de
Nico Pranke
mailto:pranke@informatik.tu-freiberg.de

Unterlagen zur Vorlesung:

http://ara.informatik.tu-freiberg.de/vorlesungen/DistSys2009.Doc

1. Taxonomie der Kommunikationsdienste

- Raum und Zeit
  - gleich- versetzt (synchron - asynchron)

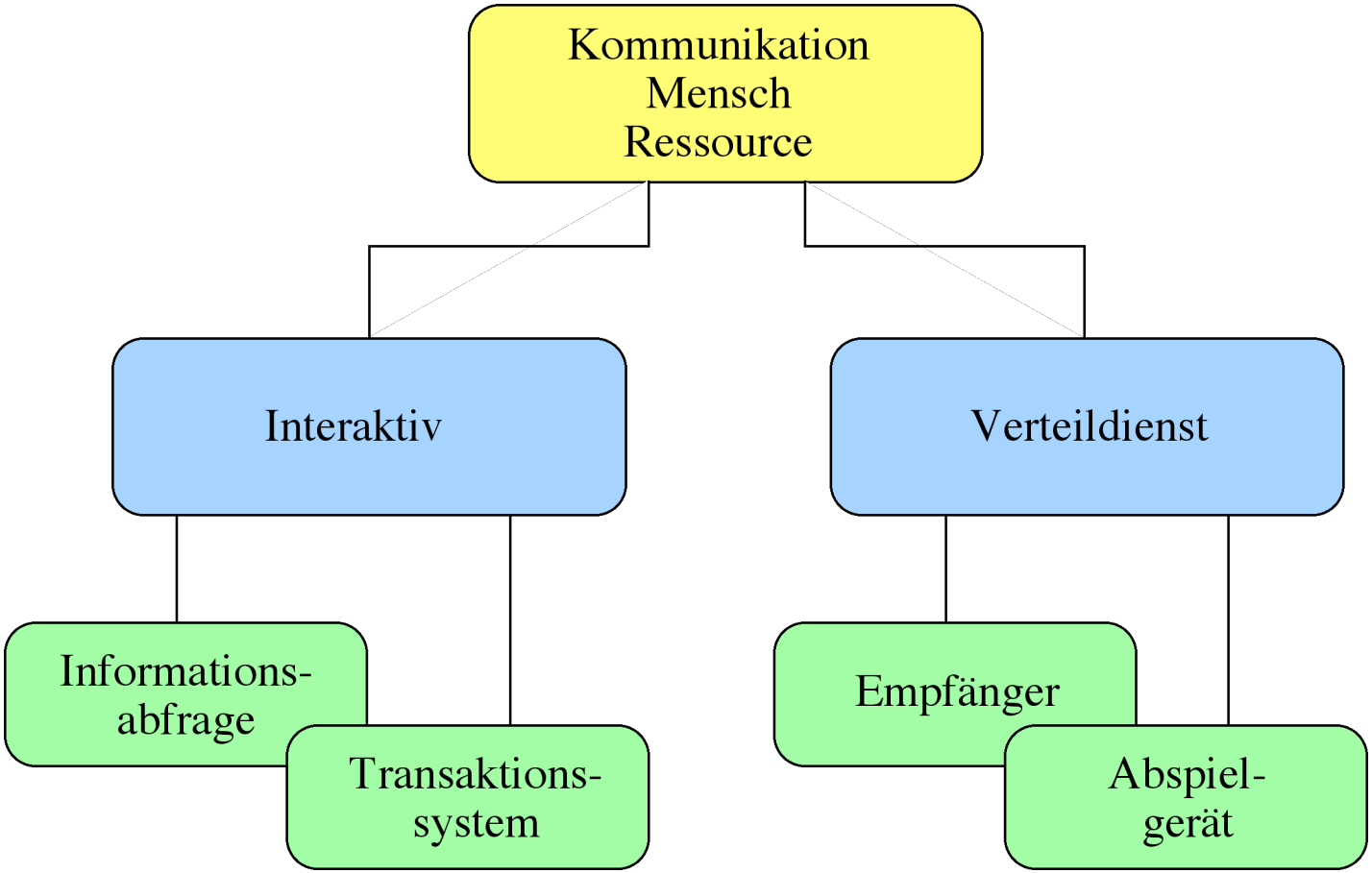
Ort \ Zeit	gleich (synchron)	versetzt (asynchron)
lokal	Gespräch Konferenz	Schwarzes Brett Informationssystem Video-on-demand
verteilt	Telekonferenz Telearbeit	Brief, Paket verteilte Dokumente verteiltes Informationssystem

- Topologie und Zeit

Assoziation / Zeit	1:1	Definierte Gruppe	Beliebige Gruppe
synchron	Zwiegespräch	Besprechung Konferenzen	Radio Fernsehen
asynchron	Datenbankabfrage Nachrichten-austausch	Rundschreiben	Massenversand Schwarzes Brett

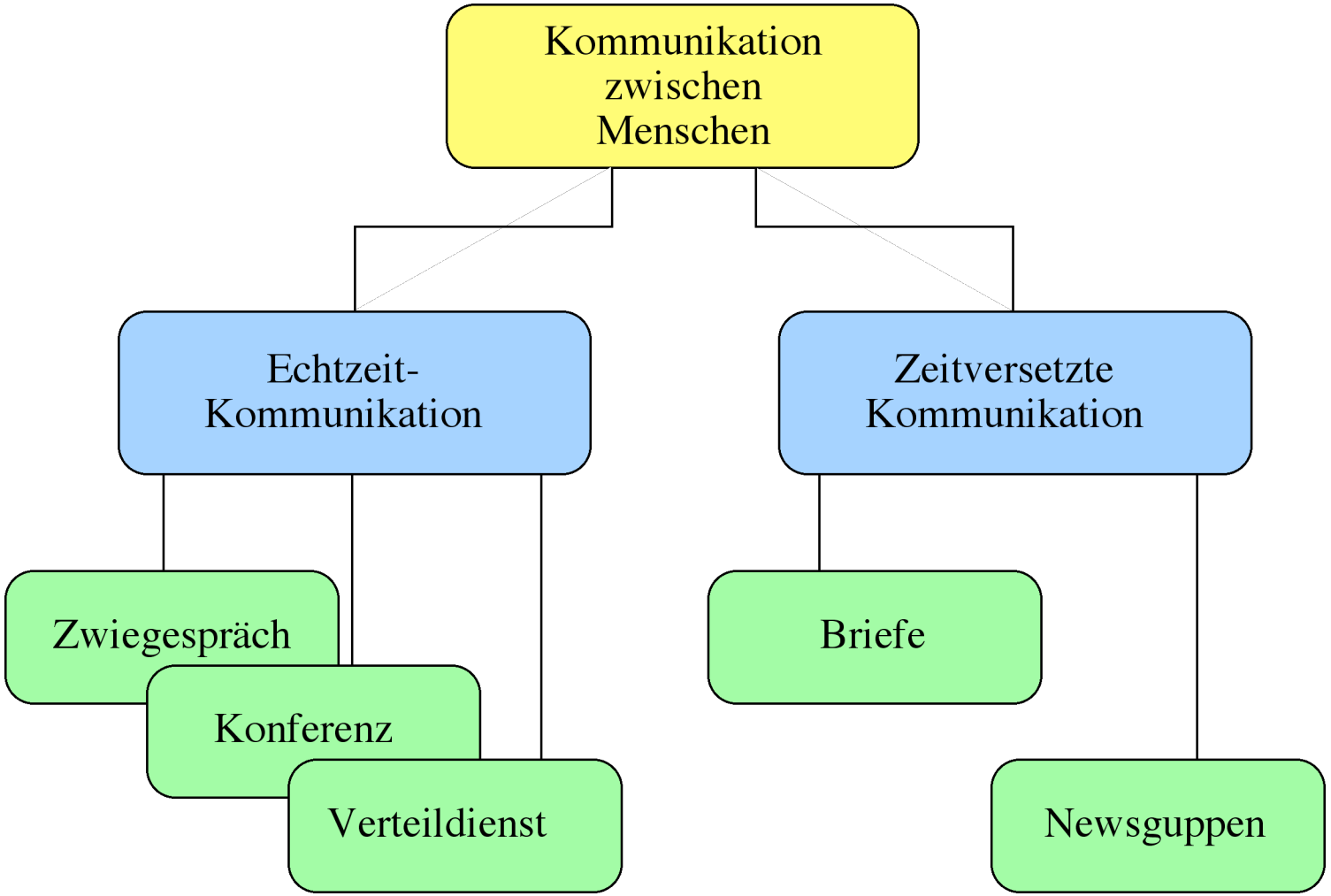
- Teilnehmerorientiert
  - Kommunikation zwischen Menschen
  - Kommunikation zwischen Mensch und Maschine

- Kommunikation zwischen Mensch und Maschine
  - nach F. Fluckiger



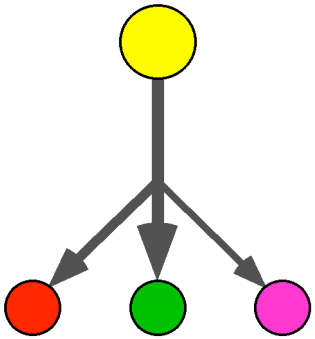
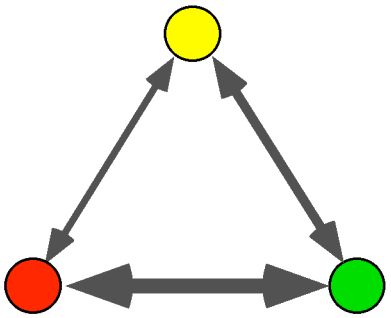
  - lokal: Film, Jukebox, Multimedia-Kiosk
  - verteilt: TV, WWW

- Kommunikation zwischen Menschen
  - automatisch verteiltes System

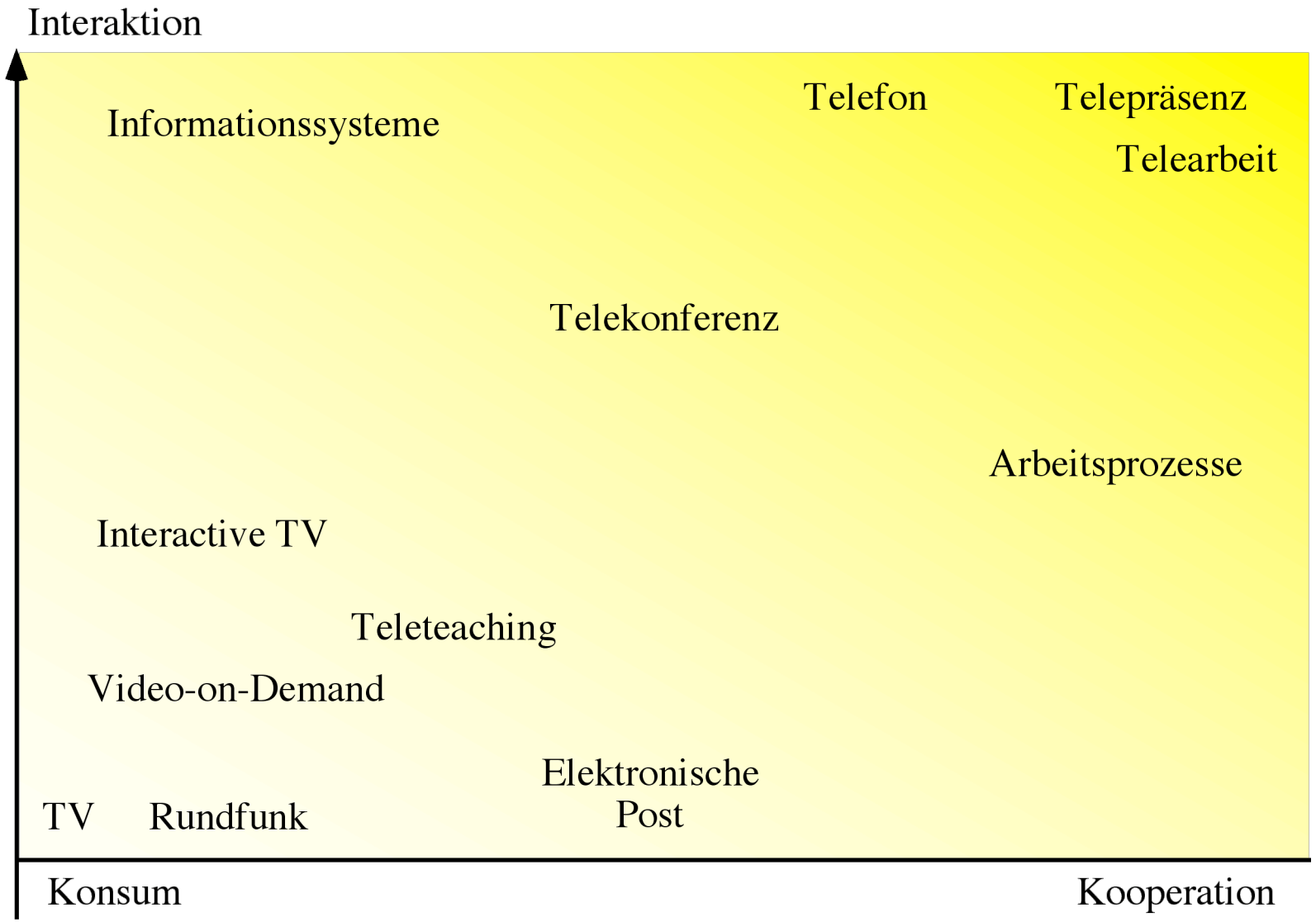


- ITU klassifiziert Kommunikationsdienste
- Interactive Services (ITU-T, CCITT)
  - Conversational Services
    Echtzeitdienste
    end-to-end Informationsaustausch im Dialog
  - Messaging Services
    Benutzer-zu-Benutzer-Kommunikation
    Zwischenspeicherung im Netz
    Beispiel: elektronische Post
  - Retrieval Services
    Datenbankabfragen im Dialogbetrieb
    Beispiel: Bildschirmtext
- Distribution Services (ITU-R, CCIR)
  - Distribution Without Individual Presentation Control
    Verteildienste ohne Interaktionsmöglichkeit
    Beispiele: Rundfunk und Fernsehen
  - Distribution With Individual Presentation Control
    Benutzer beeinflußt die zeitliche Abfolge der verteilten Information
    Beispiel: Video-on Demand

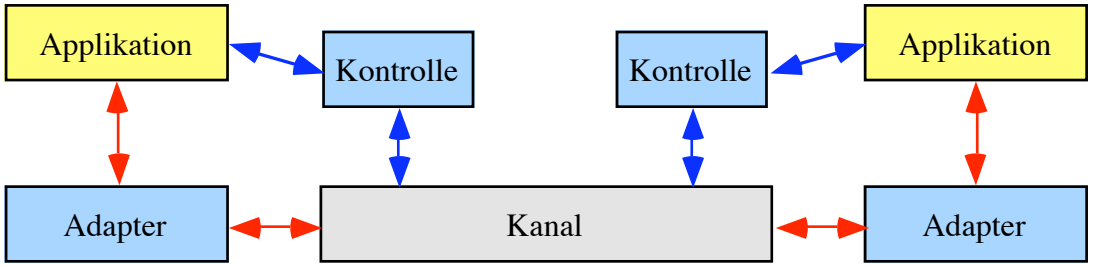
- Rollenbasierte Taxonomie der Kommunikationsdienste

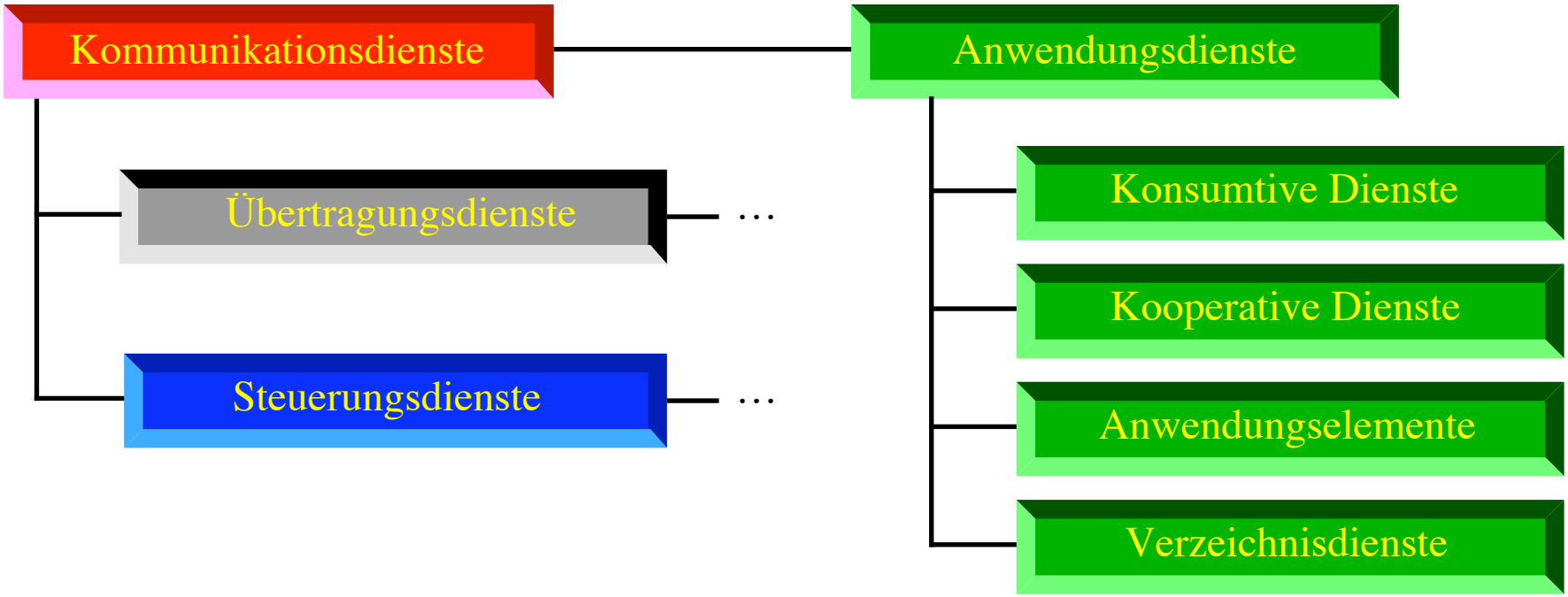
	Konsumtive Dienst	Kooperative Dienste
Dienste	Post Radio, TV Interaktives TV Informationssysteme World Wide Web	Telefon Telepräsenz Telearbeit Unterricht Arbeitsprozesse
Rollen	Programmanbieter Konsumenten	Teilnehmer
		

• Kommunikationsdienste und Interaktivität



- Komponentenbasierte Taxonomie [Froitzheim, 1997]

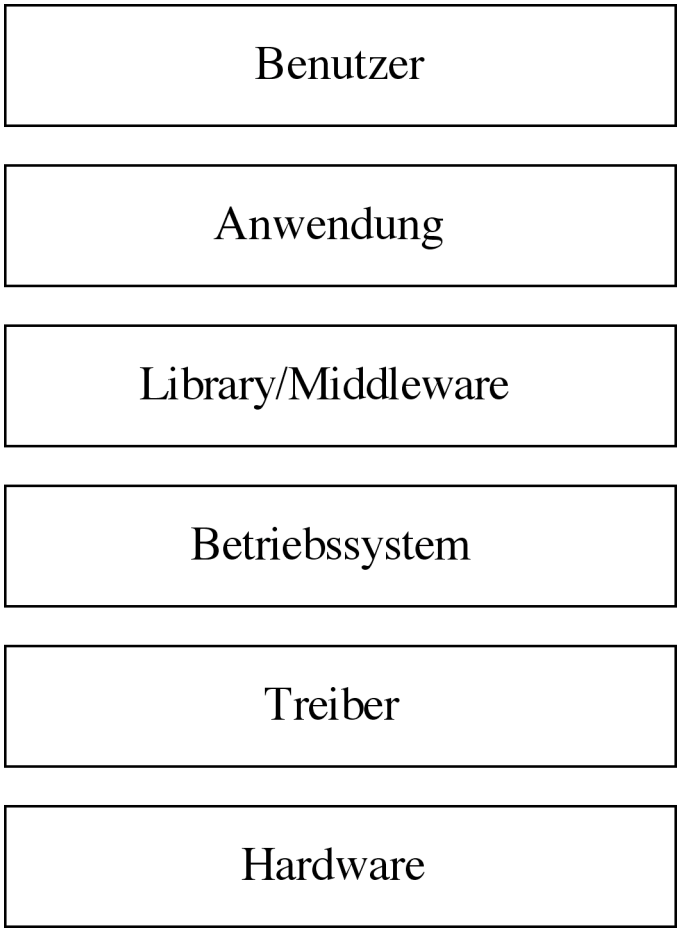




2. Anwendungselemente

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

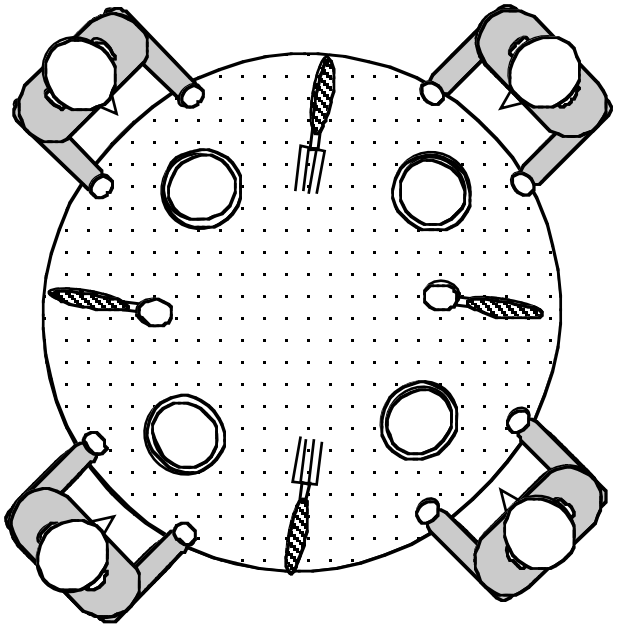
- Entwurf Verteilter Systeme
  - Komunikationsmodelle
  - Gruppenkommunikation
  - Integrationsniveau
  - Algorithmen
- Programmierung Verteilter Systeme
  - Synchronisation (Zeit?)
  - Koordination (Semaphore, Mutex)
  - Auswahlverfahren
  - Zuverlässigkeit
  - Programmiermodell, ACE
- Ein besonderes System: DSM
- Wichtige Beispiele
  - Verteilte Dateisysteme
  - Verteiltes Betriebssystem: Mach



- 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;`

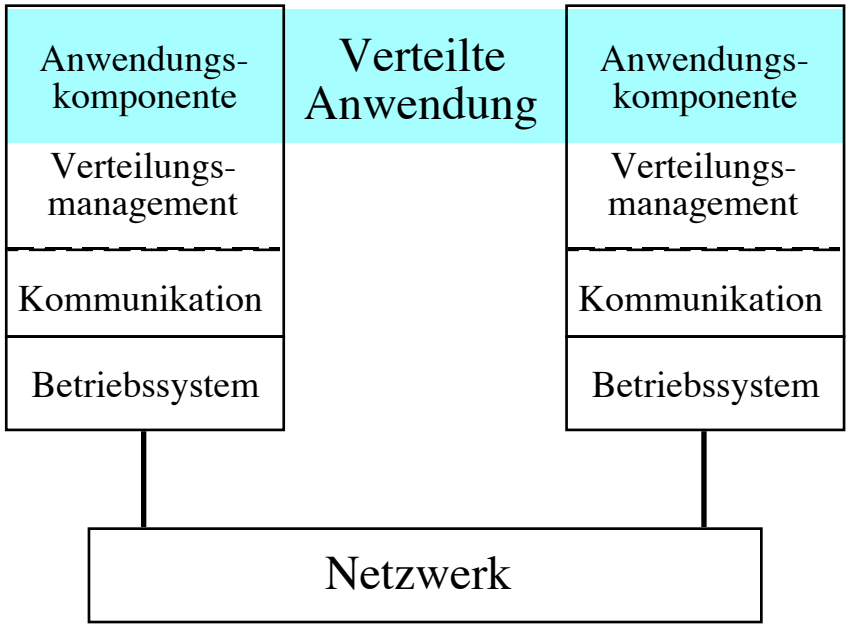
- Nebenläufige Ausführung ohne Synchronisierung
  - verschiedene Resultate möglich für c
  - 120 mögliche Permutationen (5*4*3*2)
  - zum Beispiel für drei Prozessoren sei c = { 0, 10, 20, 30 ... }

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

2.0 Infrastruktur Verteilter Systeme

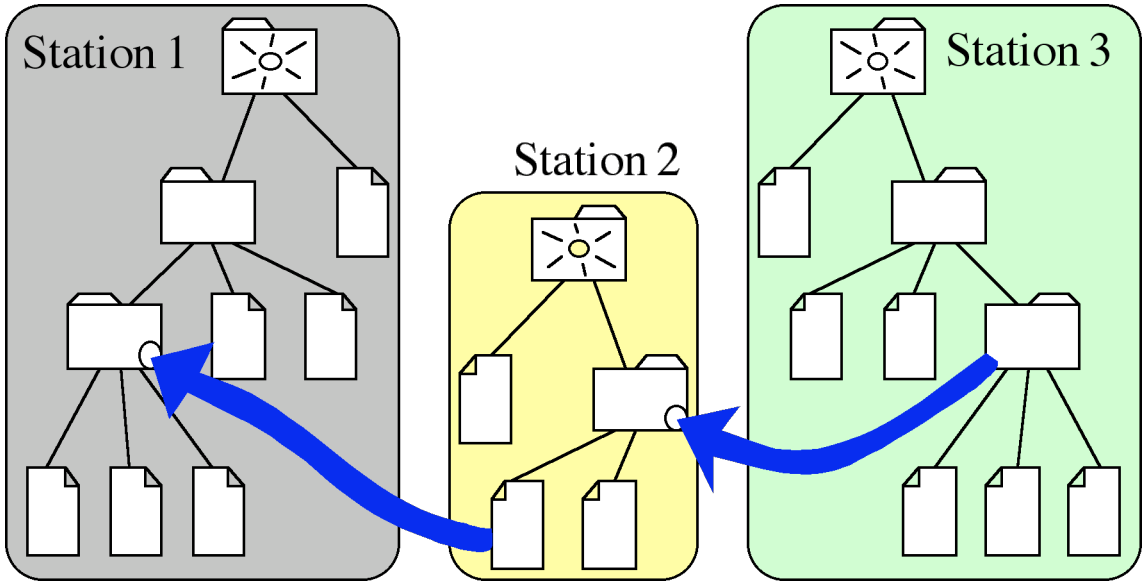
2.0.1 Do it Yourself

- 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



2.0.2 Verteilte Dateisysteme

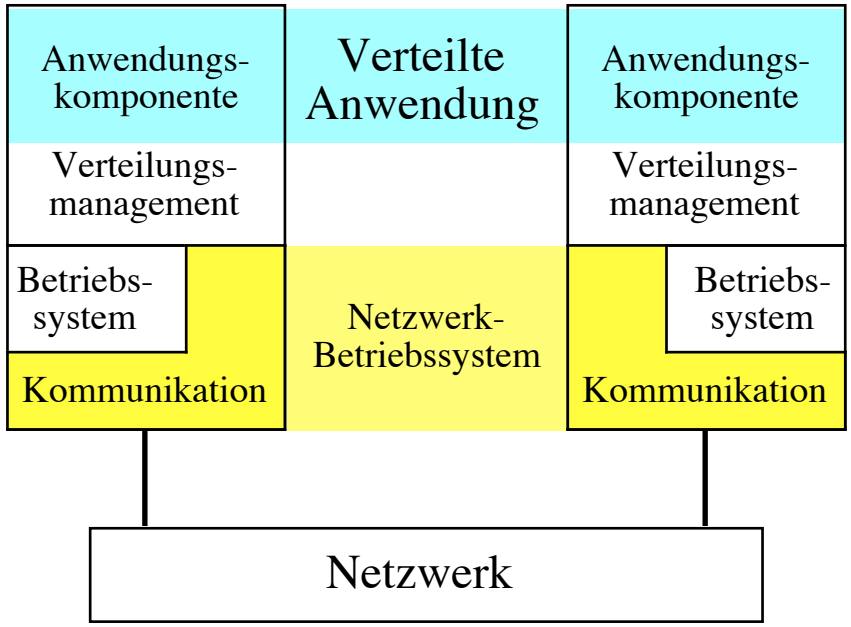
- Remote Disk
- Remote Files
- Zugriff auf entfernte Volumes
- Zugriff auf nicht-lokale Dateien
- Allgemein montierbare Verzeichnisbäume



- Replikation von Dateien und Verzeichnissen

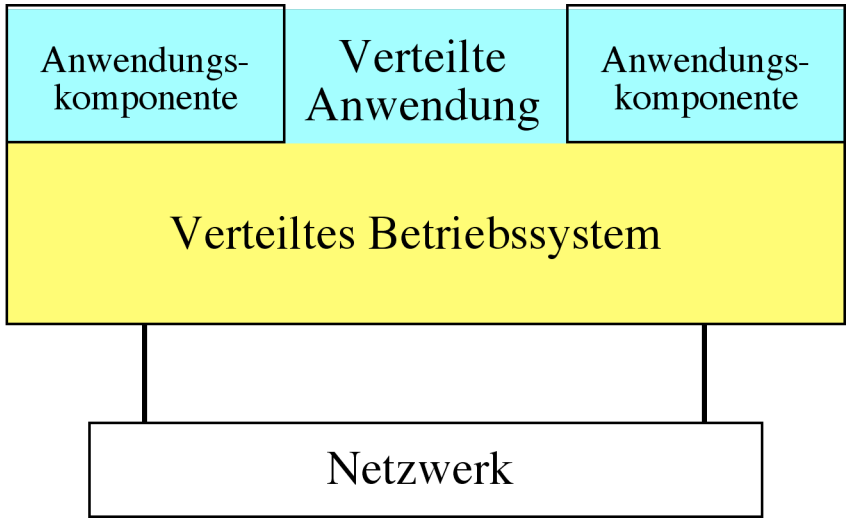
2.0.3 Netzwerkbetriebssysteme

- Übertragen von Daten zwischen Stationen
- 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



2.0.4. Verteilte Betriebssysteme

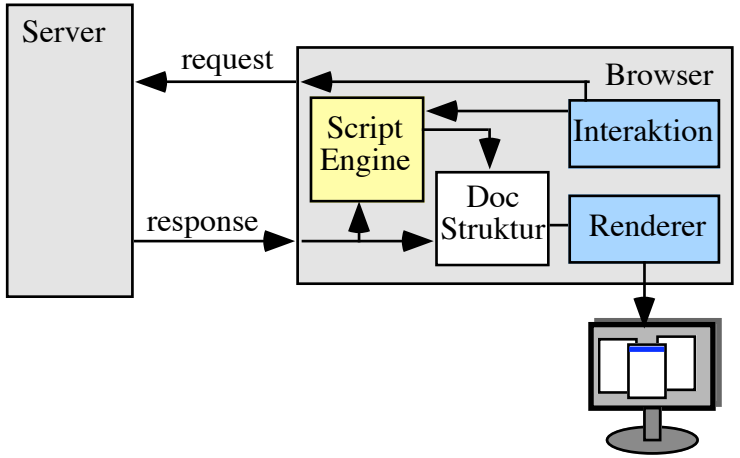
- Explizite Kommunikation
  - Datagramm-Nachrichten
  - Sockets und Streams
  - Entfernte Prozeduraufrufe
  - Verteilte Objekte & Methoden (RMI, DCOM)
  - Corba, Java Spaces (Jini) ...
- Implizite Kommunikation
  - transparente Prozeduraufrufe
  - transparente Variablenzugriffe
  - identisches API -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

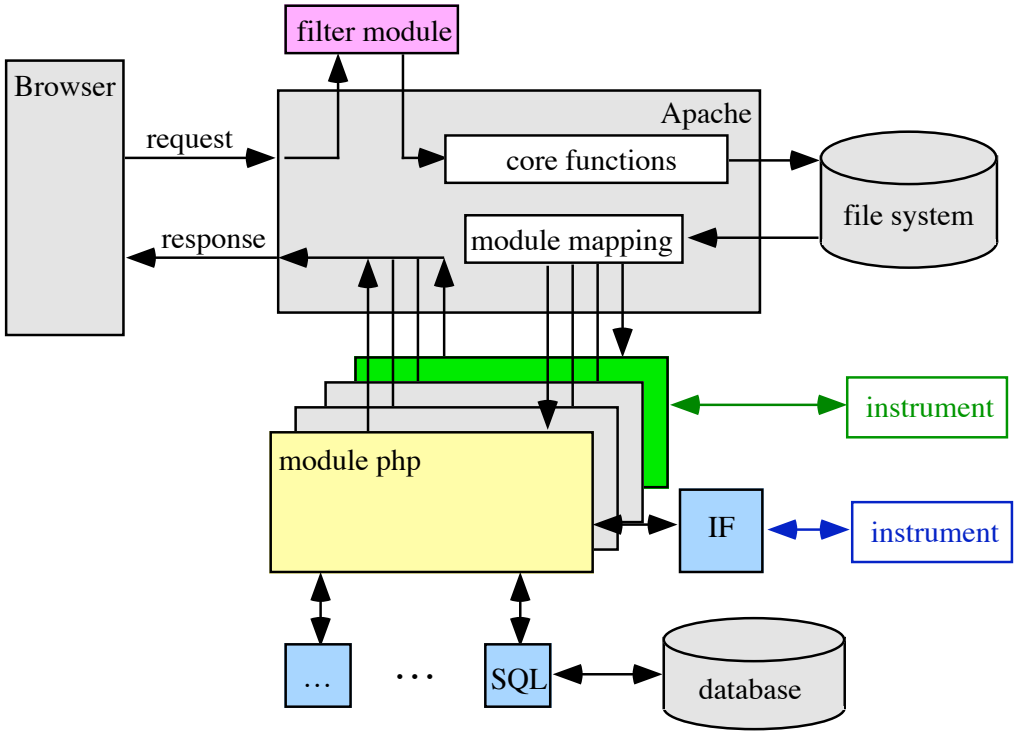


- Gemeinsamer verteilter Speicher DSM
  - explizite oder implizite Kommunikation
  - verteilte gemeinsame Objekte
  - verteilte gemeinsame Variablen
  - verteilte gemeinsame Adressräume ...
- Zielsetzung
  - skalierbares System
  - einfache Programmierung
  - lokale und verteilte Programme
  - sichere und effiziente Ausführung
- Probleme
  - Komplexes Betriebssystem
  - Semantikverlust: wenig Anpassung an das verteilte Problem
  - besondere Schnittstellen zur Dienstegütewahl

2.0.5 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



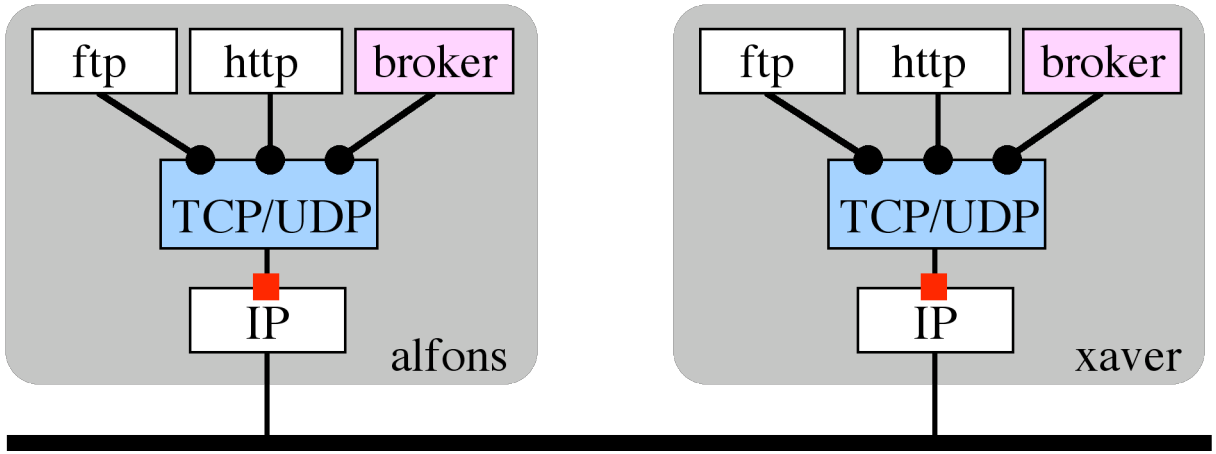


2.0.6 Anforderungen und Probleme

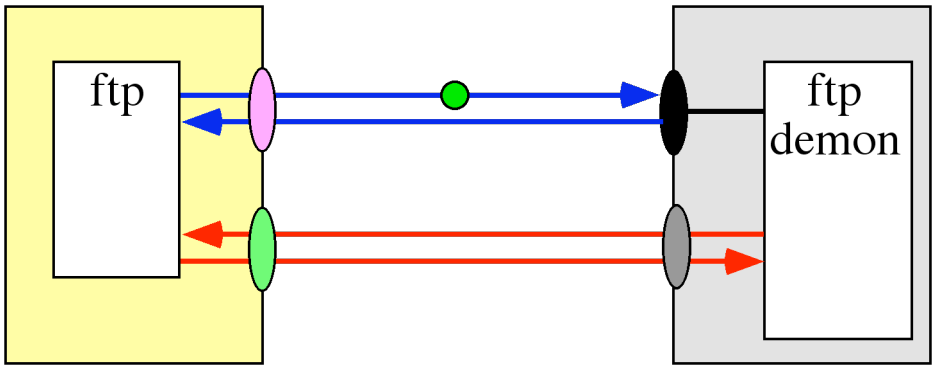
- Zugriffstransparenz
  - Einheitliches API
  - Single-System image
  - Logische Namen (nicht physische)
  - Yellow-Pages
- Ortstransparenz
  - Dateien und Verzeichnisse
  - Terminalzugang, Arbeitsumgebung
  - Rechenleistung
  - Hauptspeicher
  - Prozesse
- Dienstetransparenz
  - Drucker und Server
  - 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
- Nebenläufigkeit

2.1 Kommunikationsmodelle

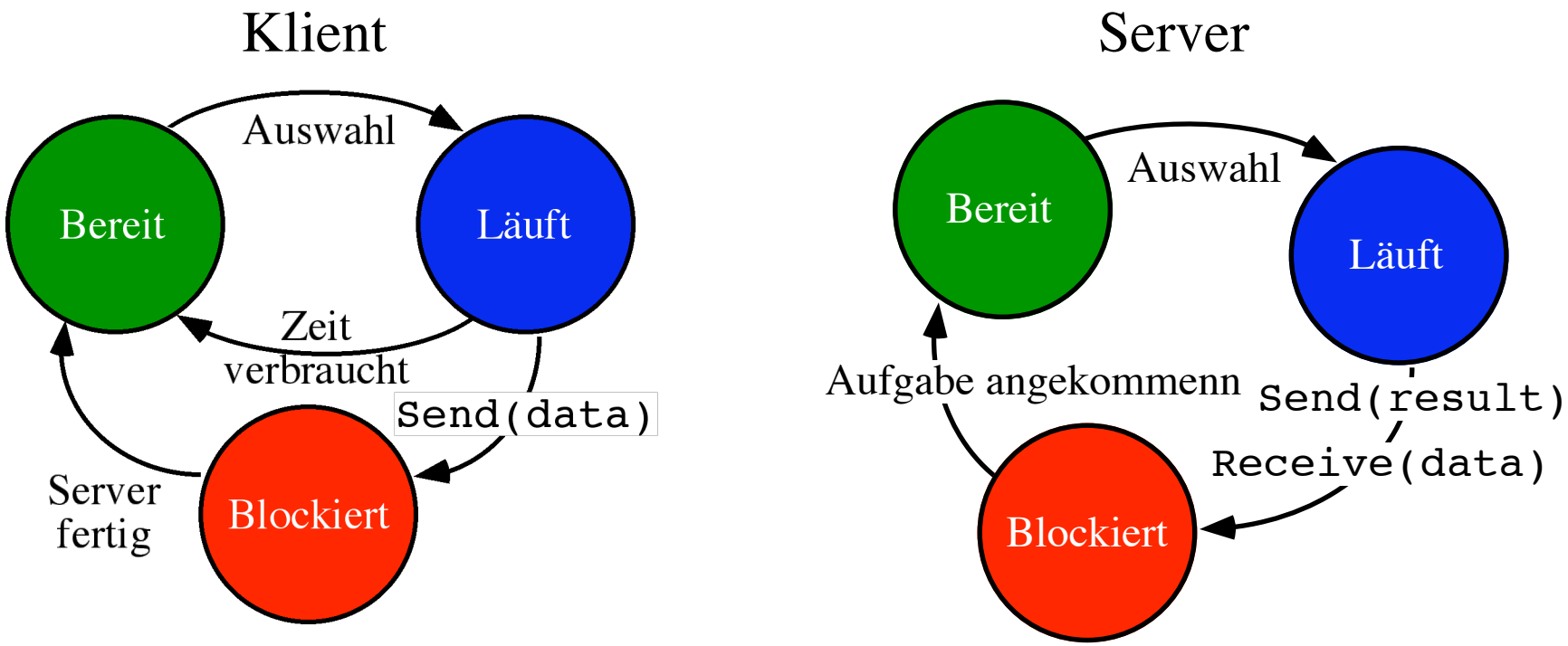
- Adressierung
  - -> Vorlesung Rechnernetze: Adressen
  - -> Vorlesung Rechnernetze: Ports



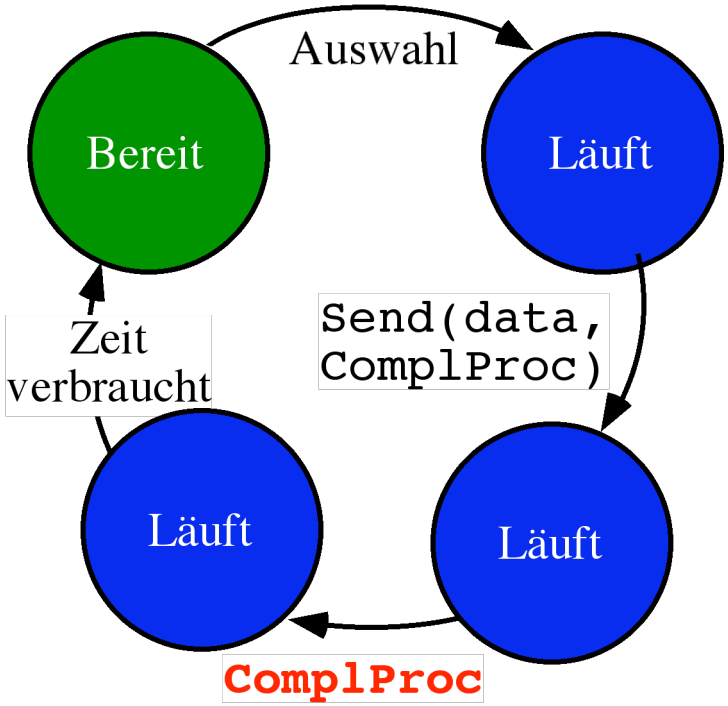
- Port-Auskunftssysteme: Broker
  - Teil von Corba, Jini, …



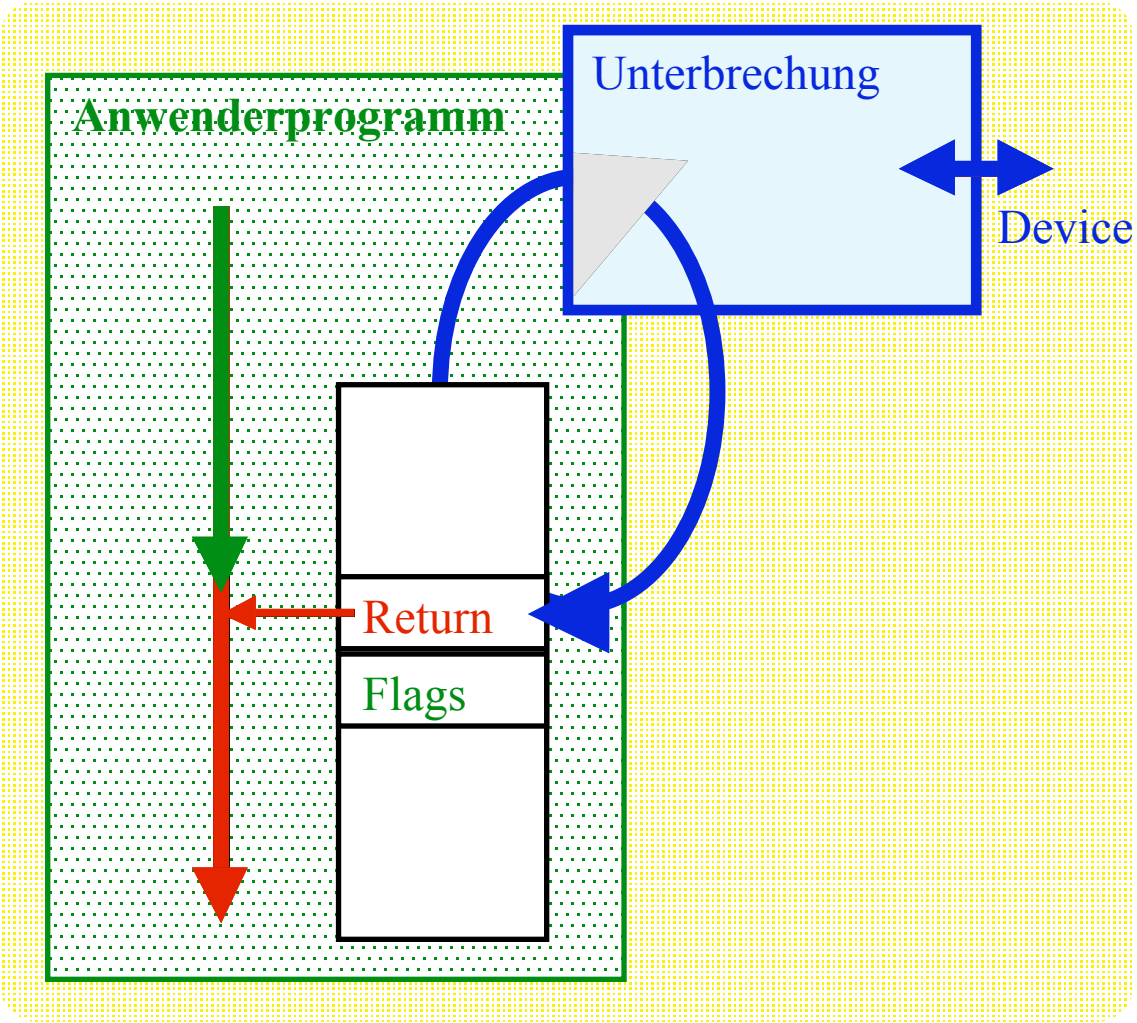
- Blockierung
  - Semantik der Diensteanforderung
- Serverprozess wartet mit `receive(data)`
- Synchrone Kommunikation
  - Prozess ruft Dienst mit `send(data)`
  - Prozess wartet bis Aufgabe erledigt
  - Prozess läuft weiter



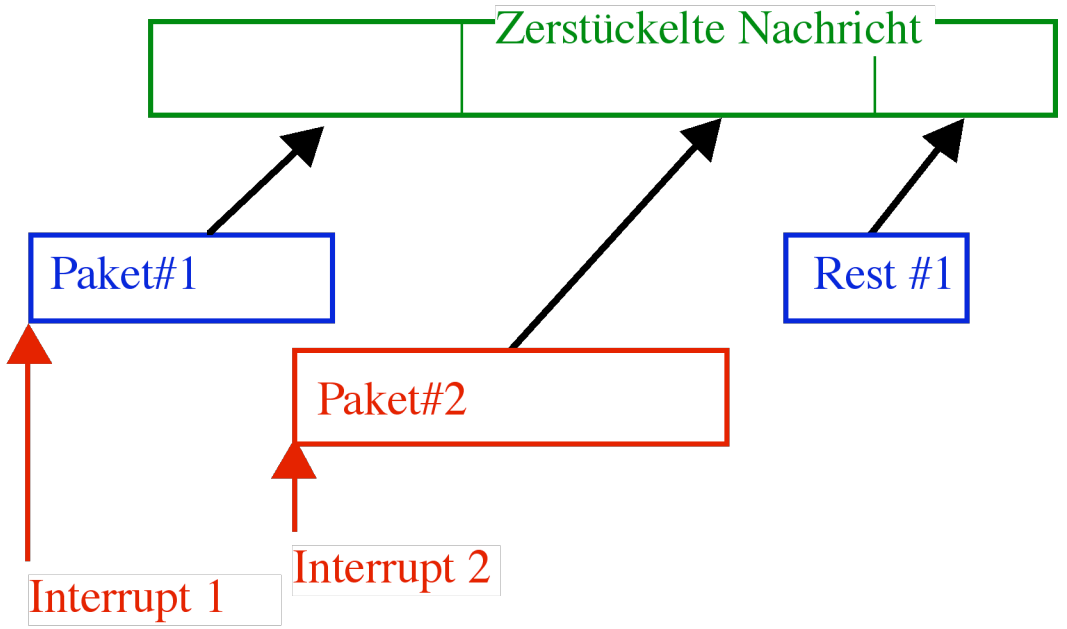
- Asynchrone Kommunikation
  - Prozess setzt Empfänger (Interrupt Server etc) auf
  - Prozess ruft Dienst mit `send(data,ComplProc)`
  - Prozess wartet bis `send` zurückkehrt
  - Prozess arbeitet weiter
  - Completion Prozedur wird gerufen im Interrupt

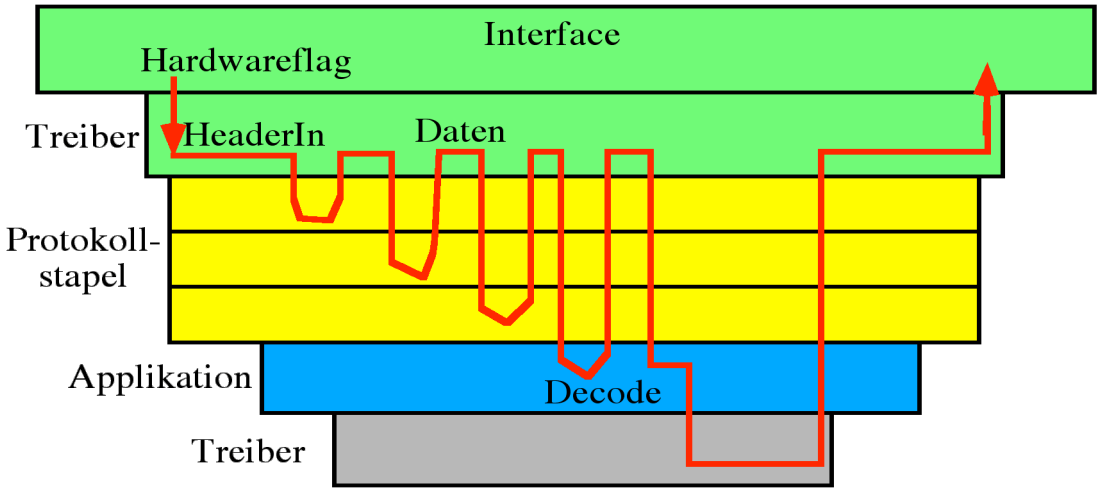


- Interrupt
- Programm unterbrochen
  - Aufgabe mit höherer Priorität
  - Zustand des unterbrochenen Programmes aufzeichnen
  - Zustand nach Abschluß der Unterbrechung restaurieren
  - Fortsetzung an Unterbrechungsstelle
- Behandlung von Fehlersituationen
  - Speicherschutzverletzung
  - Ungültige Maschineninstruktion
  - 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



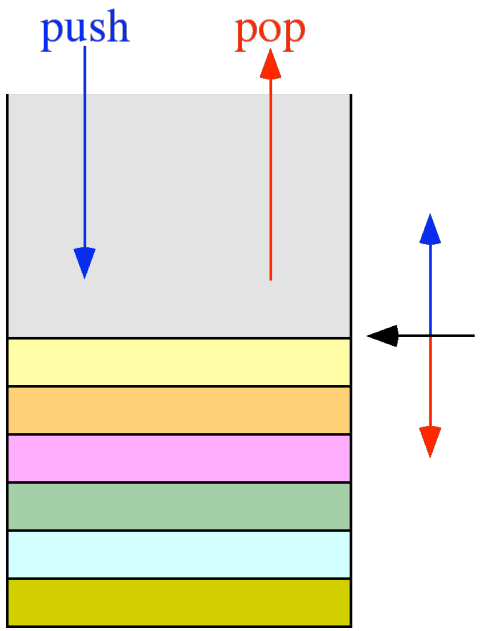


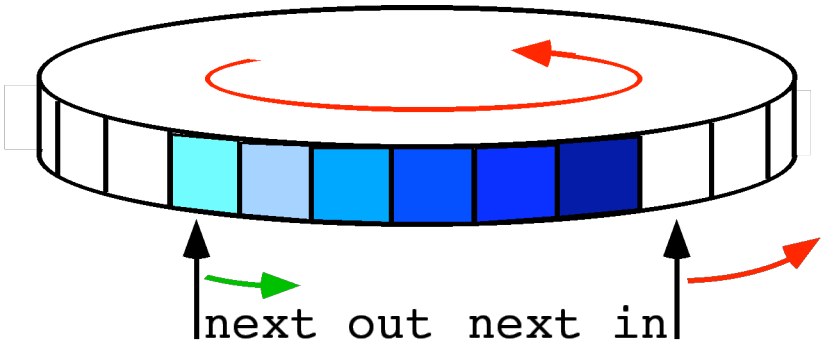
[1]

- 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

- Puffer
  - Platz für überraschend eintreffende Daten
  - Prozesse kommunizieren über Speicherplatz
  - Puffer können überlaufen
- Bearbeitungssemantik
  - Stack: last in - first out
  - Warteschlange: first in - first out
- Pufferverwaltung
  - freie Puffer finden
  - gebrauchte markieren (lock)
  - nicht mehr gebrauchte freigeben
  - Fehlerroutinen aufrufen
- Ringpuffer
  - 2 Zeiger: next_in, next_out
  - Schreiben inkrementiert next_in
  - Lesen inkrementiert next_out
  - inkrementieren modulo buffersize
  - am besten 8, 16, 32, 64, …

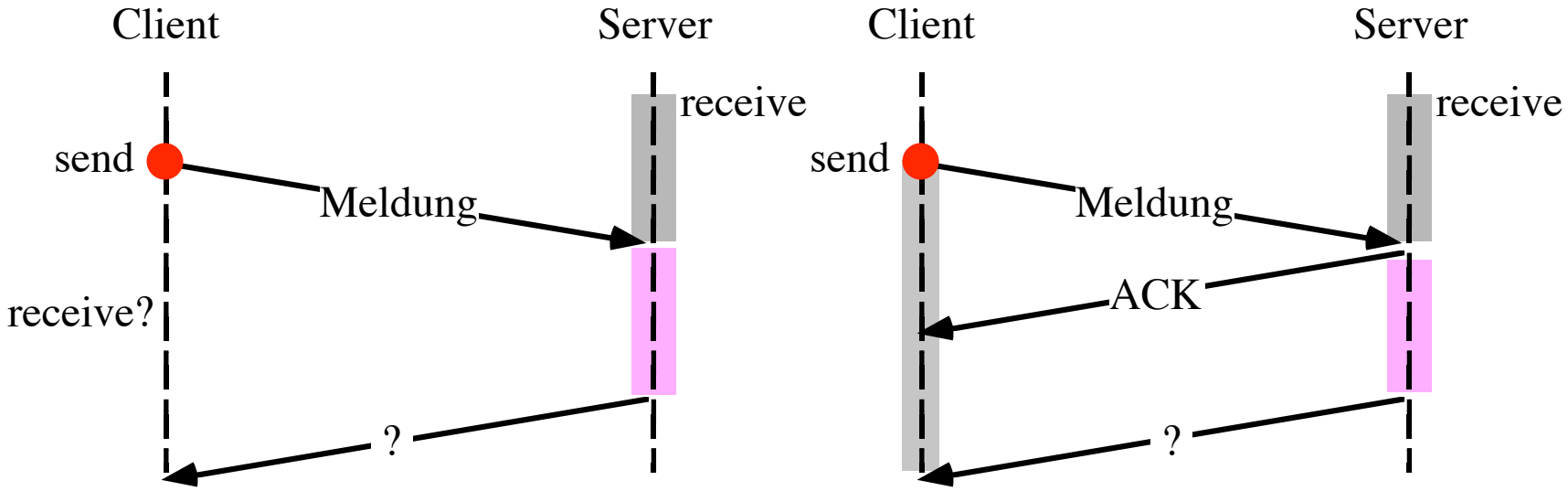




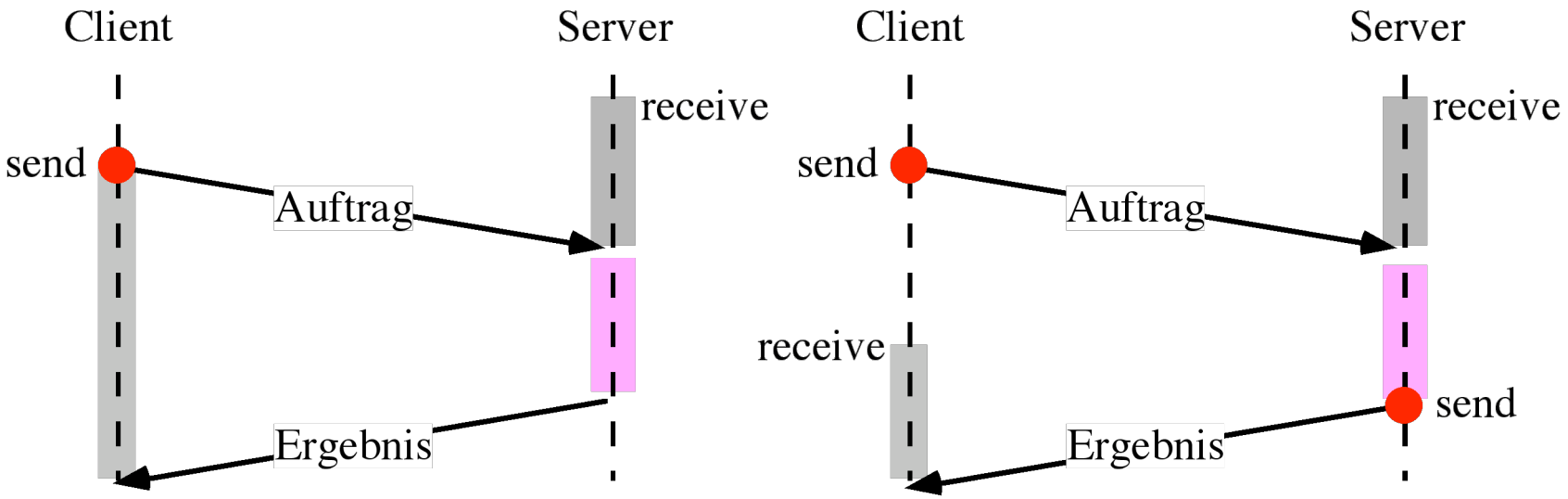
2.1.1 Kommunikationsformen

	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



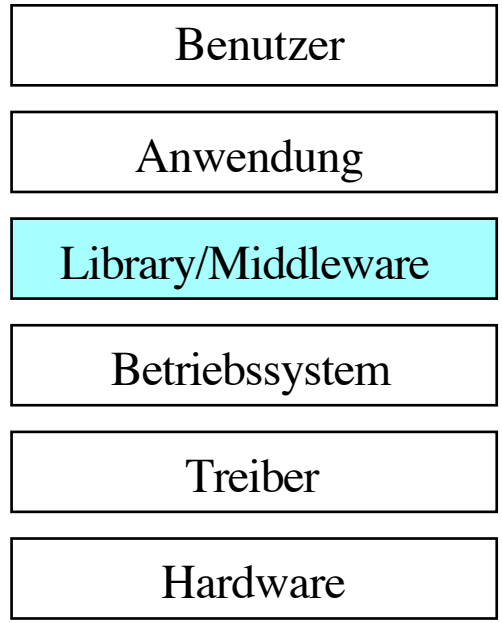
- Auftragsorientiert
  - ähnlich Prozeduraufruf
  - normal oder 'nebenläufig'
  - Parameterverpackung oft implizit
  - Versand oft implizit

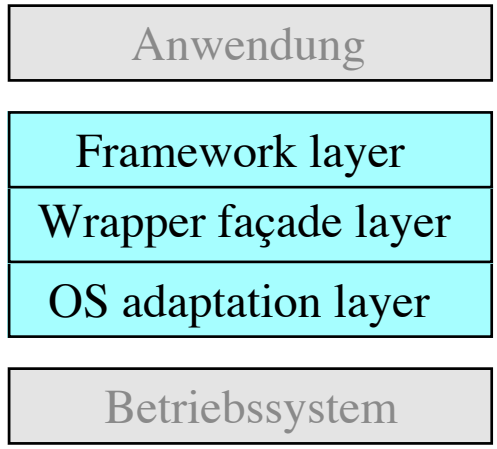


- Empfindlich gegen Fehler

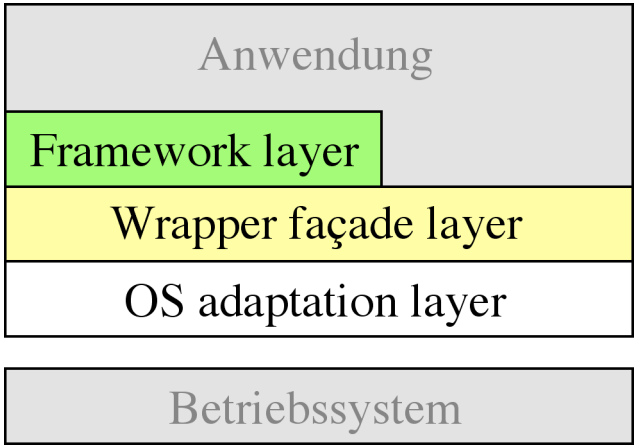
2.1.1.0 Beispiel: Toolkit für Verteilte Systeme

- 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

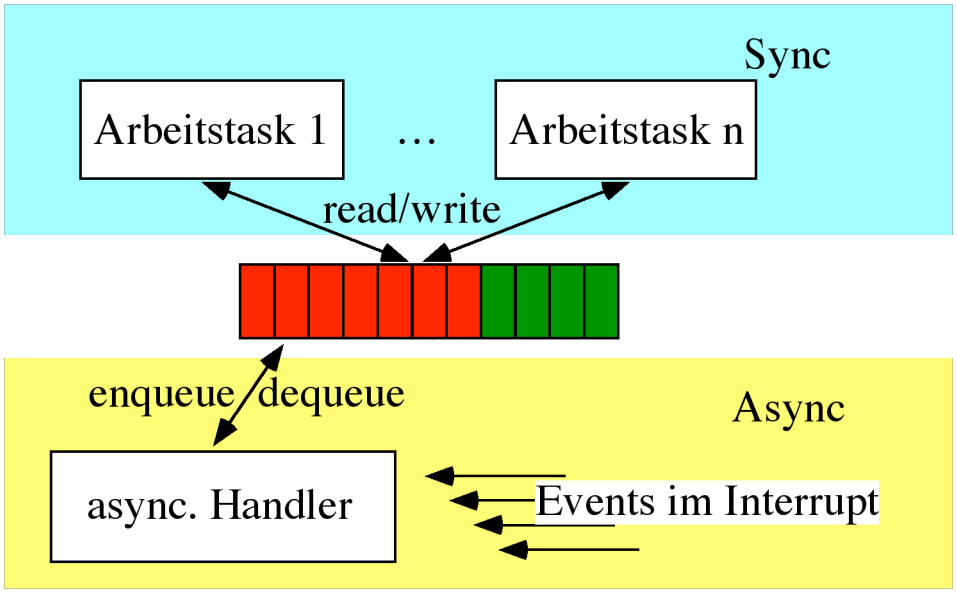




- Weitere Wrapper 'Façades'
  - Event Demultiplex
  - Process, Threading
  - Synchronization: Guard, Mutex, Locks, Semaphor
- Events: Anreize für Programm
  - Bsp: Paket, Mouse-Click, Request
  - klassische Programme haben 'Eventloop'
  - Eventloop verteilt Events an Service-Funktionen
  - Eventloop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



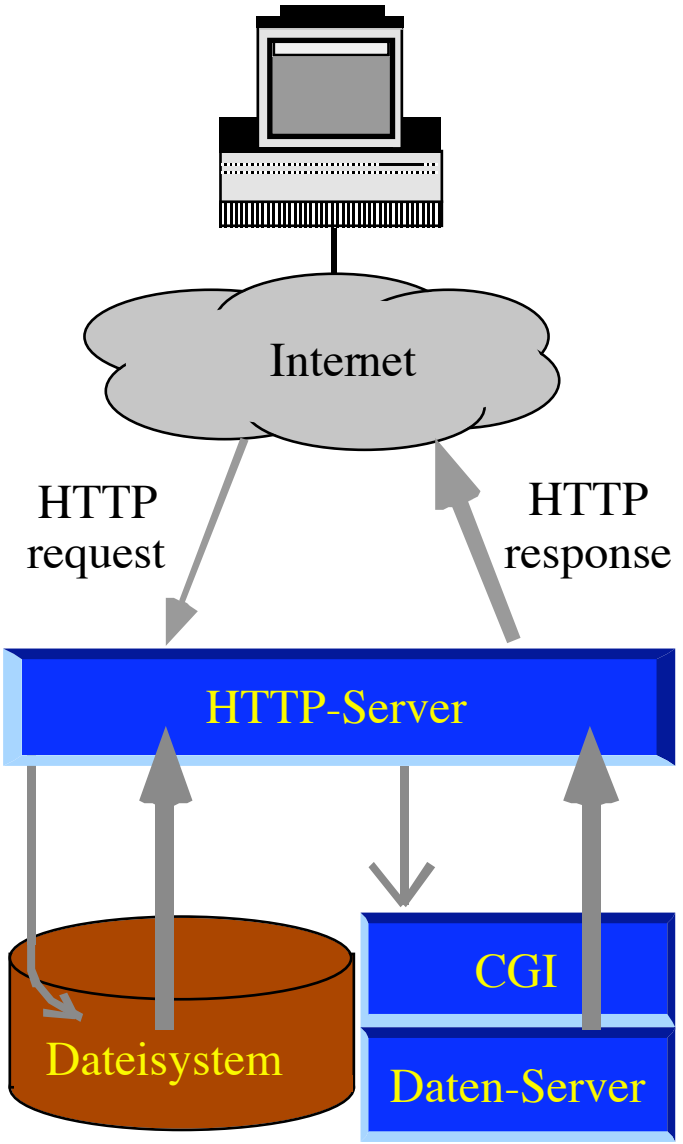
- Design-Patterns
  - Entwurfsmuster, Standard-Vorgehensweise
  - Schmidt et al. haben Patterns erarbeitet, in ACE implementiert
- Reactor
  - register_handler(handler: ACE_Event_Handler *, mask: …):int
  - Programm übergibt Kontrolle: handle_events(): synchron
- Proactor
  - asynchroner I/O
  - ACE 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'



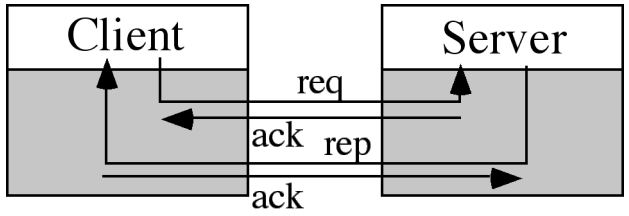
2.1.1.1 Client-Server

- Beispiel Webserver
  - http als Protokoll
  - URLs und html als Nutzlast
- Request-Typ (Methode)
  - **GET**, PUT
  - POST, DELETE, ...
- Response-Typ / Code
  - **OK** 200 (2xx), Error 4xx, 5xx
  - No Response 204, Redirection 3xx, ...
- Objekt-Typ
  - Beispiel Response:

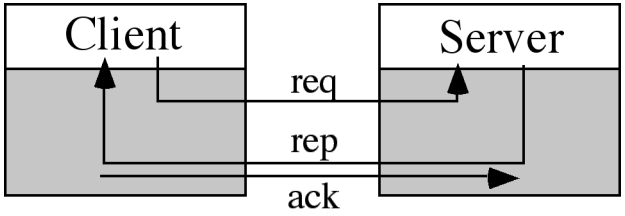
```
HTTP/1.0 200
Content-type: text/plain
Expires: Sun 26 Mar 95 17:50:36
GMT

Dies ist ein Beispieltext
```



- 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)



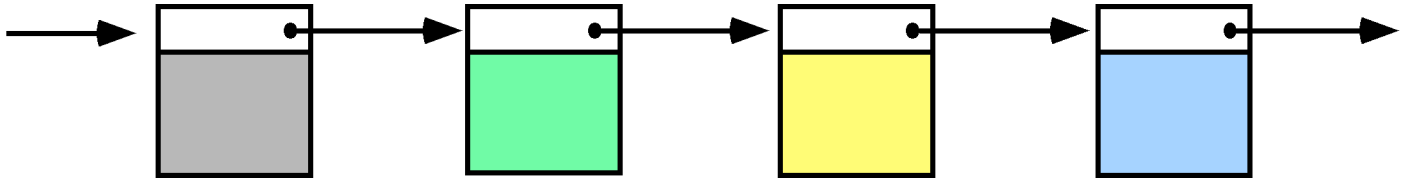
- Zuverlässigkeit 'Anwendungssache'
  - Semantik-abhängig
  - Sicherung in der Anwendung: timeouts, …
- Parameter meist selber verpacken
  - Standards für Format und Protokollfelder

• 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

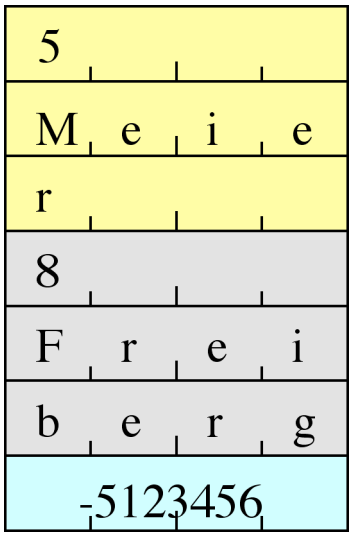


- Bäume
  - durchlaufen und elementweise verpacken
  - in-order, post-oder, pre-order?
  - allgemeine Graphen?

- 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



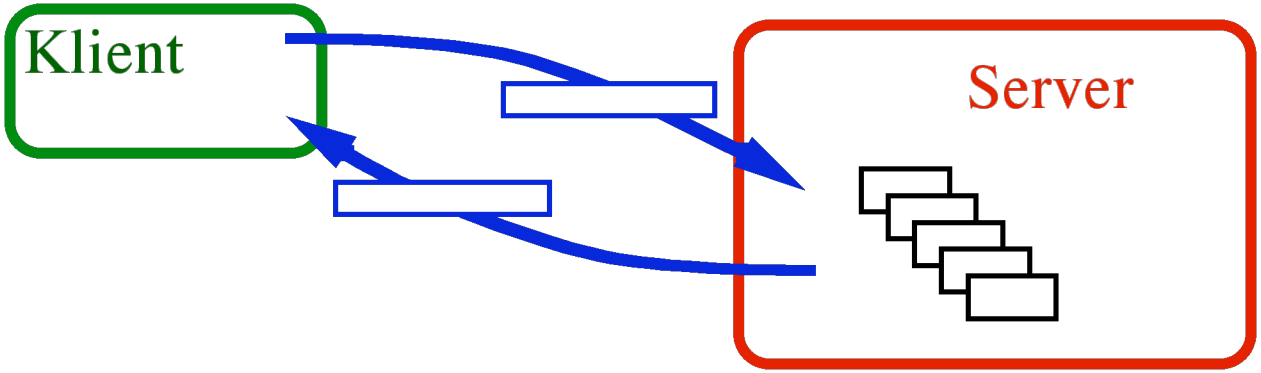
- Marshalling
  - Vorgang des Ein- und Auspackens
  - in der verteilten Anwendung

```
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

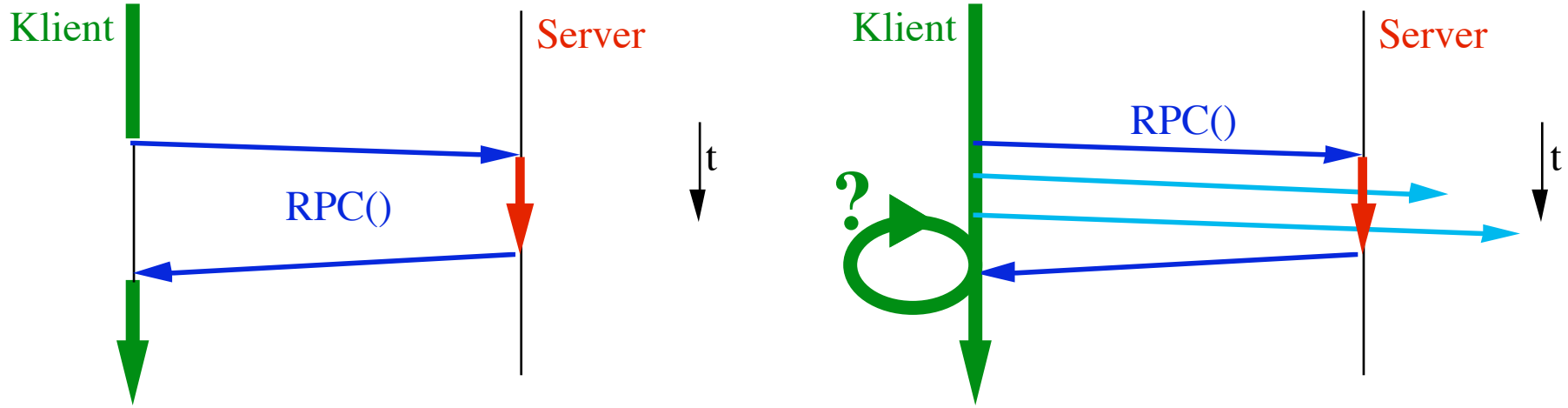
2.1.1.2 Remote Procedure Call

- Nachrichtenbasierte Kommunikation zwischen Knoten im Netz



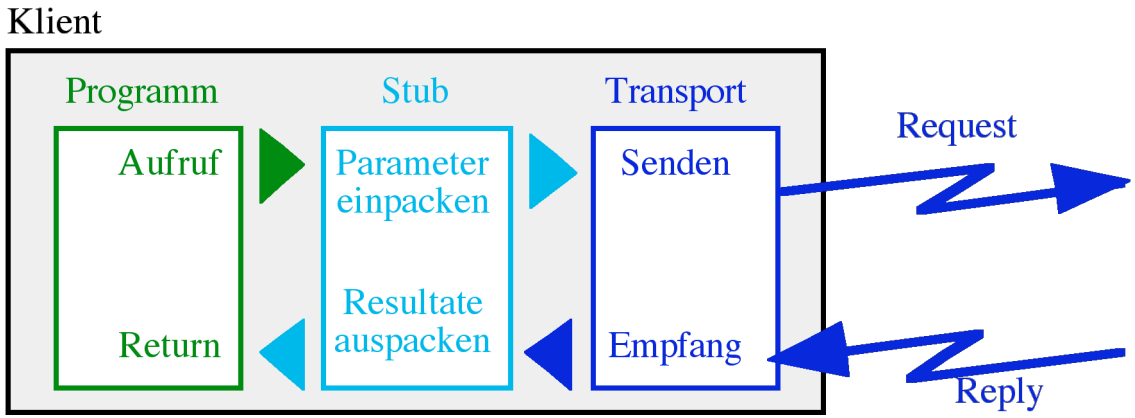
- Teilweise vergleichbar einem lokalen Prozeduraufruf
  - Parameter an die ferne Prozedur übergeben
  - Resultate an Klienten zurückliefern (Return)
  - eventuell warten
- Probleme im Klienten-Programm
  - globale Variablen
  - äußere Prozeduren
  - Zeiger auf Datenobjekte im Heap
  - komplexe Datenstrukturen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

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



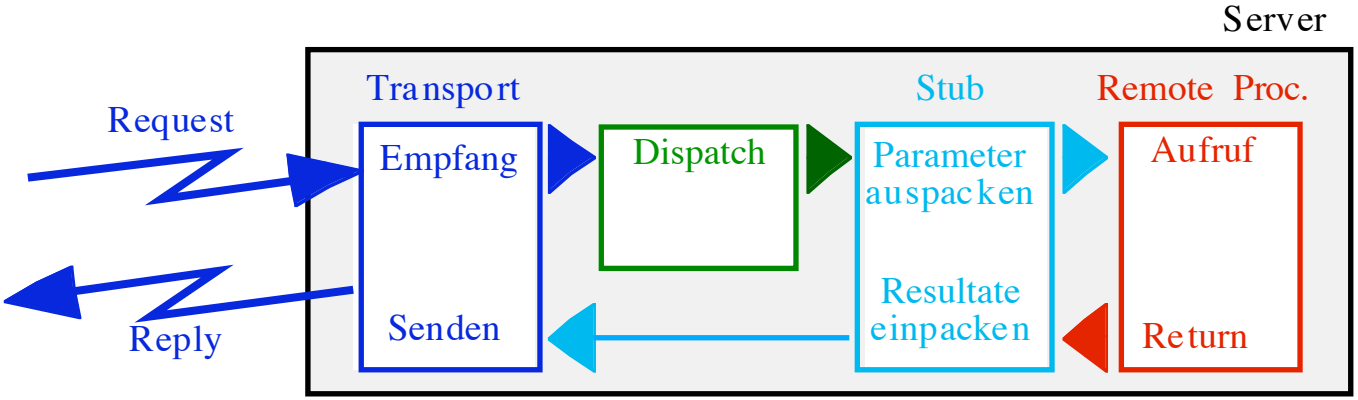
- Asynchroner RPC:
  - Klient läuft weiter
  - kann weitere RPCs absetzen
  - explizites Abfragen, ob Antwort schon da
  - oder "Completion routine" oder HW-Interrupt ...

- 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)

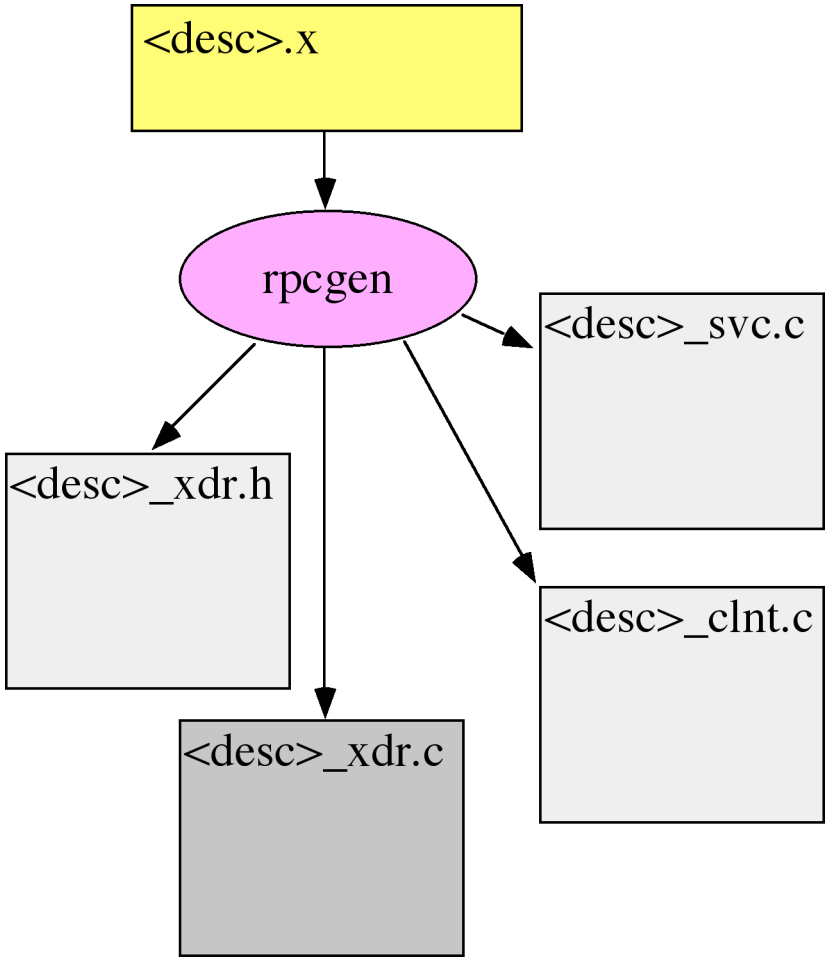


- Rückgabe der Resultatparameter
  - netzkonforme Darstellung herstellen (einpacken)
  - an Transportsystem übergeben
- Zuverlässigkeit

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) {
    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);
    }
```

(5)

• 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);
}
```

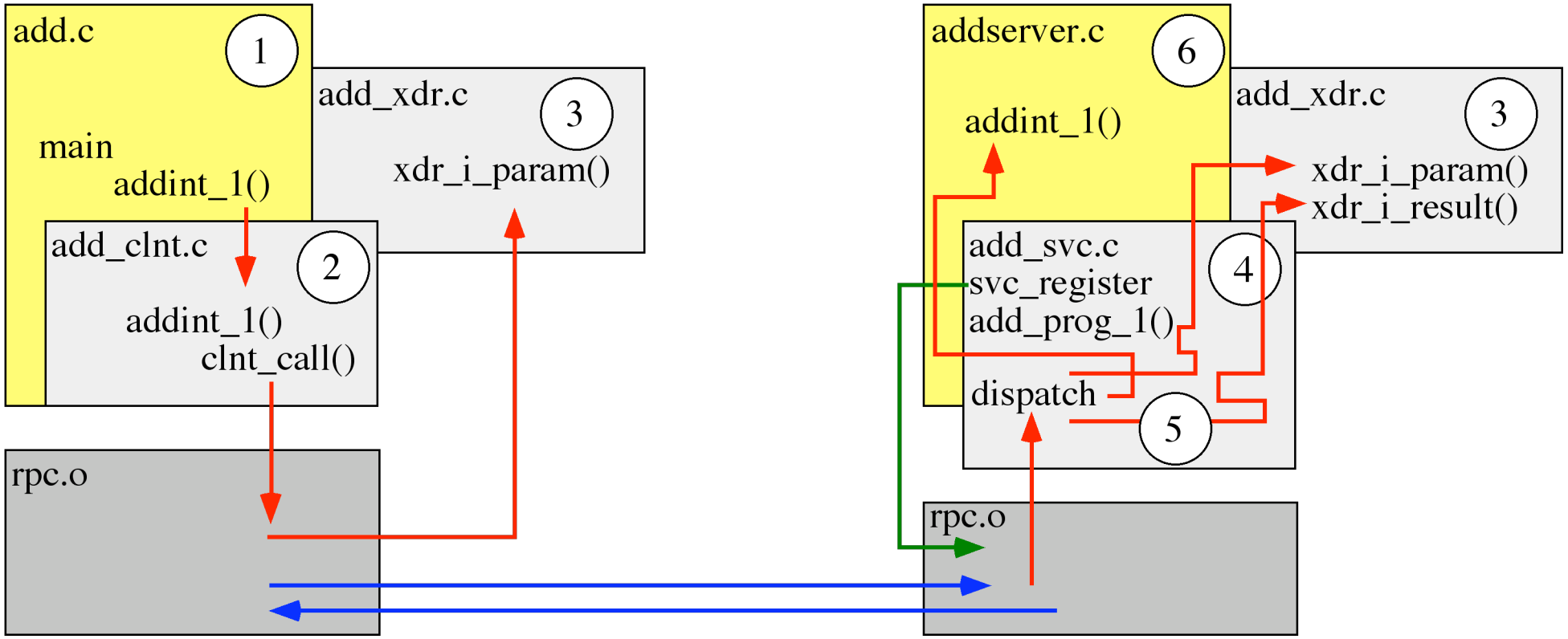
• Client-Hauptprogramm (1)

```
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);
```

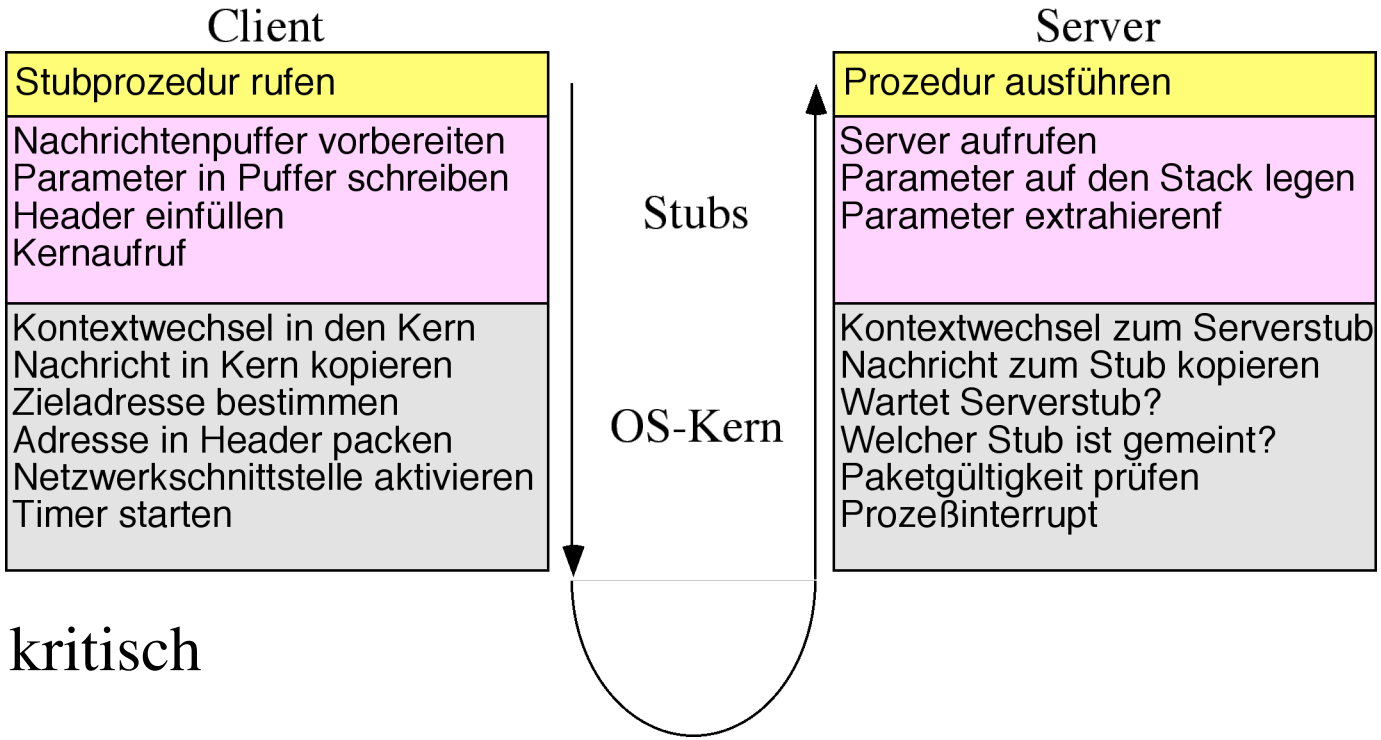
```
}
```

- 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;
/* ... */
};
```

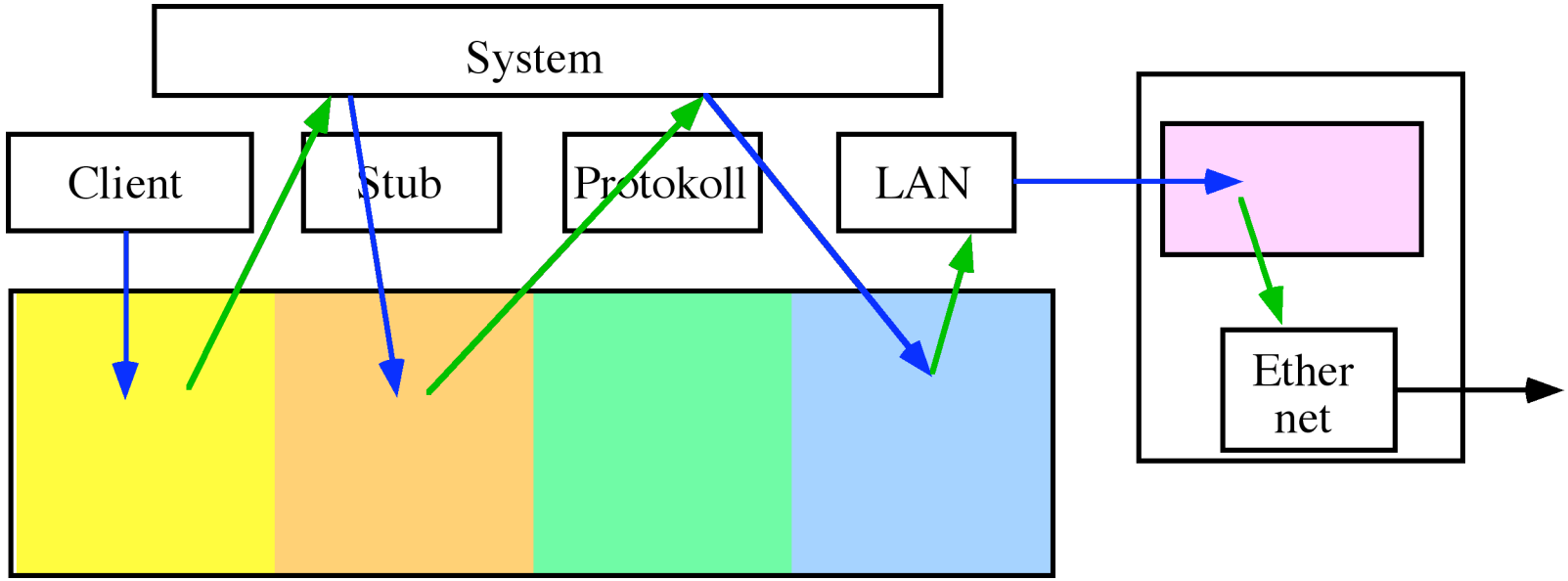

add.c
1
main
addint_1()
add_clnt.c
2
addint_1()
clnt_call()
add_xdr.c
3
xdr_i_param()
rpc.o
addserver.c
6
addint_1()
add_svc.c
svc_register
add_prog_1()
4
dispatch
5
add_xdr.c
3
xdr_i_param()
xdr_i_result()
rpc.o

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

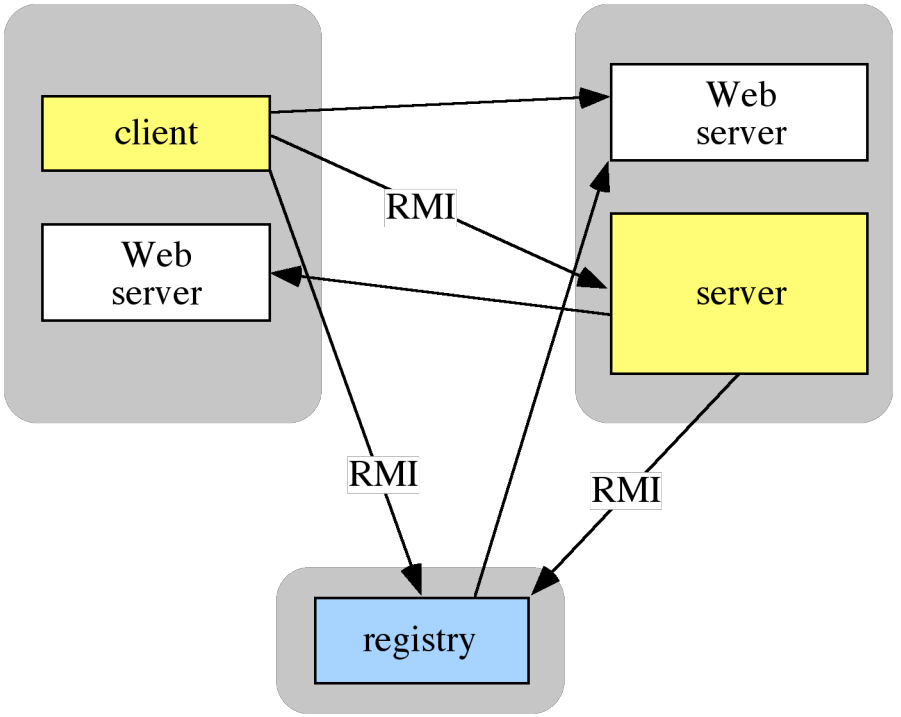


- Anzahl Datenkopien kritisch

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

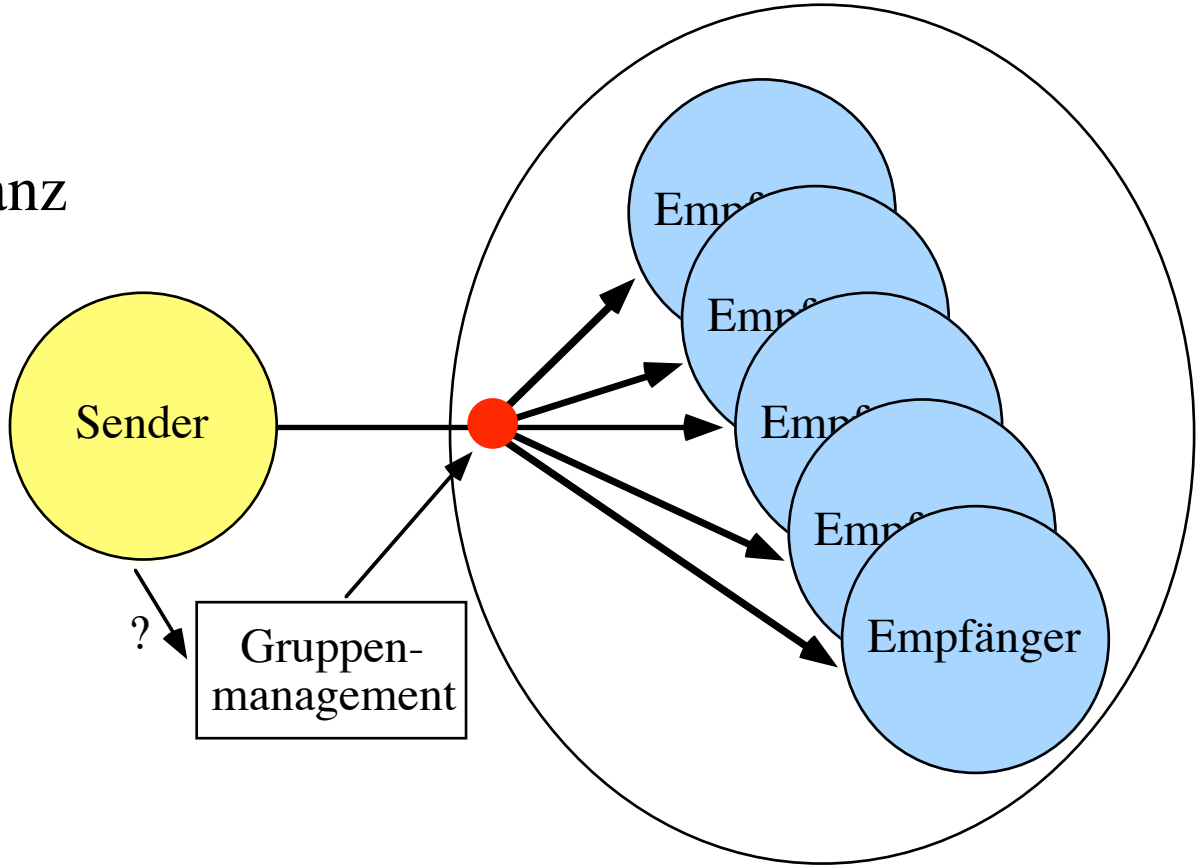


- Sonderfall objektorientiertes Programmieren
  - Selektor an entferntes Objekt schicken
  - remote method invocation: RMI
  - Serializable Java object
  - keine IDL
- Distributed Object Application
  - RMIRegistry
  - WWW-Transfer des Bytecodes
- Remote Interface
  - macht Objekte benutzbar
  - **`java.rmi.Remote`**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

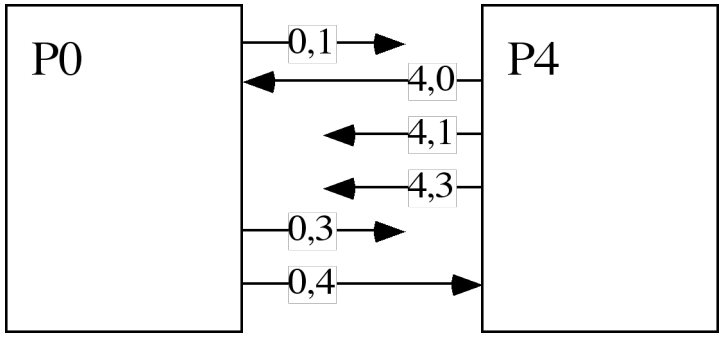


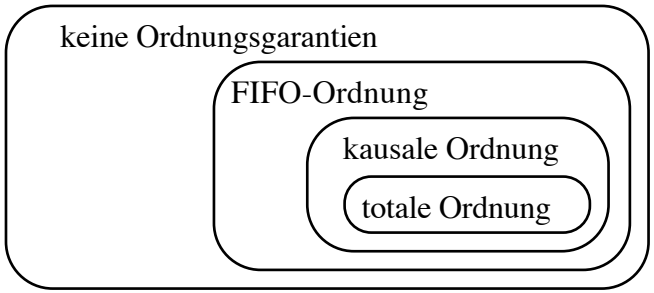
2.1.2 Gruppenkommunikation

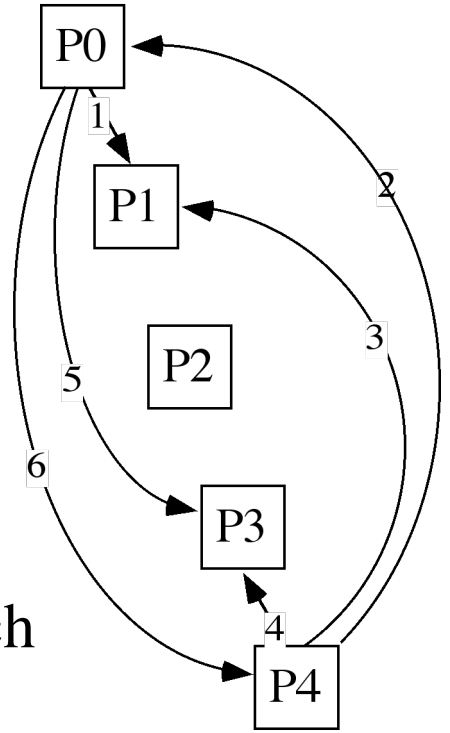
- 1:m Kommunikation
  - Replikation und Fehlertoleranz
  - Lookup (Objekte, Dienste)
  - Konferenzsysteme
  - …
- Gruppenzugehörigkeit
  - statisch oder dynamisch
  - join, leave, kick, …
  - create, discard, …
  - deterministisch?
- Zugangsregeln
  - offen oder geschlossen
  - Rechte: read, write, …
- 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: $e_1 <_k e_2$, $e_2 <_k e_3$ => $e_1 <_k e_3$

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

Statements	Einfacher Stackcode
`a:= 0;`	`const(0); write(a)`
`b:= 0;`	`const(0); write(b)`
`a:= 10;`	`const(10); write(a)`
`b:= 20;`	`const(20); write(b)`
`c:= a+ b;`	`read(a); read(b); add; write(c)`

- 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;*	

- Inkorrekte Ausführungssequenzen
  - Verzögerung von Nachrichten im Netz
  - präemptive Prozessumschaltungen
  - die Reihenfolge der Lese- und Schreib-Operationen unbestimmt

c = 0:

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

c = 10:

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

- Einige Sequenzen mit korrektem Resultat:
  - pro Variable Zugriff in der richtigen Sequenz garantieren

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

=30	=30	=30	=30	=30

- Nicht sequenzerhaltend, aber korrektes Resultat

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

=30	=30	=30	=30	=30

  - für kausal nicht zusammenhängende Operationen kann die Reihenfolge auch innerhalb einer CPU vertauscht werden

- Pro Variable Zugriff in der richtigen Reihenfolge garantieren
  - z.B. Locks

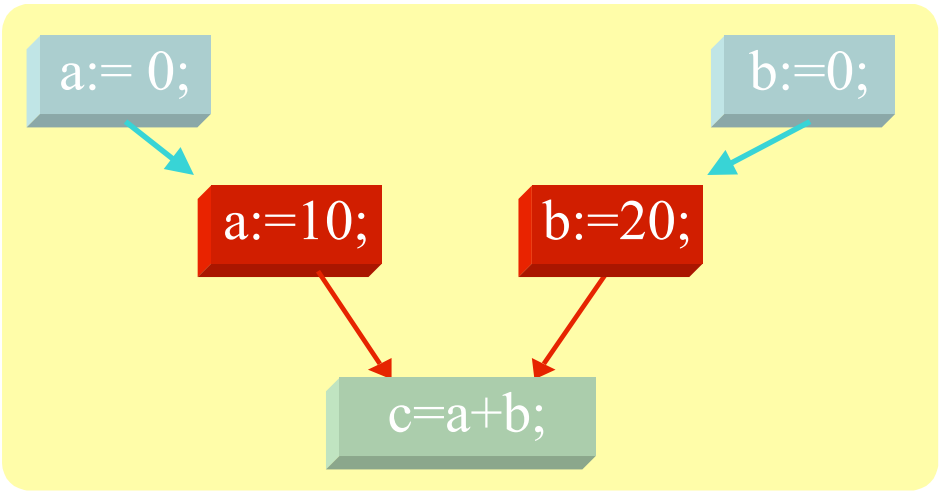
```
30:
```

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

- Kausaler Zusammenhang zwischen:

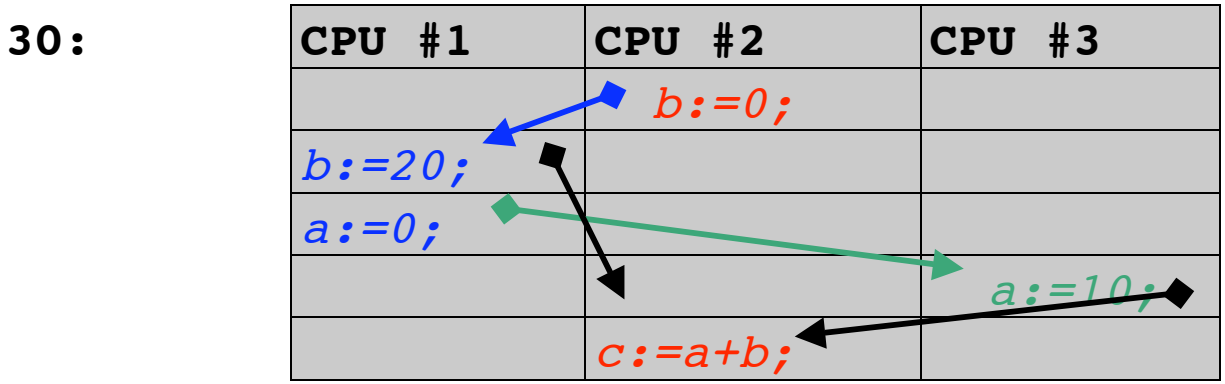
```
"a:=0;"   und   "a:=10;"
"b:=0;"   und   "b:=20;"
"a:=10;"  und   "c:=a+b;"
"b:=20;"  und   "c:=a+b;"
```

  - „a:=0“ und „b:=0“ überflussig …



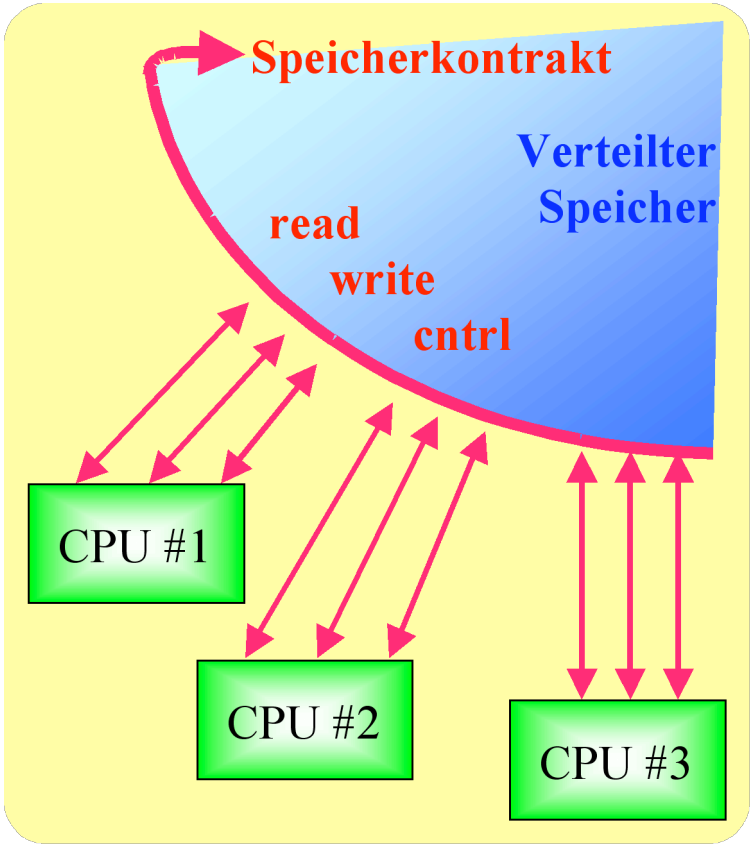
- Datenflussgraph
  - Abhängigkeiten zwischen Instruktionen
  - berücksichtigt „Happens before“ Relation
  - Operationen ohne Abhängigkeit können nebenläufig stattfinden
  - weitergehende Nebenläufigkeit bzw. Parlellisierung kaum realisierbar

- Relaxierung:
  - Beispiel vertauscht die Reihenfolge von Instruktionen
  - trotzdem korrektes Resultat, da der Datenflussgraph berücksichtigt



- Kausal nicht zusammenhängende Operationen
  - Reihenfolge kann auch innerhalb einer CPU vertauscht werden
  - auch durch parallelisierende Compiler ausgenutzt
  - es gab Prototypen von Datenflussmaschinen
  - Abhängigkeiten zwischen Instruktionen an die Hardware delegieren
- Multiprozessorkonsistenz
  - aus Sicht der einzelnen CPUs Speicherzugriffe streng sequentiell
  - die Sicht auf den Speicher bleibt "konsistent"
  - ausgefeiltes Bus-Protokoll sorgt für scheinbar sequentiellen Zugriff

- Speicherungsmechanismus
  - Sichtbarkeit von Schreiboperationen nach verschiedenen Regeln
- Beispiele
  - sofort, bzw. getaktet
  - sichtbar nach Anforderung
  - gleiche Sicht für alle Stationen
  - konsistent aus der Sicht einer Station
  - entsprechend einer verteilten Zeitskala
  - geordnet nach Zeitstempelverfahren ...
- „Kohärenz“:
  - bezüglich einzelner Cache-Zeilen
  - bezüglich einzelner Speicherwörter
  - einheitliche Sicht nur auf einen Teil des Systemzustandes



- Was geschieht mit Variablen in Registern bei einer Prozessumschaltung ?
- Konsistenzmodell: 'Vertrag zwischen Programm und Speicher'
  - Garantien für Ergebnis des konkurrierenden Datenzugriffs
  - Ergebnismenge für gegebene Zugriffssequenz

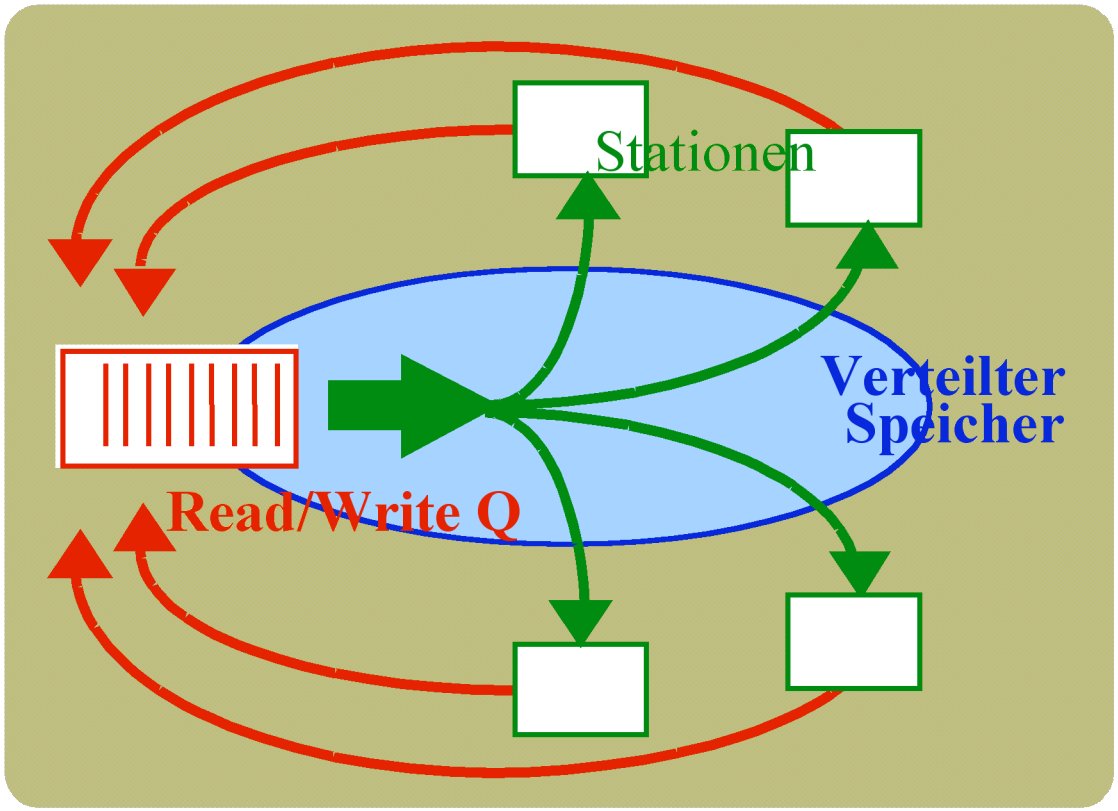
- Strikte Konsistenz
  - alle Leseoperationen liefern den 'zuletzt' geschriebenen Wert
  - unabhängig davon welcher Prozess geschrieben hat
  - globale Zeit im verteilten System?-> Begriff "zuletzt" ist unscharf
  - Lese- und Schreiboperationen in Warteschlange
  - atomare Lese- und Schreiboperationen
  - Instruktionsausführung /Warteschlangeneintrag in starrem Zeitraster

  ~ nicht verteilt

P1:	W(x,1)		
P2:		R(x,1)	R(x,1)

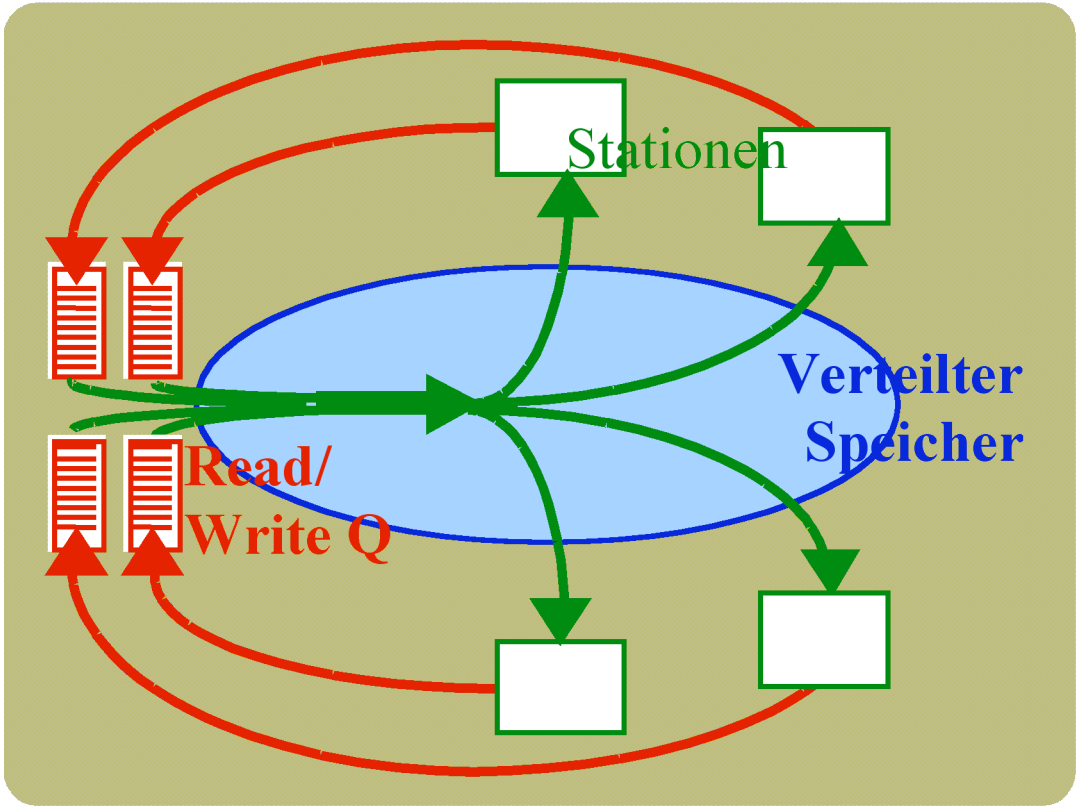
P1:		W(x,1)	
P2:	R(x,0)		R(x,1)

P1:	W(x,1)		
P2:		R(x,0)	R(x,1)

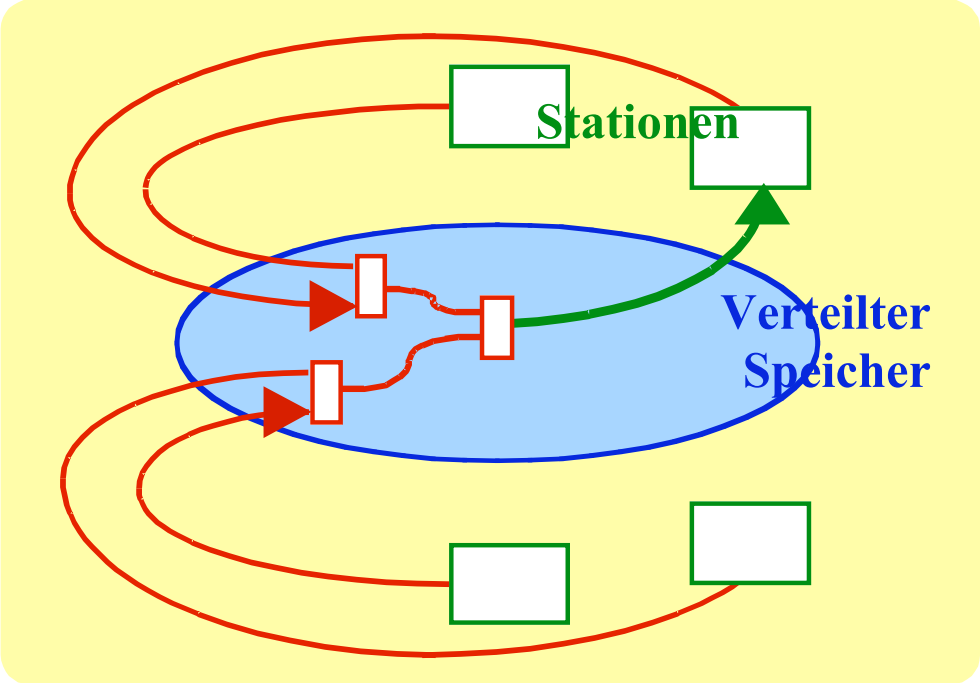


- Sequentielle Konsistenz [Lamport]
  - 'Multiprozessor erscheint wie multitasking Monoprozessor'
  - alle Zugriffe in der gleichen *Reihenfolge* wahrgenommen
  - zeitliche Verschränkung unbestimmt
  - minimale Knotenkommunikationszeit t, Lesen r, Schreiben w:
  - $r + w \geq t$

P1:	W(x,1)		
P2:		R(x,0)	R(x,1)



- Kausale Konsistenz
  - nur kausal voneinander abhängige Operationen global geordnet
  - Semantik des sequentiellen Programmes -> kausale Abhängigkeit
  - evtl. explizite Synchronisierung eines parallelen Programmes
  - Rechnersystem muß Datenflussgraphen realisieren
  - siehe Vektorzeit

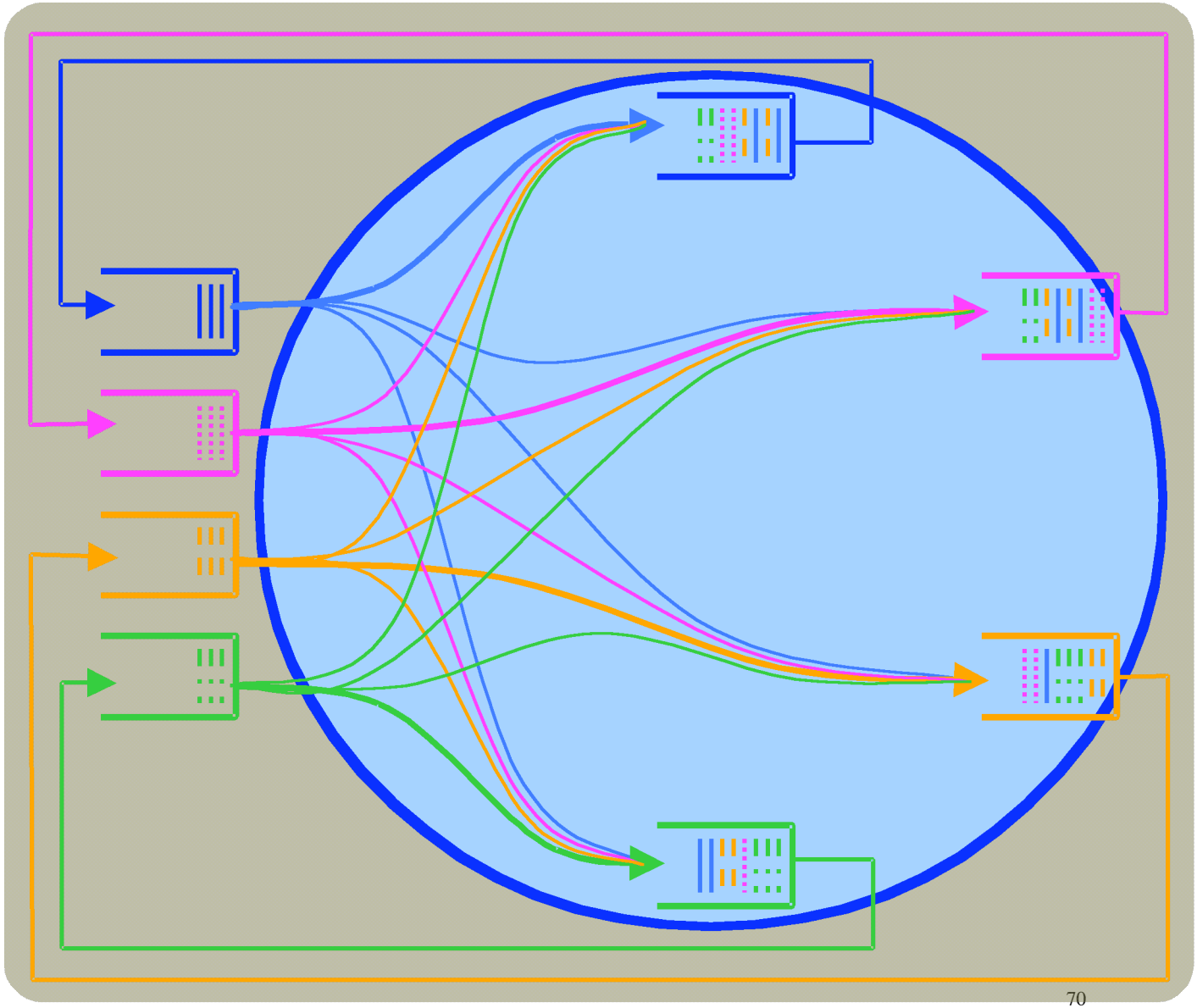


- PRAM Konsistenz: Pipeline RAM (FiFo Konsistenz)
  - Schreibreihenfolge innerhalb eines Prozesses unverändert
  - von allen gleich wahrgenommen
  - Schreiben aus versch. Prozessen unterschiedlich verschränkbar

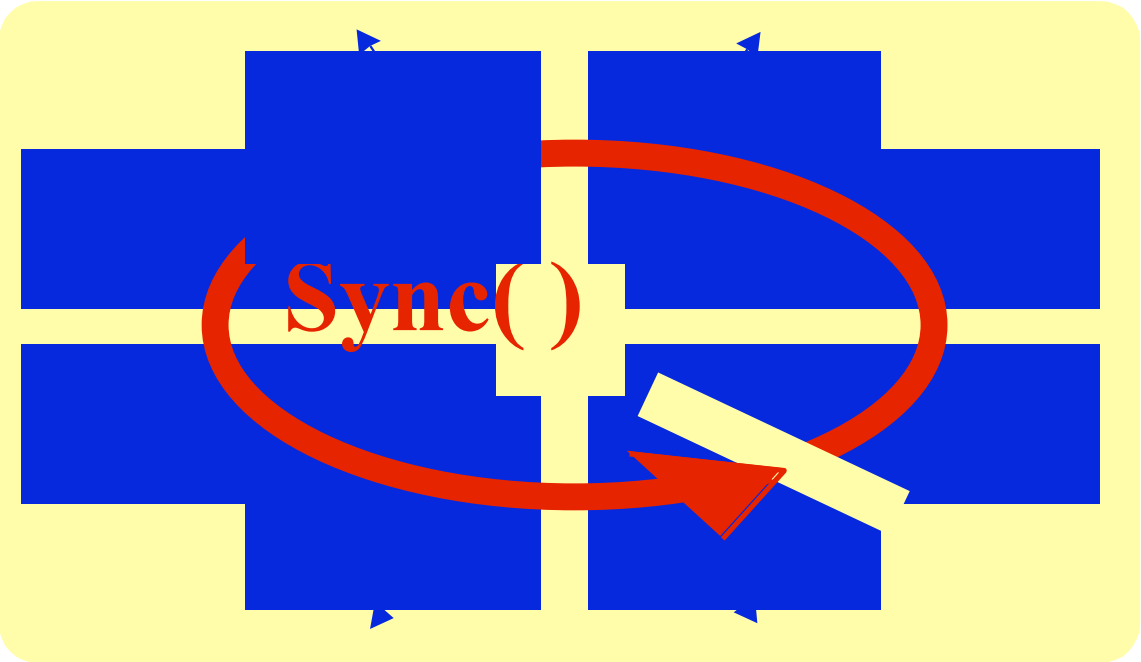
P1:	W(x,1)	W(x,2)		
P2:			R(x,2)	R(x,1)

P1:	W(x,1)	W(x,2)		
P2:			R(x,1)	R(x,2)

P1:	W(x,1)		W(x,2)	
P2:		R(x,1)		R(x,2)

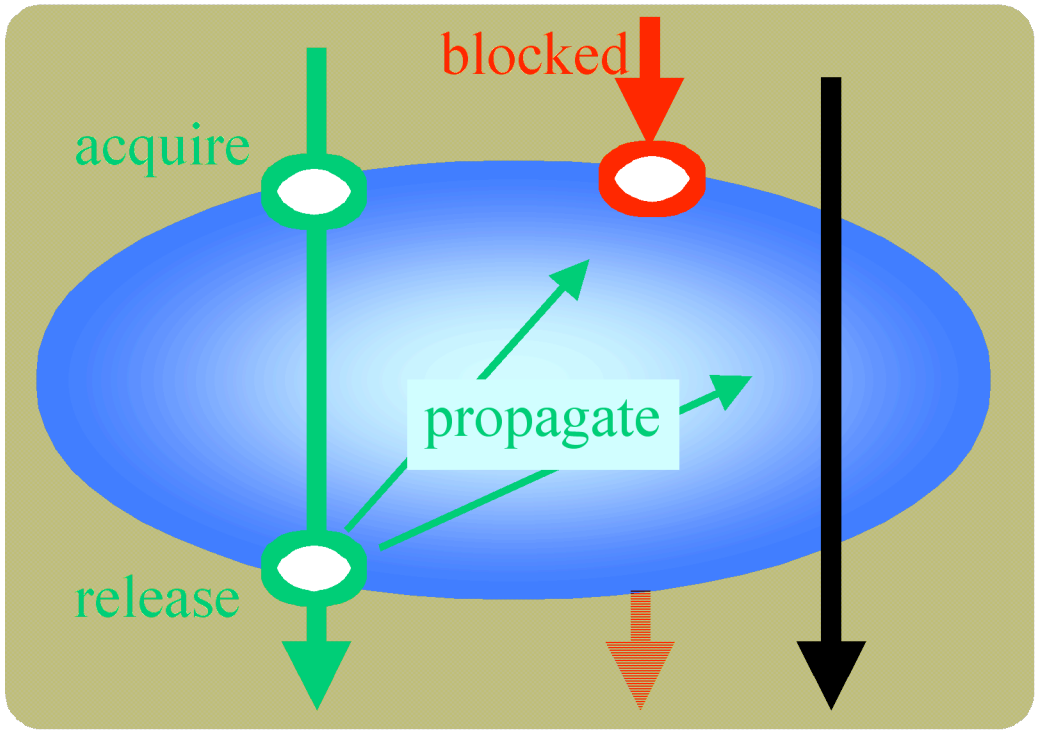


- Schwache Konsistenz
  - beteiligten Prozesse sehen Schreibzugriffe in beliebiger Reihenfolge
  - Synchronisierte Variablen:
    - Zugriff sequentiell konsistent
    - nur Zugriff nach Ausführung aller vorhergehender Schreibops.
    - kein Schreiben vor vorhergehendem Lesen

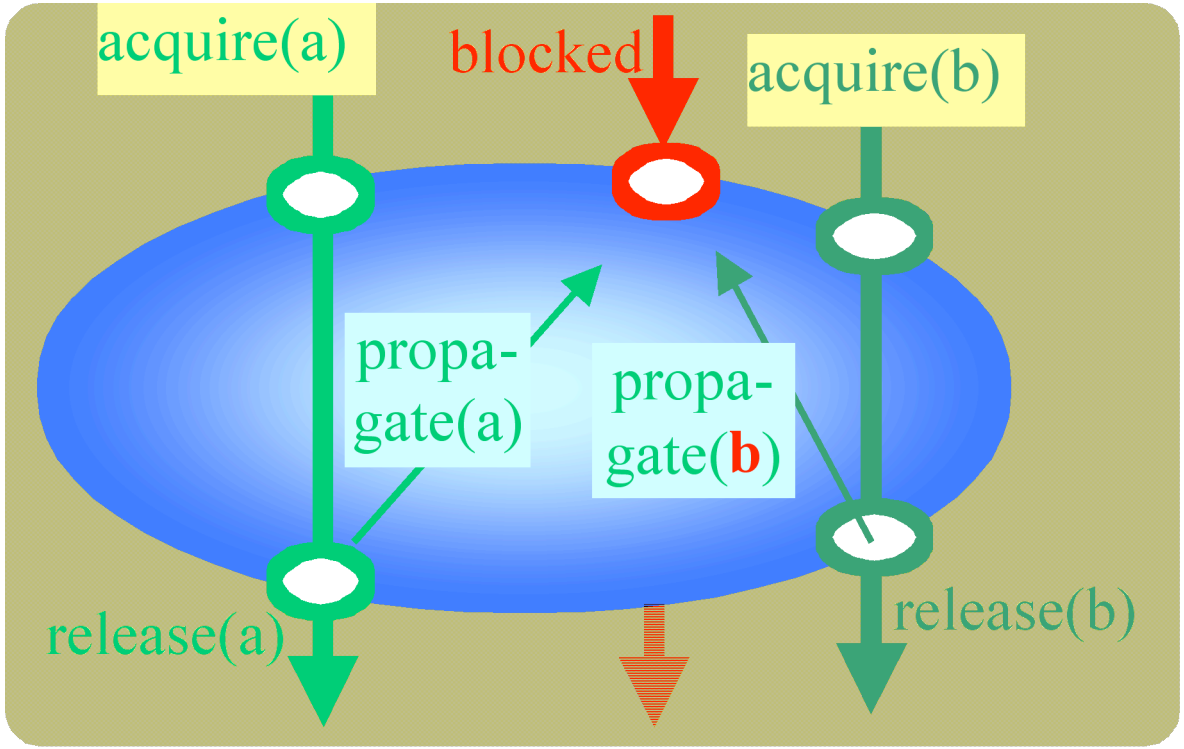


  - Update: schreibender Prozeß verschickt seine Änderunge (propagate)
  - Invalidate: schreibender Prozeß erklärt Daten für ungultig
  - bei Bedarf werden Daten angefordert

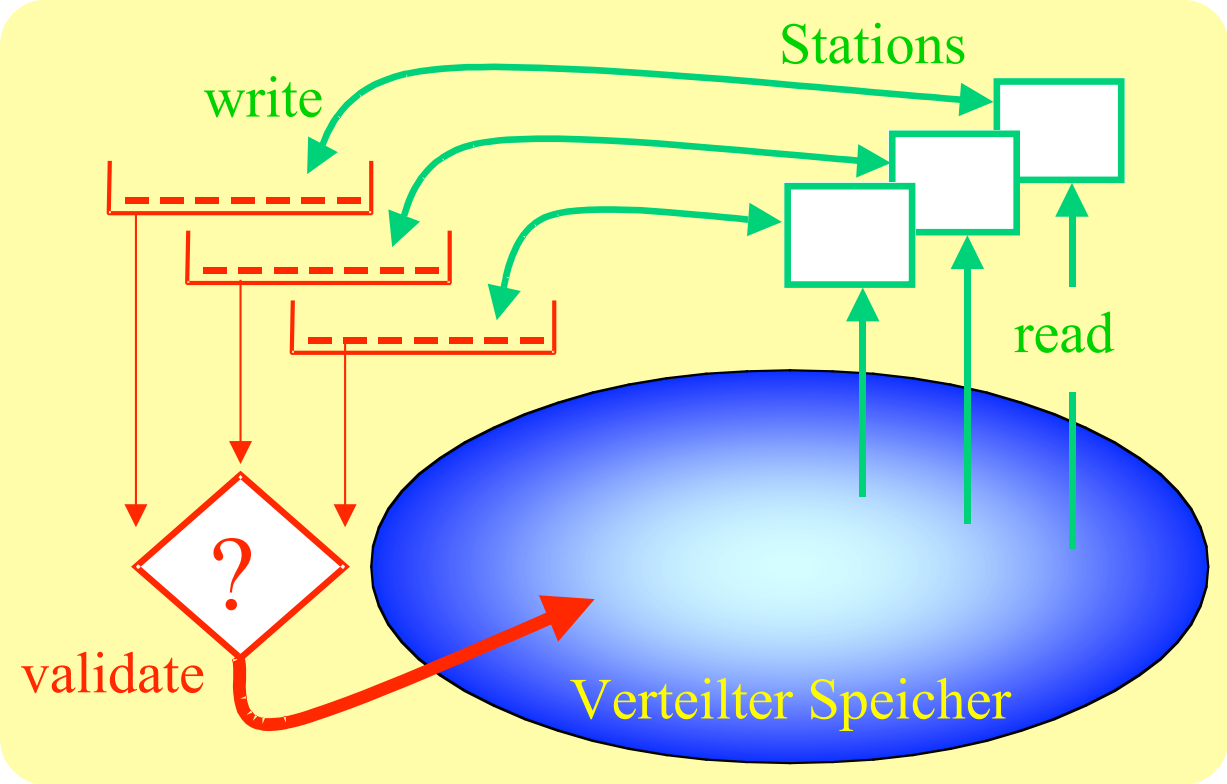
- Release-Konsistenz
  - "acquire" versucht den Eintritt in eine kritische Region
  - "release" verlässt kritische Region
  - "acquire" blockiert, falls kritische 'Region besetzt'
- Teilnehmender Prozess tritt mit "acquire" in kritische Region ein
  - arbeitet mit gemeinsamen Variablen
  - propagieren der Variablen
  - "release" rufen
- Lazy Release: nächster acquire besorgt Variablen



- Entry-Konsistenz
  - Zugriffsrechte auf gemeinsame Variablen
  - separat pro Variable
  - exklusiv zum Schreiben
  - nicht exklusiv nur zum Lesen
  - Vergabe geschützt mit Synchronisierungsvariablen (Semaphore etc.).

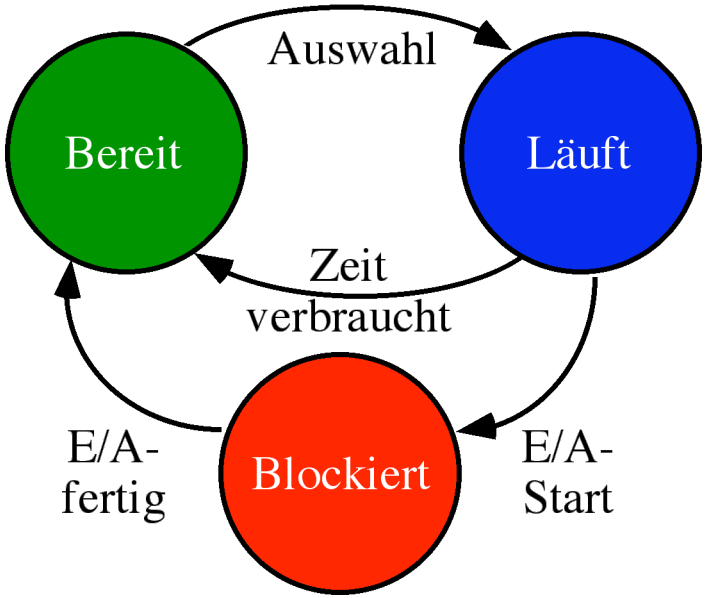


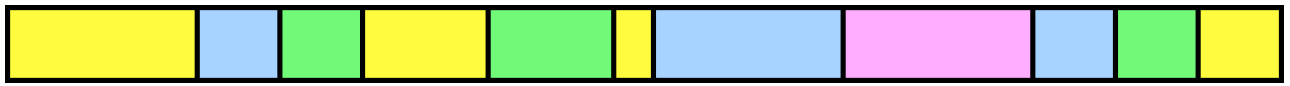
- Transaktionale Konsistenz
  - alle Schreiboperationen vorerst im lokalen Speicher
  - Berechungen in rücksetzbare Transaktionen aufteilen
  - Transaktionsende: aktualisieren bzw. synchronisieren
  - Schreib/Lesekonflikte bei der Aktualisierung: zurücksetzen
  - wahlweise update- oder invalidate-Protokoll zur Synchronisierung
  - „Optimistische Synchronisierung“



2.2.1 Prozesse und Threads

- Prozesse
  - selbständige Codeeinheiten
  - Object-Code
  - Speicher
  - Attribute
- Multi-Tasking
  - mehrere Prozesse laufen 'verschachtelt'
  - "Zeitmultiplex"
  - für den Benutzer gleichzeitig
  - verschiedene Strategien des Wechsels





- Prozesswechsel
  - Anhalten eines Prozesses
  - Speichern der Status-Information (PC, Register etc.)
  - Laden der Status-Information des neuen Prozesses
  - Wiederaufnahme der Ausführung bei der nächsten Instruktion
  - z.B. nach Zeit t, wenn gewartet werden muß, …

- Einplanung ≠ Scheduling
- Scheduler
  - gibt Ausführungsrecht befristet ab an Benutzerprozesse
  - verteilt Ausführungsrecht 'gerecht'
  - hat besondere Rechte
  - kennt alle anderen Prozesse
- Rückwechsel vom Benutzerprozeß zum Scheduler
  - freiwilliger Sprung in den Scheduler
  - erzwungen nach Fristablauf (=> Unterbrechung)
- Prozeßeigenschaften
  - Wichtigkeit (Priorität)
  - benutzte Ressourcen (Festplatte, Drucker, …)
  - verbrauchte Zeit und Deadlines
  - Adressraum
  - Zustandsinformation

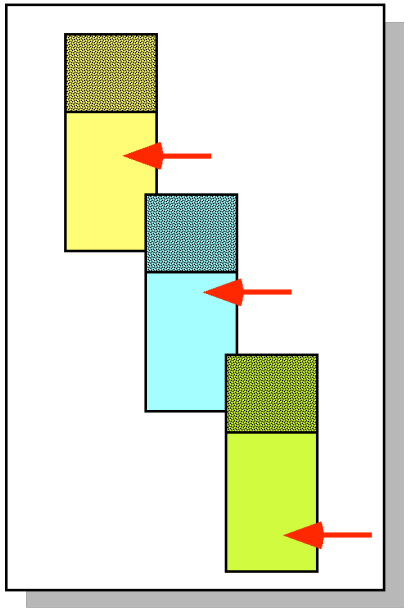
- Non-Preemptive Scheduling
  - Prozesse nicht unterbrechbar

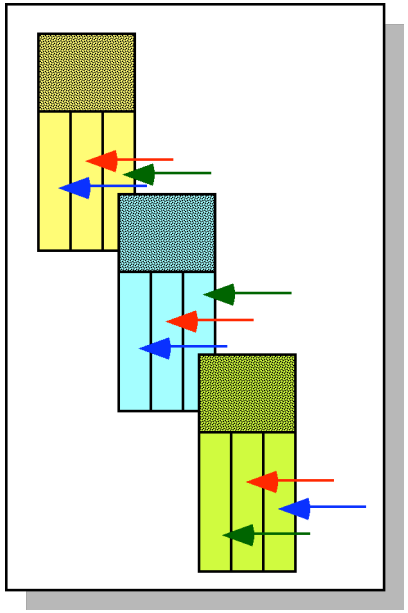
  - Kritische Prozesse können nicht unterbrochen werden
- Preemptive Scheduling
  - Prozesse durch andere Prozesse mit höherer Priorität unterbrechbar

  - Oft in Betriebssystemen vorhanden für CPU
  - Prozesswechsel häufig und teuer
- Prioritätsfestlegung
  - Relevanz für Gesamtaufgabe
  - nahe 'Abgabe'-Termine
  - lange gelaufen => Priorität sinkt
- Problem Prioritäts-Umkehr
  - Prozess mit höherer Priorität wird gestartet
  - Priorität anderer Prozesse steigt während der Bearbeitungszeit
  - Unterbrechung des Ersten, Thrashing

- Threads
  - lightweight process
  - Zustand
  - kein eigener Adressraum

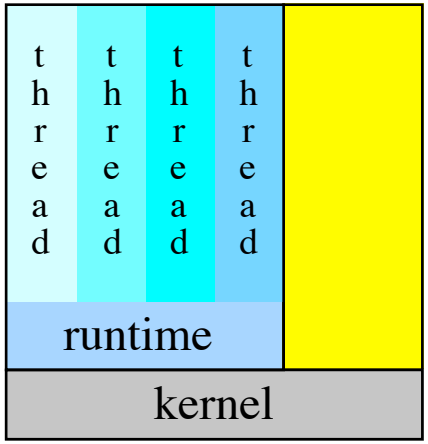
Prozess	Thread
Instruktionszähler Register und Flags Stack Kind-Prozesse	Instruktionszähler Register und Flags Stack Kind-Threads
Adressraum globale Variablen offene Dateien Timer Signale, Semaphore Verwaltung	

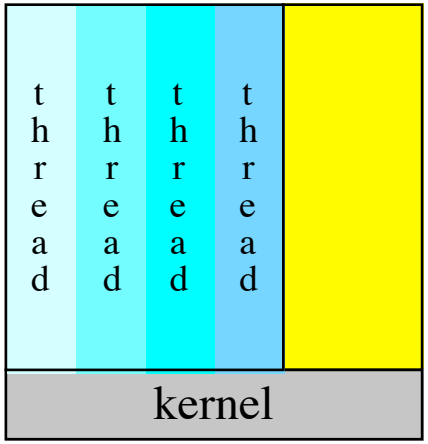




- Thread blockiert, Prozess nicht
  - einfaches Programmier-Paradigma

- Implementierung von Threads
- Scheduler im Prozess (runtime)
  - blockierende Calls in der runtime umsetzen?
  - Rechte?
  - OS unverändert, portabel
  - Scheduling kontrollierbar





- Scheduler im Kern
  - Adressraumwechsel teuer
  - neues Betriebssystem

2.2.2 Zeit in verteilten Systemen

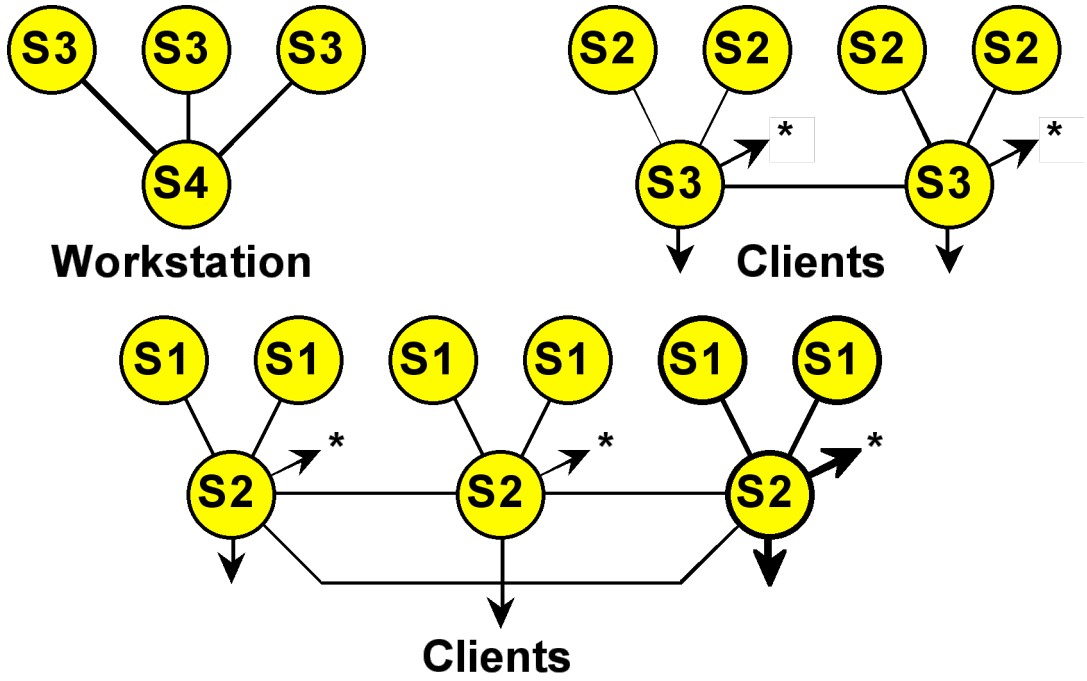
- Synchronisation
  - extern: Gleichzeitigkeit
  - intern: Intervallmessung
- Zeitstempel
  - Börse: Order-management und Zeitstempel für Transaktionen
  - Netzwerk-Managment
  - distributed multimedia stream synchronization
  - RPC at-most-once Transaktionen
  - Maßnahmen gegen replay-Angriffe
  - Experimente: 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'
  - Erddrehung wird langsamer (Gezeiten) und schwankt
- Internationale Atom-Zeit (TAI):
  - 9.192.631.770 Caesium 133 Zustandsübergänge
- Universal Time Coordinated (UTC)
  - Schaltsekunden zum Ausgleich
  - Änderung der Netzfrequenz 51 statt 50 Hz
- 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?

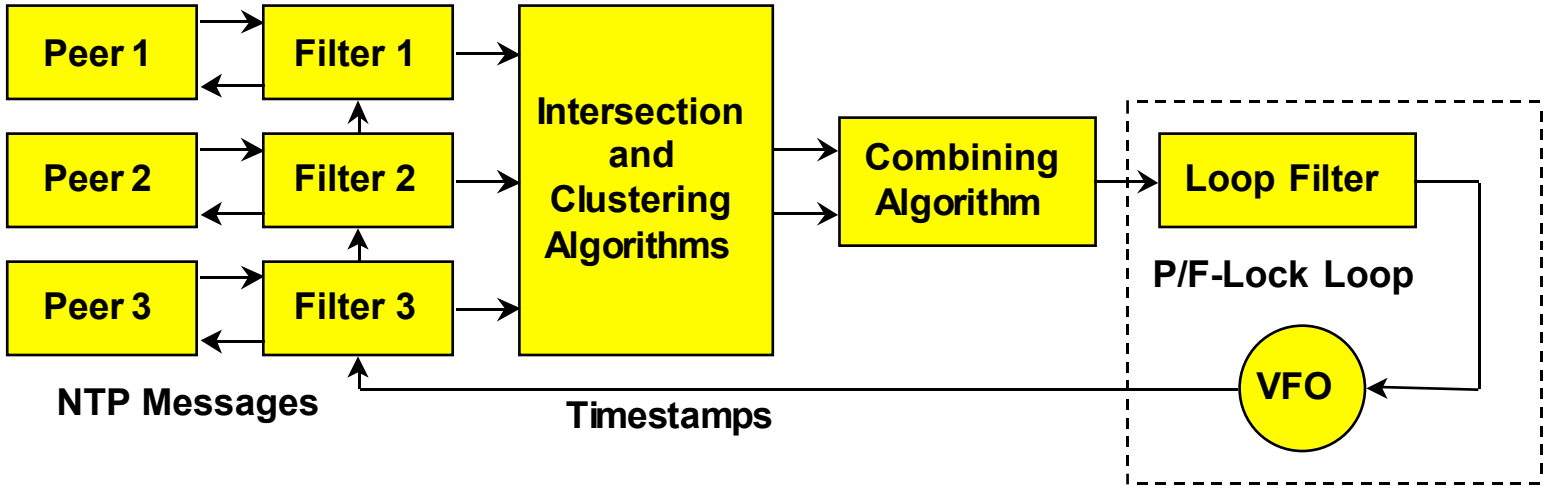
- Berkeley-Zeit-Algorithmus
  - System ohne 'gute' Zeit => Abstimmung über die Zeit
  - zentraler time-deamon fragt alle Geräte nach der Zeit
  - round-trip delay messen und berücksichtigen
  - Zeit wird gemittelt
  - jeder Klient erhält Korrekturinformation
  - Fehlertoleranzmaßnahmen
- Network Time 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



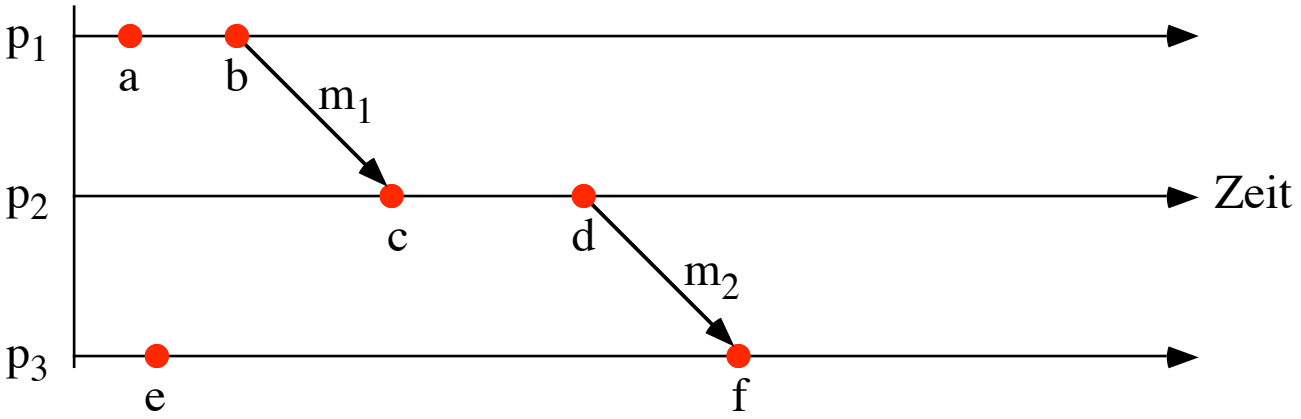
- NTP Kommunikationsmodi
  - multicast mode im LAN
  - symmetric mode synchronisiert Server-Uhren
  - procedure call mode ähnlich Christian's protocol

- 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:

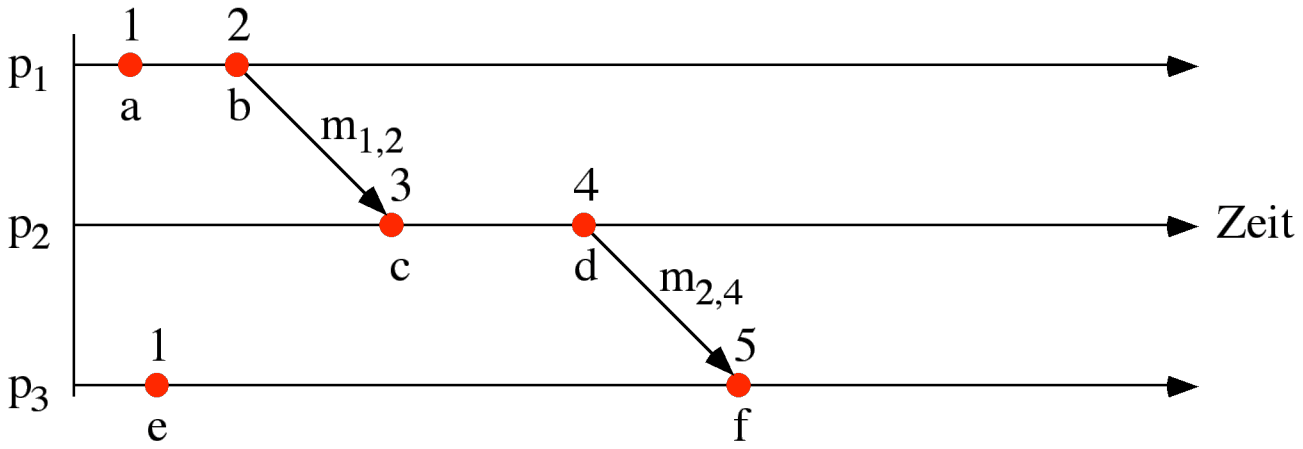
```
c_local++;
```

  - Sendeereignis:

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

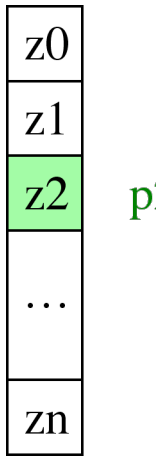
  - Empfangsereignis:

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

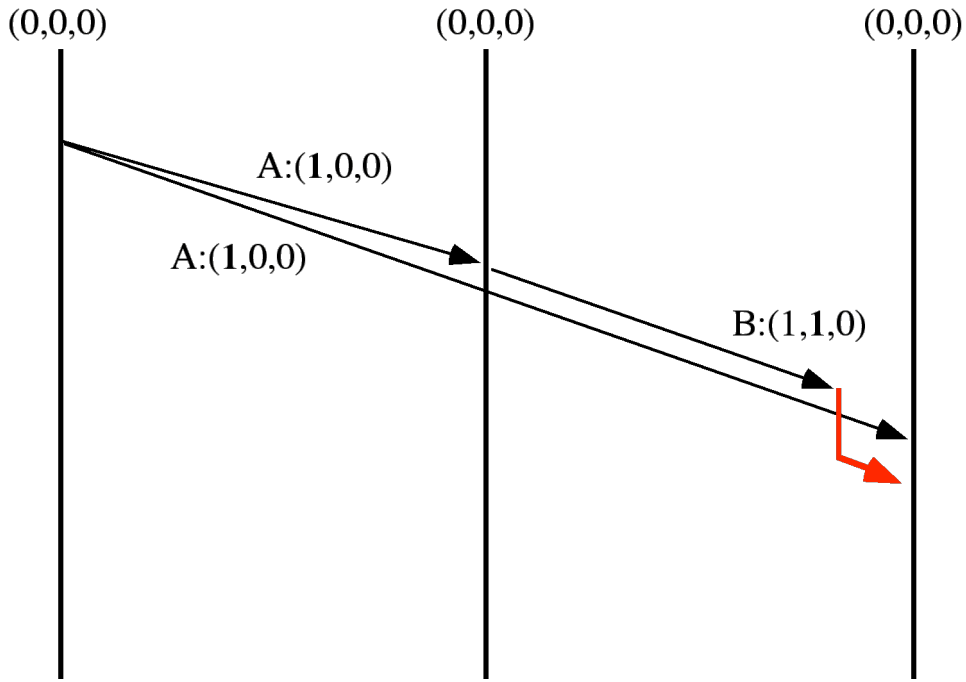


- C(a)<C(b): a→b oder allb
  - keine totale Ordnung

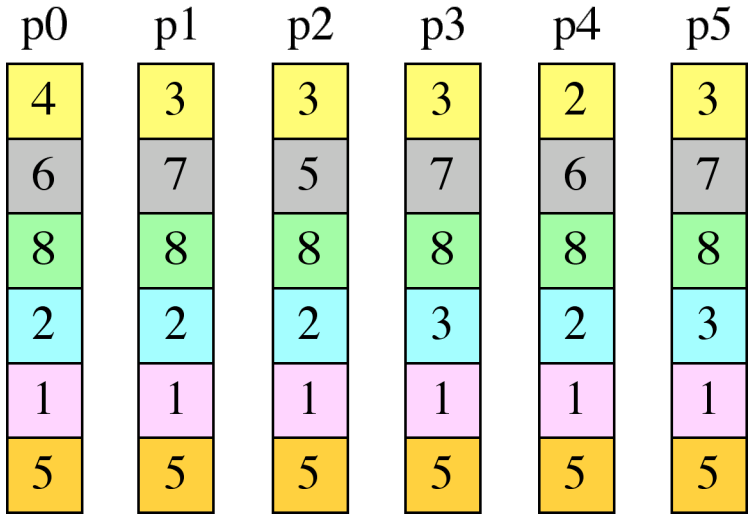
- Zeitstempel
  - jeder Prozess p_i hat *logische* Zeit z_i
  - bei Nachrichtenversand inkrementieren
- Vektorzeit
  - Zeitstempel-Vektoren
  - jede Station hält lokalen Vektor
  - Nachrichten enthalten Vektoren
  ->Vektor der Quelle beim Senden
  - Empfang: Vergleich mit lokalem Vektor
  - Empfänger verzögert evtl. Nachrichten

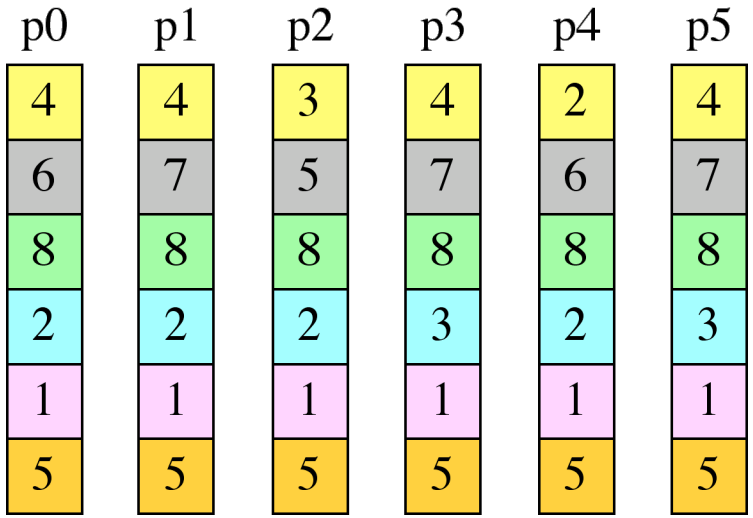


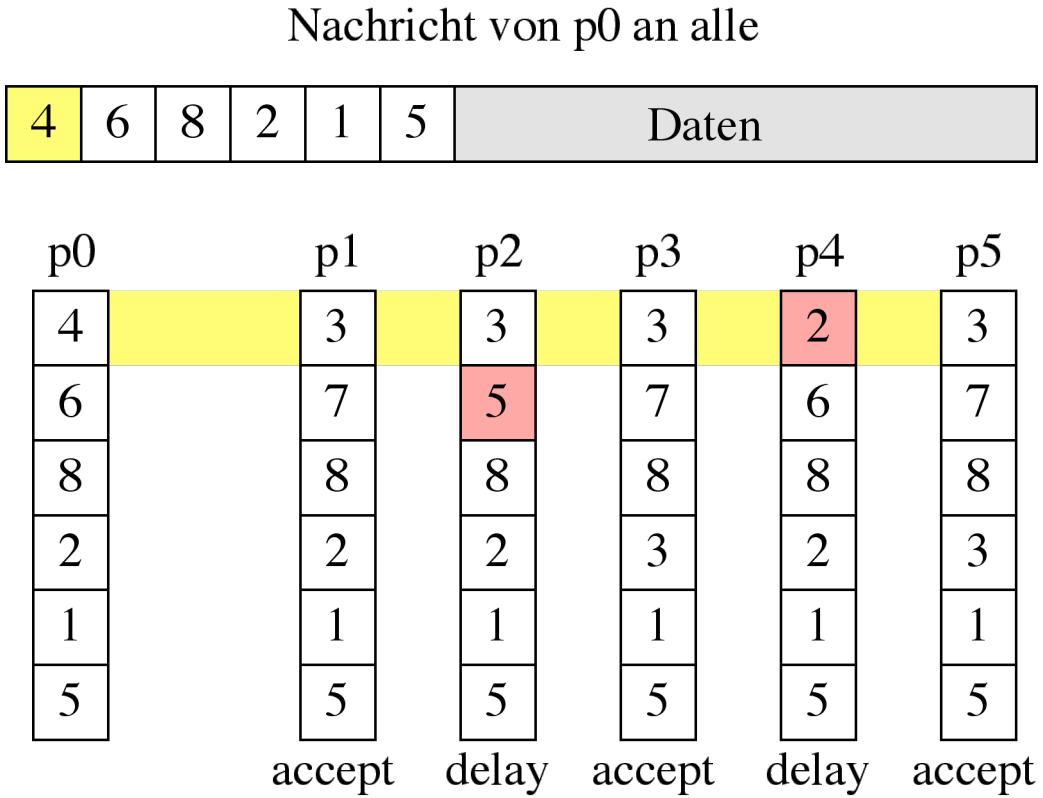
p2: inc(z2)



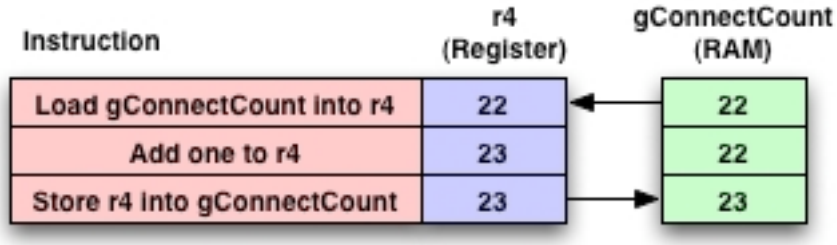
- Implementierung der Vektorzeit
  - Sender k packt Zeitstempel-Vektor in Nachrichten
  - Empfänger vergleicht Stempel-Vektor V mit lokalem Vektor L
  - accept if $(V_k = L_k + 1)$ and $(V_i \leq L_i \; \forall \; i \neq k)$



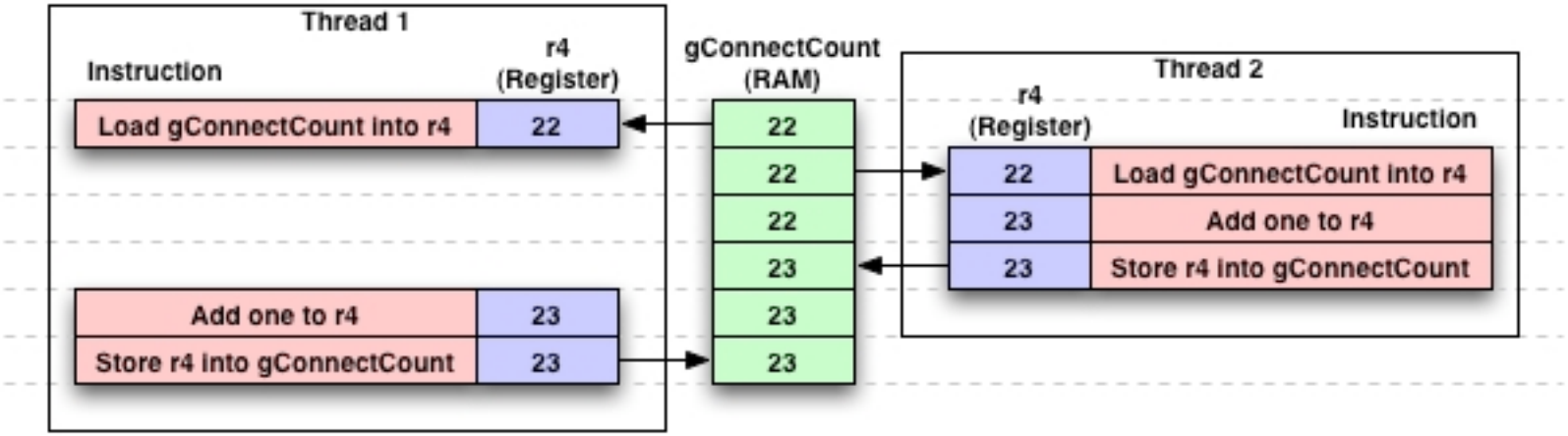




2.2.3 Koordination



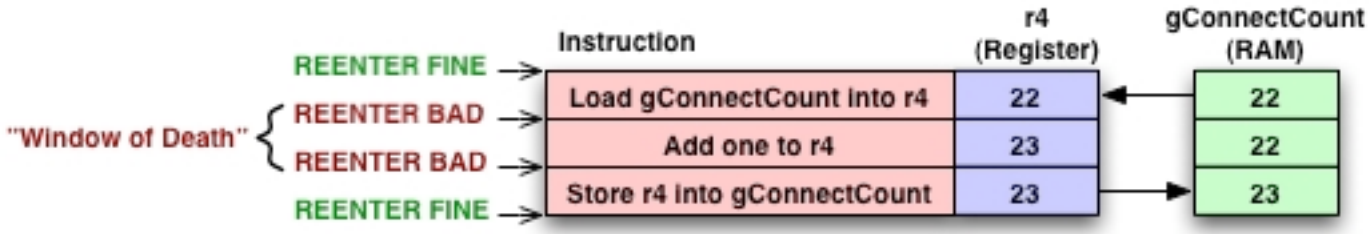
- Nebenläufigkeit der Prozesse
  - in einem Rechner durch Interrupts
  - im Parallelrechner mit Shared Memory
  - im Verteilten System
- Mehrere Prozesse wollen eine Ressource benutzen
  - Variablen (bool, Zähler), Objekte, Geräte
  - Bsp.: Kontostand, Drucker, Verwaltungsvariablen Ringpuffer



- Zugangskontrolle nötig
  - Schutzvariablen?
  - auch der Zugriff auf Schutzwariablen kann überlappen
  - Operationssequenz: Lesen-Testen-Setzen-Schreiben
  - Testen - *Testen - Setzen* – Setzen

2.2.3.1 Mutual Exclusion

- Lock schützt kritisch Region
  - Betriebssytemkonzept
  - 'Schutzbit' kontrolliert Eingang
  - Testen - Setzen - (Betreten - Arbeiten - Verlassen) – Rücksetzen

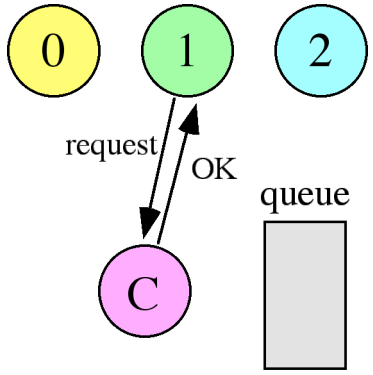


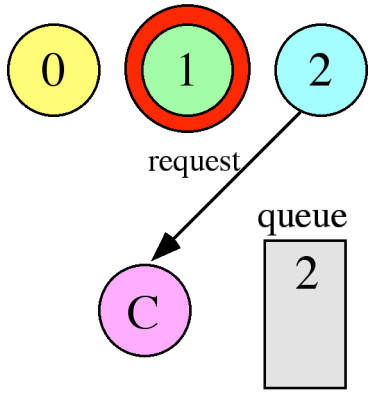
  - unteilbare Operationen: Read-Modify-Write
  - 68000: TAS (Test and Set), IA: XCHG
- PPC: lwarx+stwrx
  - Load Word And Reserve indeX
  - STore Word Conditional indeX

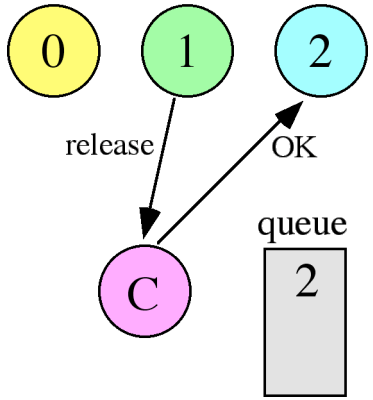
```
retry: lwarx  r4,0,r3   //integer lesen und reservieren
       addi   r4,r4,1   //increment
       stwcx. R4,0,r3   //Versuch, integer zurückzuschreiben
       bne-   retry     //Falls fehlgeschlagen – nochmal versuchen
```

  - http://www-128.ibm.com/developerworks/library/pa-atom/

- Singleton pattern: Objektinstanziierung als Lock
  - höchstens eine Instanz existiert
- Verallgemeinerung: Semaphor [Dijkstra, 1970]
  - Passeeren (`m--`) und Vrijgeven (`m++`)
  - 'Zählende' Locks (`if (m<=0) wait;`)
  - Locks werden auch binäre Semaphore oder Mutex genannt
- Allgemeines Verteiltes System
- Zentraler Verwalter
  - Request vom Klienten an den Verwalter
  - Permission wenn frei
  - Verzögern oder Ablehung wenn 'Besetzt'
  - Warteschlange
- Bewertung
  - einfach zu implementieren
  - fair
  - zuverlässiges Protokoll vorausgesetzt
  - 'single point of failure'



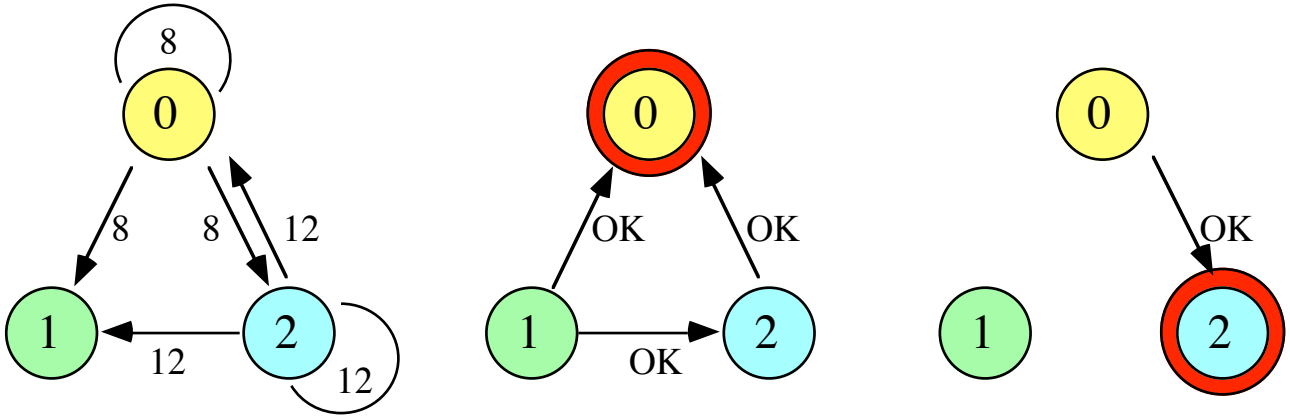




- Verteilter Algorithmus [Ricart, Agrawal 1981]
  - Request an alle: CR, Prozess#, Lamport-Zeit

```
if (requestin)
    if (inCritRgn == request->Rgn) queue(request);
    else if (ownrequest_pending)
            if (lowesttime(allRequests)->process == self)
            {  queue(request); sendOK(self); }
            else sendOK(request->process);
         else sendOK(request->process);
if (OKIn) {
inCritRgn = OKIn.critRgn; DoCritRgn; inCritRgn = NULL;}
sendOK(dequeue(request)->process);
```

- n points-of-failure
  - immer antworten
  - timeouts



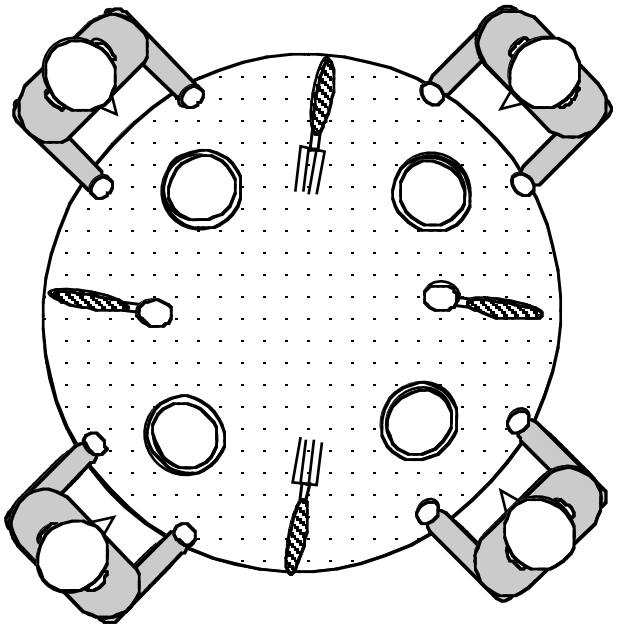
- Token Ring
  - logische Ringstruktur
  - Token = Erlaubnis, kritische Region einmal zu betreten
  - an logischen Nachfolger weiterreichen
  - Bestätigung über Empfang
  - keine Bestätigung -> nächsten Nachfolger
  - komplettes Topologiewissen nötig
- Vergleich

Algorithmus	Nachrichten/Vorgang	Verzögerung	Probleme
Zentral	3	2	Koordinator abgestürzt
Verteilt	2(n-1)	2(n-1)	irgendeiner abgestürzt
Token passing	1 - ∞	0 bis n-1	Tokenverlust Prozess abgestürzt

=> Verteiltes Verfahren ist schlecht

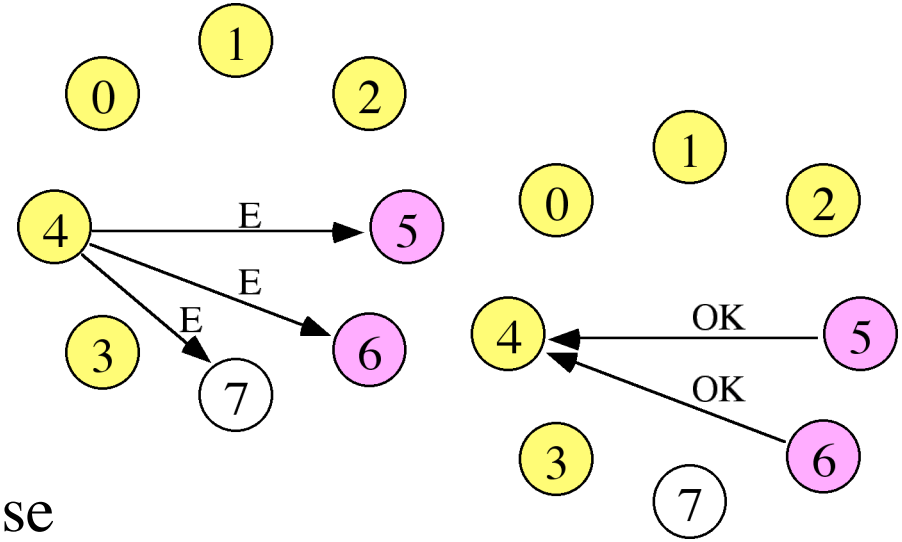
2.2.3.2 Deadlocks

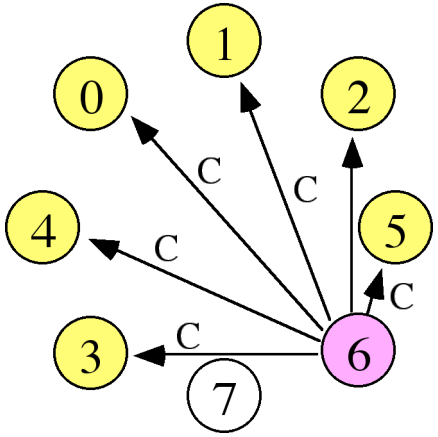
- Mutex besetzt: warten
  - dining Philosophers
  - Abhängigkeiten und Warten erzeugen Wartegraph
  - Zyklen im Wartegraph => Verklemmung
- Entdecken und Lösen
  - Prozess beenden
  - Transaktion abbrechen
  - zentral oder verteilt
  - Graph aufbauen und Zyklen suchen
- Verhindern
  - nur eine Ressource pro Prozess
  - release then lock
  - Ressourcenordnung
  - Zeitstempel: junge und alte Transaktionen
- Vermeiden (Wissen?)



2.2.3.3 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



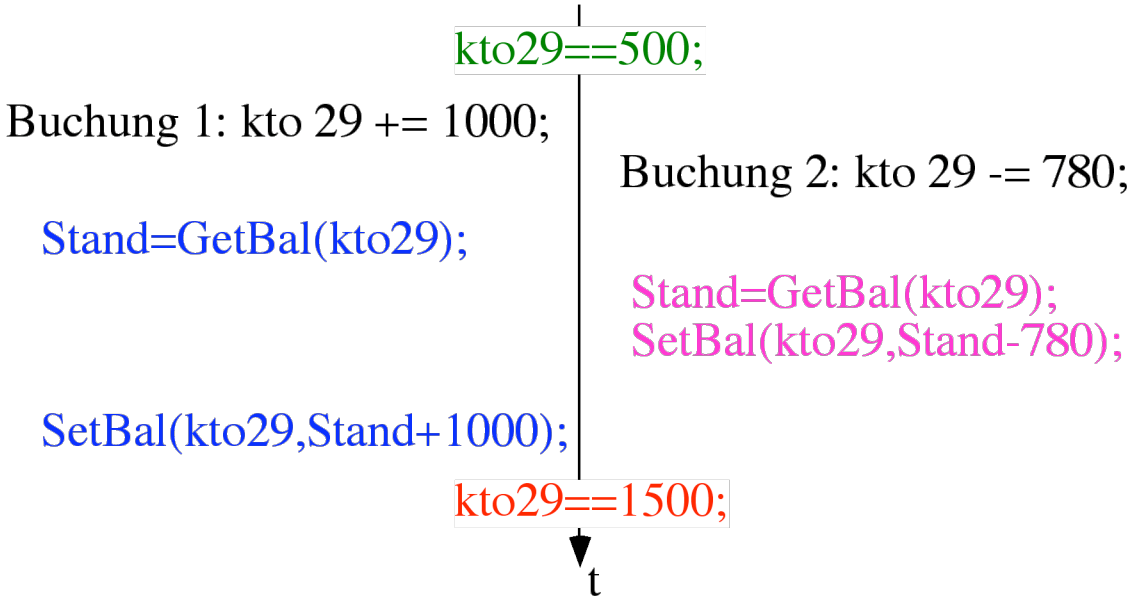


2.2.4 Transaktionen

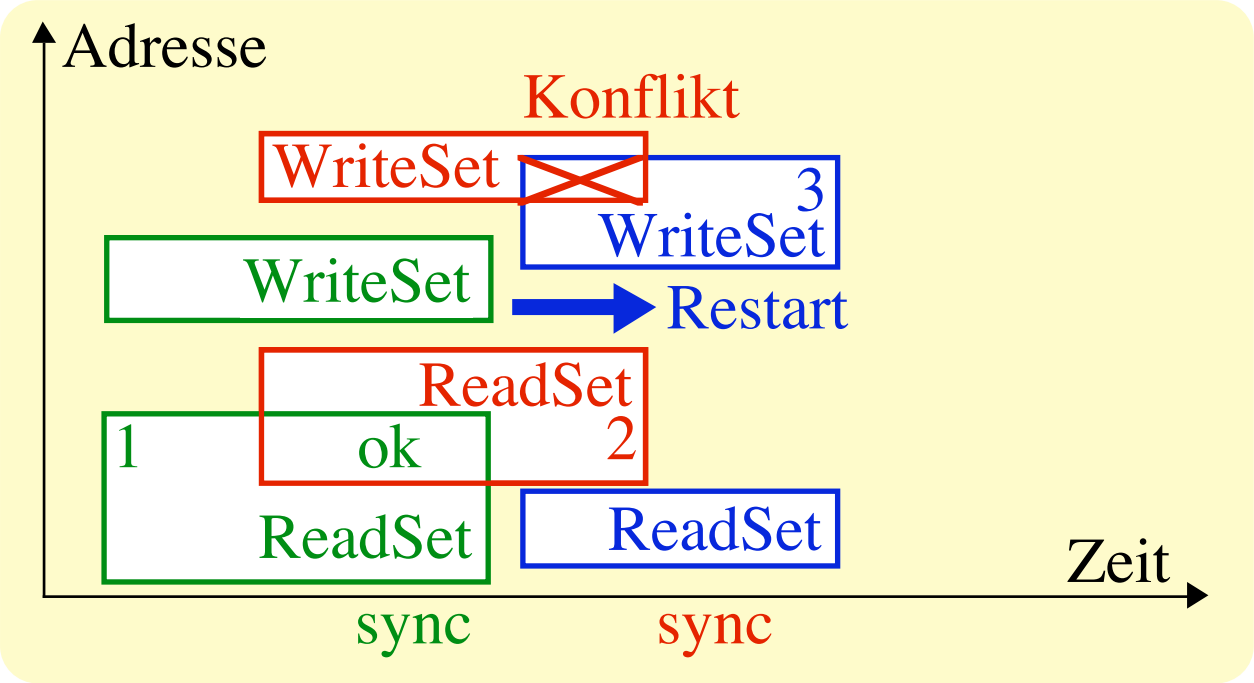
- Verteilte Dateisysteme, Datenbankoperationen
  - Lesen + Schreiben = Update
  - Überweisung = Abheben(kontoX) + Einzahlen(kontoY)

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

- Probleme bei Parallelität



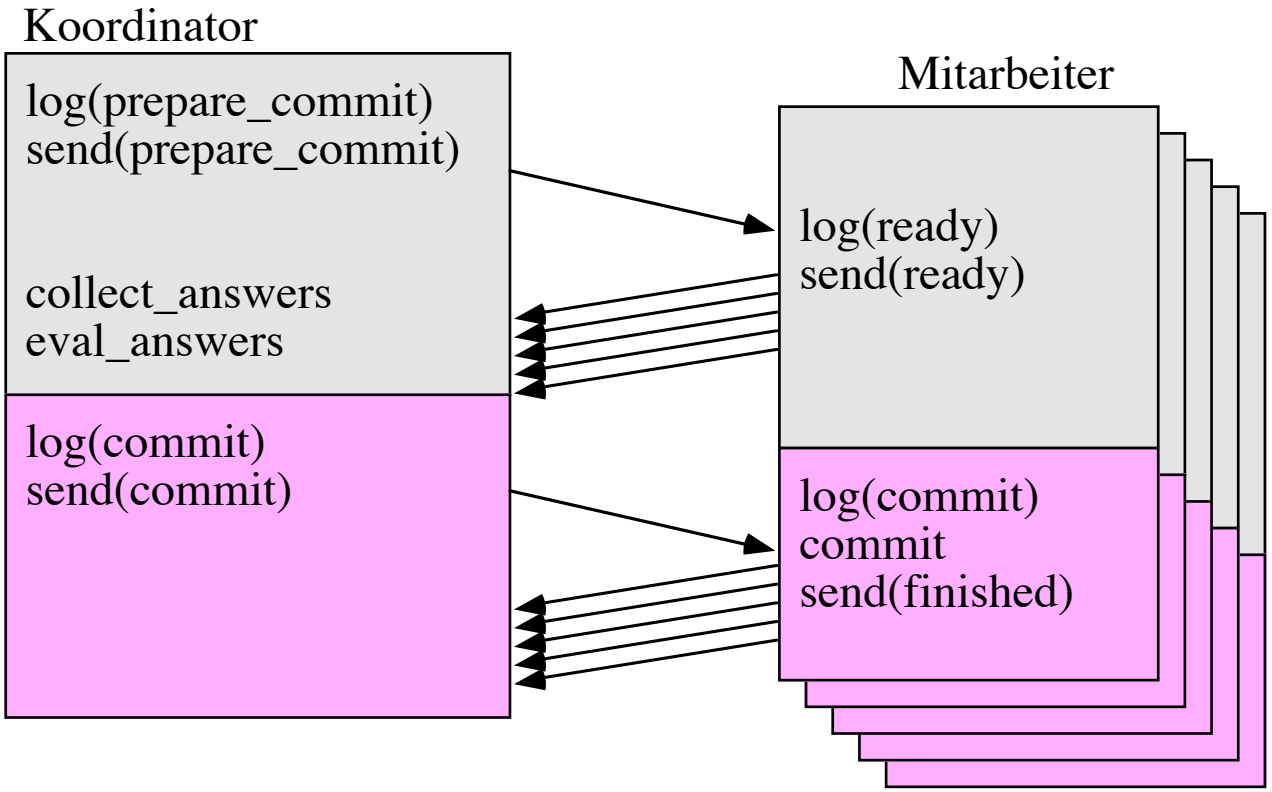
- Atomare Operationen
  - Transaktion
  - ACID
- Atomicity
  - entweder alle Teile oder keiner ausgeführt
  - commit oder abort
- Consistency
  - Zustandsüberführung konsistent -> konsistent
- Isolation
  - Serialisierbarkeit
  - erlaubt Nebenläufigkeit
- Durability
  - Persitenz des Resultates => Speicherung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

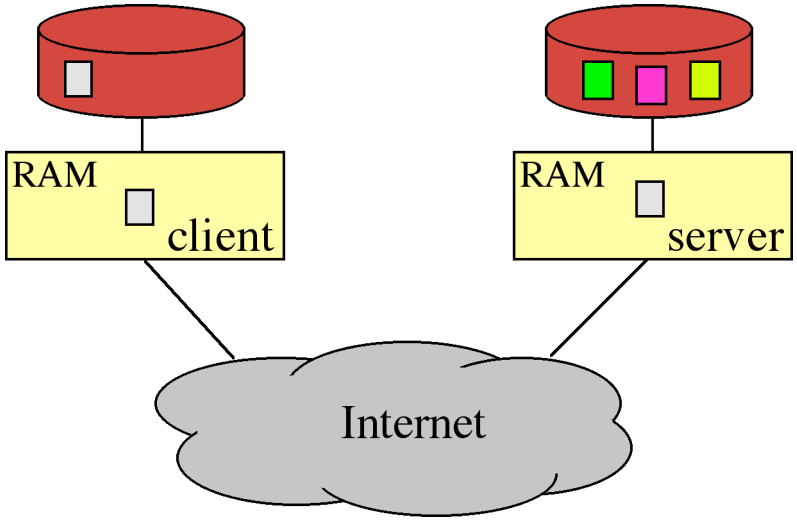


2.2.5 Effizienz beim verteilten Zugriff

- 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
  - besonders häufig in (verteilten) Dateisystemen
- Replikation
  - Verfügbarkeit
  - Fehlertoleranz
- Problembereiche: Konsistenz und Transparenz

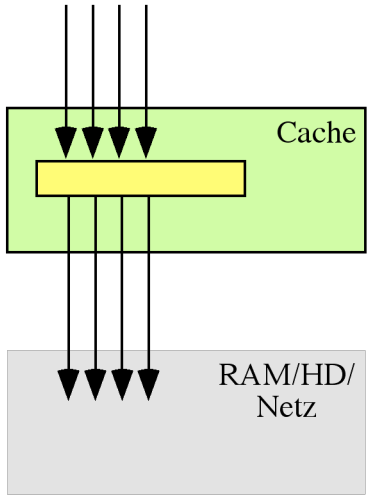
2.2.5.1 Caching

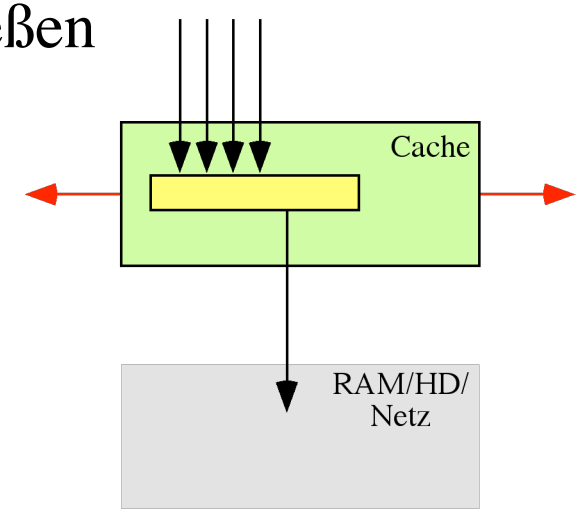
- Objekte 'in der Nähe' halten
  - Speicherzellen, Seiten, Sektoren, Tracks, Files, …
  - Cache-RAM, RAM, Controller, …
  - Zugriffskosten reduzieren
- Sonderfall verteiltes Dateisystem
  - Files
  - Server: RAM
  - Klient: Festplatte, RAM
  - Prozess, OS, Caching-Prozess



- Aufgaben
  - Einheit des Cachings: Blöcke oder Files?
  - Cache kleiner als Speicher: was soll 'gecacht' werden?
  - Prefetch?
- Last 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
  - evtl. mit Verzögerung: Löschen häufig nach Close
- Write-invalidate: andere caches konsistent halten
- Zentrale Kontrolle
  - alle Zugriffe anmelden: read, write, update
  - Remove-from-Cache bei konkurrierenden Zugriff
  - unsolicited-message



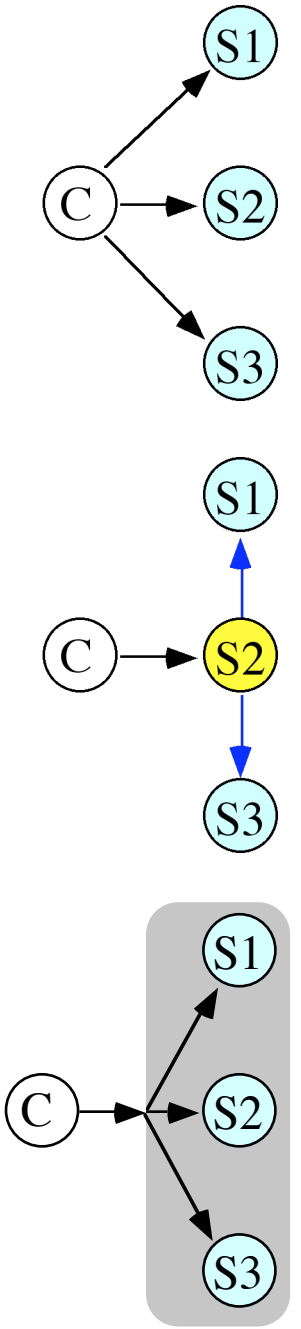


2.2.5.2 Replikation

- z.B. Mirror-Server
  - ftp, WWW, …
  - Usenet News
- Client/Server, Peer-to-peer
  - Nachrichtenaustausch
  - Lesen Regelfall
  - Schreiben Ausnahme
  - besondere Vorkehrungen beim Schreiben
- Verteilter gemeinsamer Speicher (DSM)
  - Speicherobjekte: Variablen
  - Seiten
- Replikationsmanagement
  - Konsistenzmodell
  - Verhältnis Lesen-Schreiben
  - Aufwand

- Asynchrones Replikationsmanagement
  - Lesen immer lokal
  - Schreiben lokal
  - Update anderer Kopien periodisch
  - inkohärent bis zur Synchronisation
  - entkoppelt und einfach
  - Bsp. News: Antwort vor der Frage
- Kausal geordnetes Replikationsmanagement
  - kausale Ordnung für abhängige Veränderungen
  - zeitliche Kohärenz
  - nur von einem benutzte Replikate inkohärent
- Synchrones Replikationsmanagement
  - alle Replikate atomar geändert
  - volle Konsistenz
  - hoher Aufwand

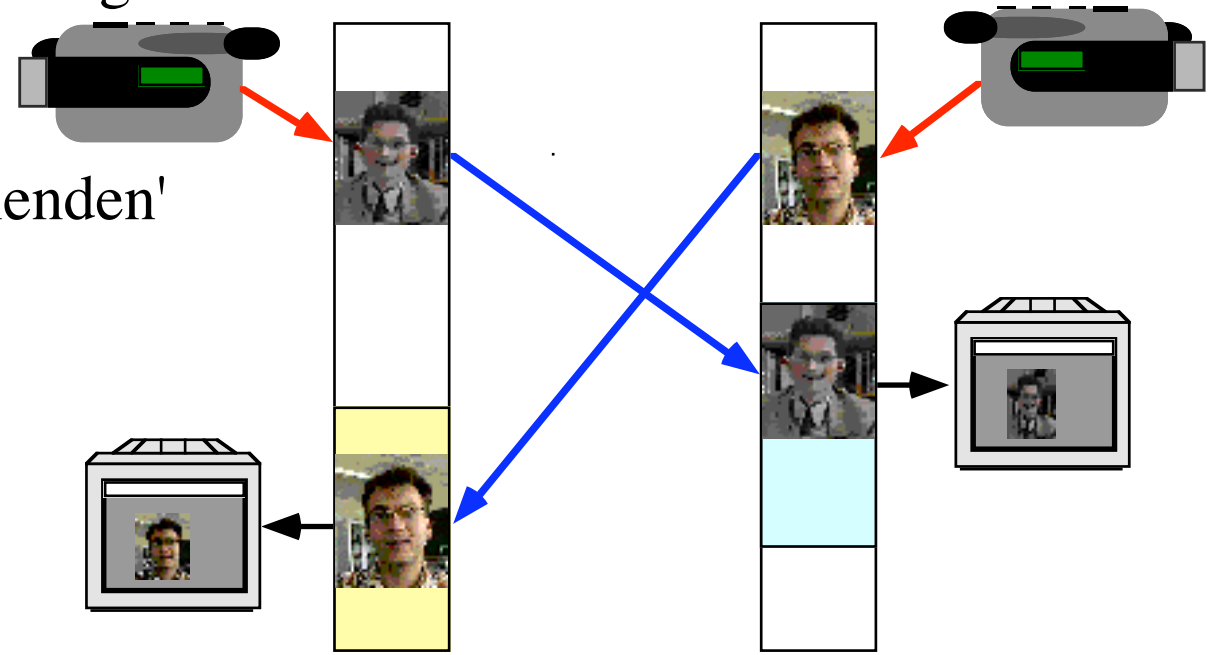
- Erzeugung von Replikaten
  - nicht alles muß repliziert werden
  - Auswirkungen auf Konsistenzerhaltung
- Explizite Replikation
  - klientengesteuert: Anforderung an mehrere Server
  - Zugriff auf ein Replikat
  - Konsistenzmanagement beim Klienten
- Lazy Replikation
  - Objekterzeugung durch Server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



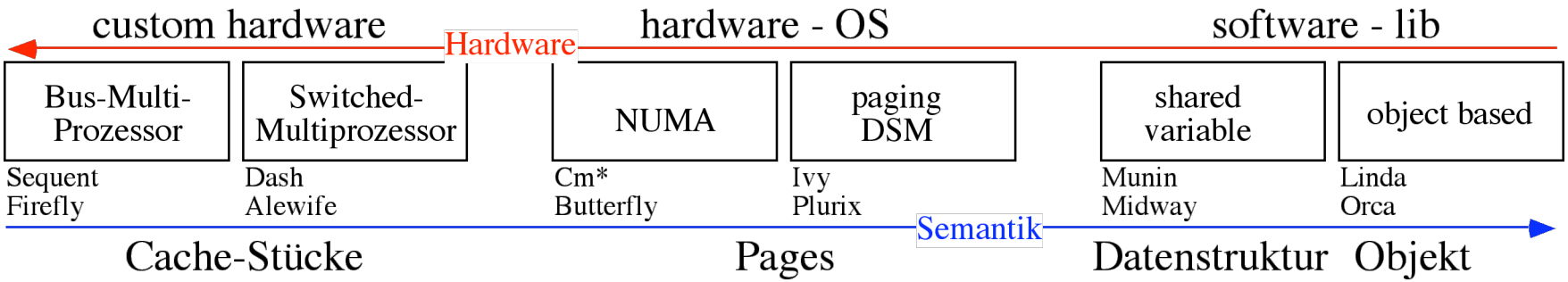
- Primärkopien
  - 'Orginal' im Primärserver
  - Replikate auf Sekundärserver
  - Lesen überall, Schreiben nur auf Orginal
  - Update der Replikate in der Servergruppe
  - Primärserver-Ersatz siehe Election
- Abstimmung
  - N Server
  - N/2+1 Replikate beschreiben mit Zeitstempel
  - N/2+1 Replikate 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
  - News
- Peermodell als Variante

3. Distributed Shared Memory

- Verteilungsoptionen
  - handkodiert
  - RPC
  - Datenbank
- Verteilter gemeinsamer Speicher
  - sehr einfach zu Programmieren
  - allgemeinste Abstraktion des Netzwerkes
  - Variablen, Code, Objekte, … => Adressraum
- Regelung des konkurrierenden Zugriffs siehe oben
- Beispiel Videokonferenz
  - entfernte Videobilder in den eigenen Speicher 'einblenden'

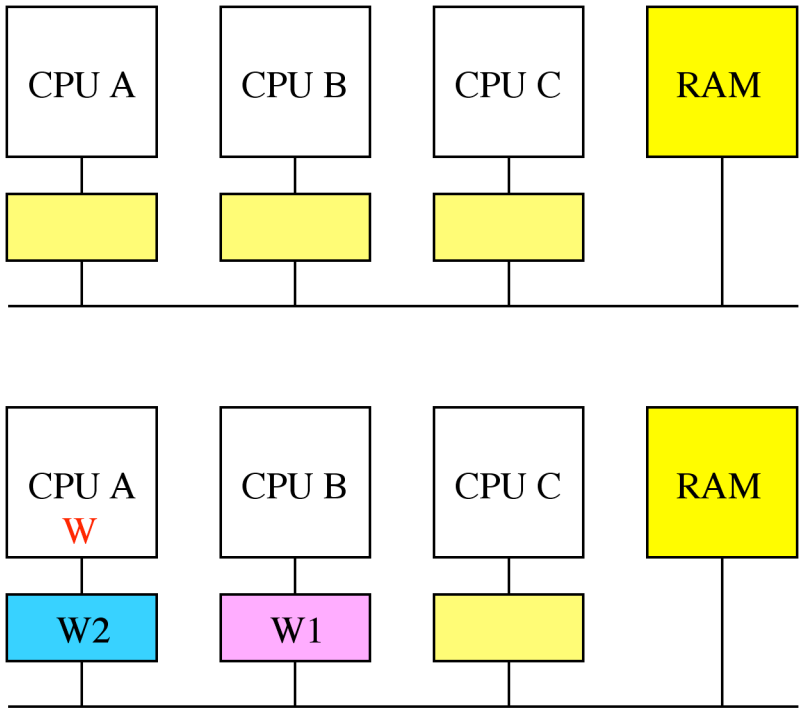


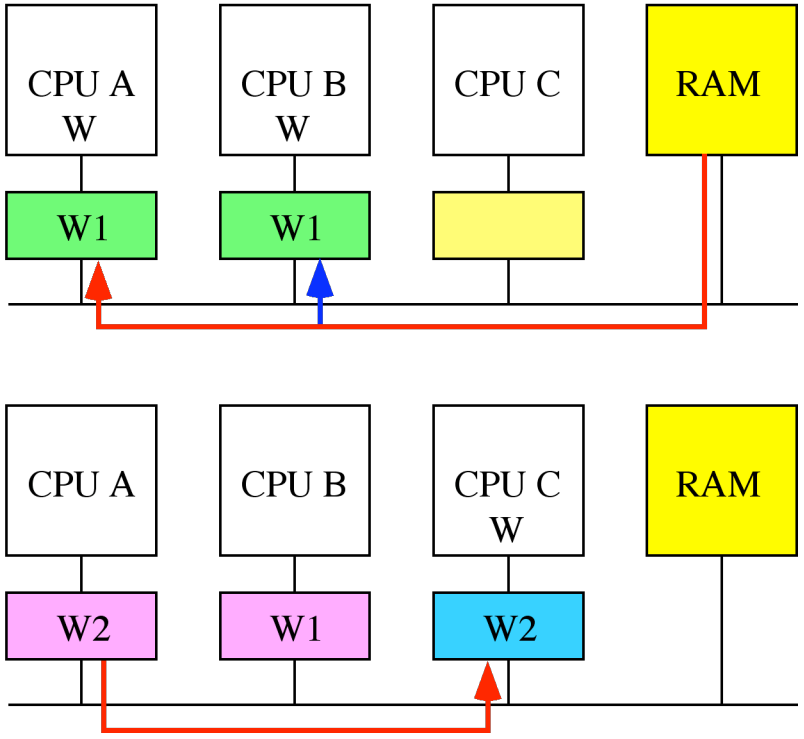
- DSM-Architekturen



- Blockbasiert
  - Maschinenbefehle greifen auf Speicher zu
  - physisch gemeinsamer Speicher, kontrollierter Zugriff
  - Caching, cache-consitency
- Seitenbasiert
  - Memory Management Unit (MMU) übersetzt Adresse
  - lokale Adresse: reale Adresse im RAM, evtl. page-fault + Paging
  - remote-Adresse: page-fault + Seite holen und einblenden
  - NUMA: Non Uniform Memory Access
- Spracherweiterungen
  - gemeinsame Variablen bzw. Objekte
  - keine Programmiertransparenz
  - mehr Wissen über gemeinsame Objekte

- Beispiel Caching Multiprocessor
  - bus snooping
  - write-xx Strategie
  - clean, invalid, dirty
  - Protokoll in einem Buszyklus





3.1 Lesen & Schreiben im DSM

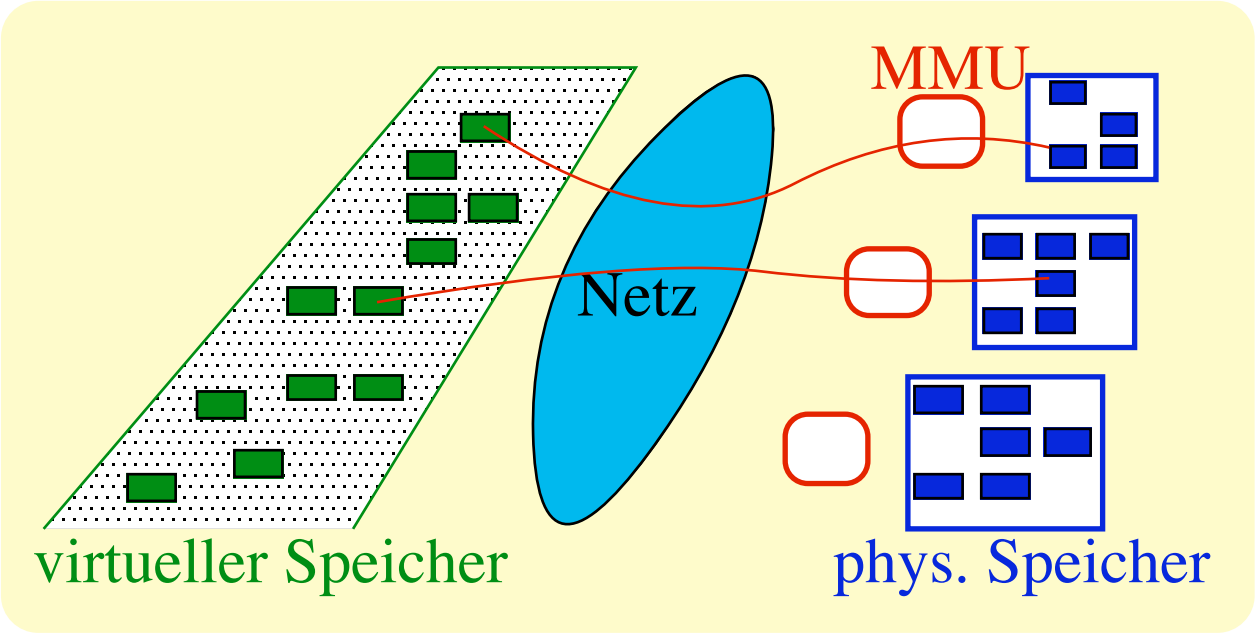
- Problem ist die (relativ) hohe Verzögerung im Netz
- Keine Kopien im Netz => keine Replikation
  - Eigentümer einer Seite kann sofort zugreifen
  - Kommunikationspartner brauchen länger
  - ortsfeste oder migrierende Seiten
- Kopien im Netz vorhanden => Replikation
  - lesen einer Seite ist sofort möglich
  - entweder auf alle Kopien schreiben (update)
  - oder alle Kopien löschen und nur auf Original schreiben (invalidate)

- Matrixdarstellung:

	ohne Replikation	mit Replikation
Ortsfeste Seiten, R/W-Operation transportieren	Verzögerung für alle, außer für Eigentümer	sofort lesen, überall schreiben, **"write update"**
Migrierende Speicherteile, lokale Operation	langsam lesen, langsam schreiben, Seitenflattern	aus Cache lesen, Original schreiben, **"write invalidate"**

- Nach "write invalidate" Seiten von anderen Lesern erneut anfordern
- Speicher-Dienstegüte?
  - Zugriffshäufigkeit
  - Zugriffstyp: read/write
  - Zugriffsmuster: once, n-read 1 write, …
  - streaming

- Abbildung des logischen Adressraumes auf Stationen im Cluster



  - hardwaremässig mithilfe der MMU und Paging
  - mithilfe einer Segmentierung (HW oder SW)
  - evtl. Lese-& Schreibbefehle über das Netz transportieren

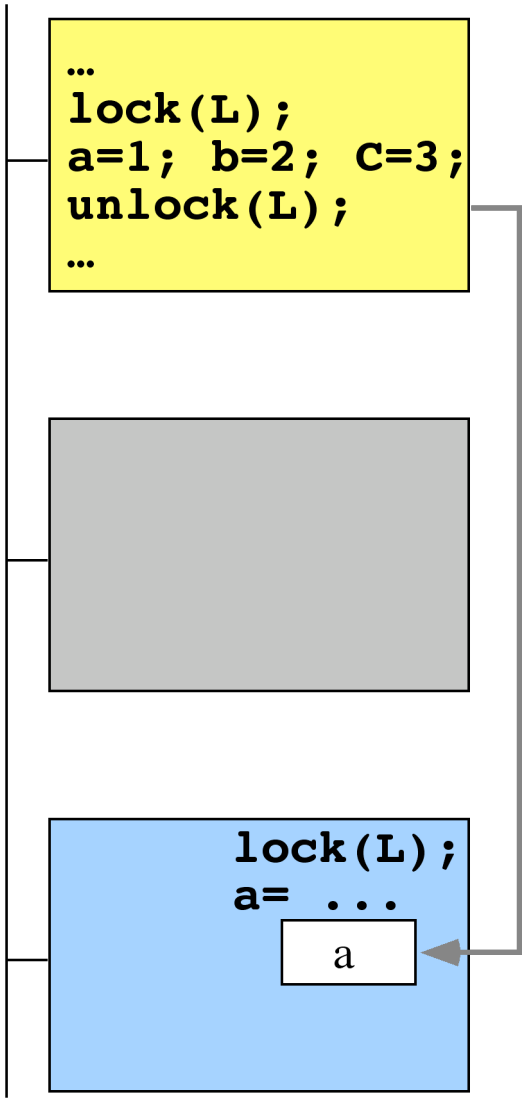
- Vorteil einfaches Programmiermodell
  - "single-system" Perspektive, keine Serialisierung

- Grundlagen Virtual Memory siehe Vorlesung Rechnerarchitektur

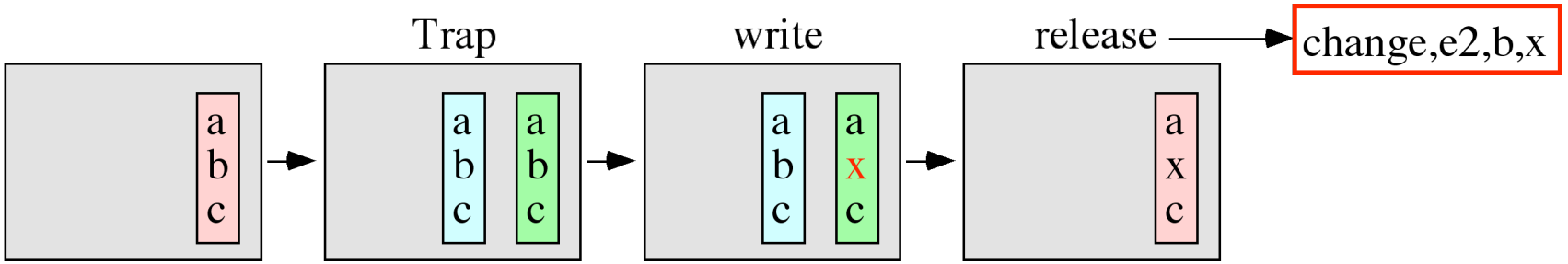
3.2 Shared Variables

- Verteilung kleinerer Objekte
  - weniger 'false sharing'
  - Variablen und Datenstrukturen
  - 'verteilte Datenbank'
  - Replikation
- Munin [Bennerr, Carter, 1990-1993]
  - verteilte Objekte auf Seiten
  - MMU kontolliert Zugriff
  - mehrere Konsistenzmodelle: Parametrisierung durch Programmierer
  - update, invalidate, Zeitpunkt
  - Anzahl Replikate, Eigentümer, Nutzung, …
- Spezialcompiler
  - Schlüsselwort 'shared' bei der Variablendeklaration
  - evtl. Benutzungsinformation
  - vorgefertigte Modelle: read-only, migratory, write-shared, conventional
  - default: eine Seite pro Variable
  - Programmierer kann Variablen zusammenfassen

- Shared Variablen
  - read MMU-kontrolliert
  - write: nur in kritischen Regionen
  - synchronization: Zugangsprozeduren
- Kritische Regionen
  - exklusiver Zugang mit lock()
  - unlock() stößt Update der Replikate an
- Read-only Variable
  - Page-fault
  - Munin sucht im Verzeichnis
  - fordert vom Besitzer die Seite an
  - MMU verhindert Schreibzugriff
- Conventional
  - eine Schreibkopie, viele Lesereplikate, invalidate
- Migratory Variable
  - Benutzung in kritischen Regionen
  - nur eine Kopie, keine Replikation
  - page-fault - Seite holen - alte Kopie löschen - benutzen



- Write-shared Variable
  - Bsp. Arrays: Elemente von verschiedenen Prozessen zugreifbar
  - Trap beim Write: twin anlegen
  - Release: vgl. write-copy und twin
  - Differenz verteilen
  - Empfänger vergleichen write-copy, twin, Differenzliste
  - run-time Fehler bei Konflikten



- Barrieren sorgen für Synchronisation

P1

```
wait_at_barrier(b);
for (i=0;i<n;i+=2)
    a[i]+=x;
wait_at_barrier(b);
```

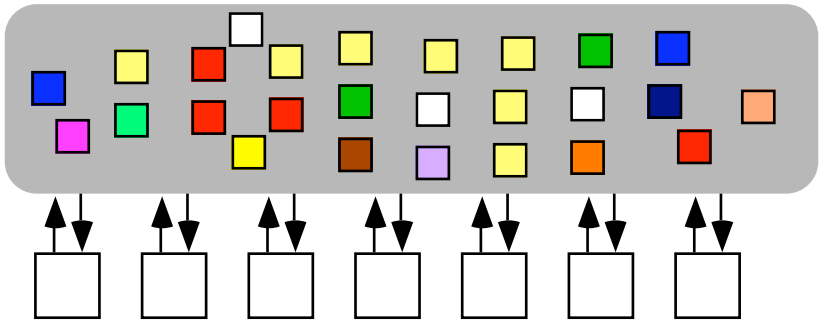
P2

```
wait_at_barrier(b);
for (i=1;i<n;i+=2)
    a[i]+=y;
wait_at_barrier(b);
```

3.3 Objektbasierter Verteilter Speicher

- Objekte verstecken Zugang zu Feldern
  - Interna bleiben Implementierungssache
  - Linda, Orca, Amber, Emerald, Cool
- Linda [Gelernter, 1985]
  - kleine Bibliothek
  - C, FORTRAN, Modula, …
  - JavaSpaces
  - Präprozessor
- Tupel
  - Gruppe von Elementen (Felder)
  - Basistypen
  - C: integer, long integer, float, …, arrays, structures
  - kein Tupel im Tupel

  ("abc",2,5)
  ("konrad","froitzheim",45054)
- Tupel-Space
  - global im verteilten System
  - insert und remove



- out("konrad","froitzheim",m);
  - schreibt Tupel in den Tupel-Space
  - Konstante, Variablen, expressions
  - Tupel werden nicht überschrieben => mehrere Kopien
- in("konrad","froitzheim",?i);
  - inhaltsadressiert
  - i erhält Wert des dritten Tupelelementes
  - Tupel wird entfernt
  - matching und entfernen atomar
  - blockiert falls kein passendes Tupel da
- Bespiel: Semaphor
  - out("mein kleiner Sema"); gibt frei
  - in("mein kleiner Sema"); => lock
- read()
  - findet match
  - entfernt Tupel aber nicht

- Beispielproblem: Ray-tracing
  - Dispatcher

```
out("tasks",L [1],1);
out("tasks",L [2],2);
…;
out("tasks",L [n],n);
```

  - Bearbeiter

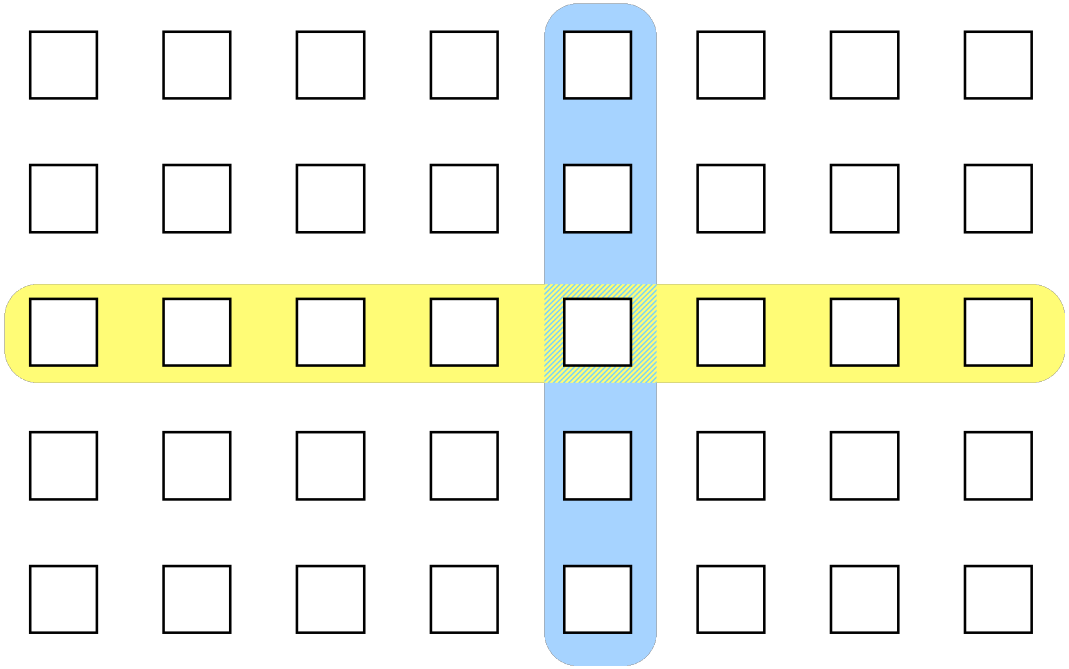
```
in("tasks",?Line,?nr);
Rechnen(Line);
out("fertig",Line,nr);
```

  - Anzeige

```
in("fertig",?disp,?nr);
VRAM[nr]=disp;
```

- Implementierung
  - assoziativer Speicher teuer
  - volle Suche …
  - Verteilung?
- Idee: Hierarchie
  - erstes Tupelelement teilt in Subspaces
  - Konvention: erstes Element String
  - Konstante im ersten Feld => subspace zur Übersetzungszeit bestimmen
  - subspaces => Verteilung
  - Hashing auf den anderen Feldern
- Multiprozessor
  - tuple subspace => hash-table
  - im globalen verteilten Speicher
  - lock subspace - enter/remove - unlock subspace

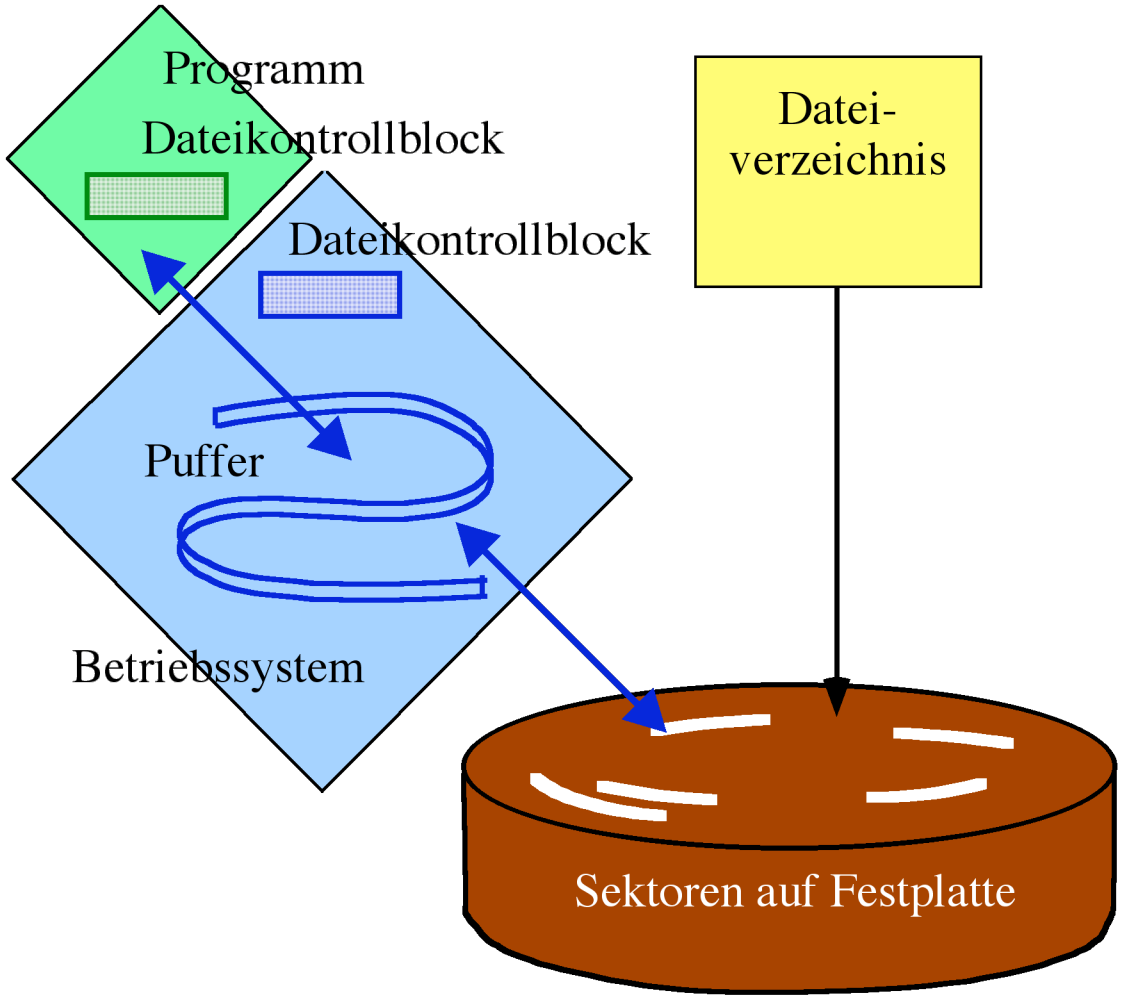
- Multicomputer mit Hochleistungsnetz
  - out() => broadcast, in() => lokale Suche
  - entfernen mit delete protokoll: 2-phase-commit
  - Probleme bei großen Systemen?
- Linda im LAN [Carriero]
  - out() lokal, in() mit Broadcast
  - keine Antwort: erneuter Broadcast
  - Treffer wird entfernt ohne besondere Probleme
  - in() blockiert evtl.
- Kombination [Krishnaswamy, 1993]
  - Prozessoren im Gitter
  - out() broadcast in der Zeile
  - in() broadcast in der Spalte
  - Kreuzung eindeutig



4. Verteiltes Dateisystem

- Vorwort Dateisysteme
- Historisch gesehen ein Magnetband
  - mit einem Schreib-/Lesekopf
  - Datenblöcke sequentiell aneinanderreihen
  - wahlfreie Zugriffe teuer bzw. langsam
  - Einfügeoperationen sehr teuer
  - beinahe beliebiges Größenwachstum
- Aus der Sicht der Programmiersprache
  - Von Dateimodulen exportierter Typ "File"
  - sequentiell Lesen und Schreiben
  - Öffnen und Schliessen, Zugriffsrechte
- Aus der Sicht des Betriebssystems
  - benanntes Objekt ("DistSys.DOC")
  - Dateikontrollblock mit Puffer im RAM
  - Assoziation einer Dateivariablen mit externer Datenregion
  - Abbildung:
    blockweise adressierte Objekte auf Platte
    => byteweise adressierte Objekte im Hauptspeicher

- Implementierung
  - Programm mit Zugriffsroutinen
  - Treiberroutinen im Betriebssystem
  - Dateivariable im Programm (file)
  - Dateikontrollblock (User & System)
  - Pufferbereich im Hauptspeicher
  - Dateiverzeichnis für Festplatte
  - Sektorzuordnung auf Festplatte
- Betriebssystemteil auswechseln
  -> virtuelles Dateisystem



- Datei
  - DOS, Windows, UNIX: Sequence of Byte
  - Macintosh: Forks, Inhaltskennzeichen
  - VM, VMS: satzweiser Zugriff, B-Baum-Struktur
- Verwaltungsinformation
  - Zugriffsrechte für Benutzergruppen: read, write, delete, execute, …
  - Creator, owner, …
  - evtl. komplexe Rechtestruktur, Rollenkonzept
  - Zeitstempel, …
  - Betriebssystemdaten
- File-Service
  - Dateien stehen den Programmen zur Verfügung
  - File-Server sind 'unsichtbar'
  - lokale Platte ähnlich File-Server
- Datei-Sharing-Semantik
  - upload/download
  - read/write/seek

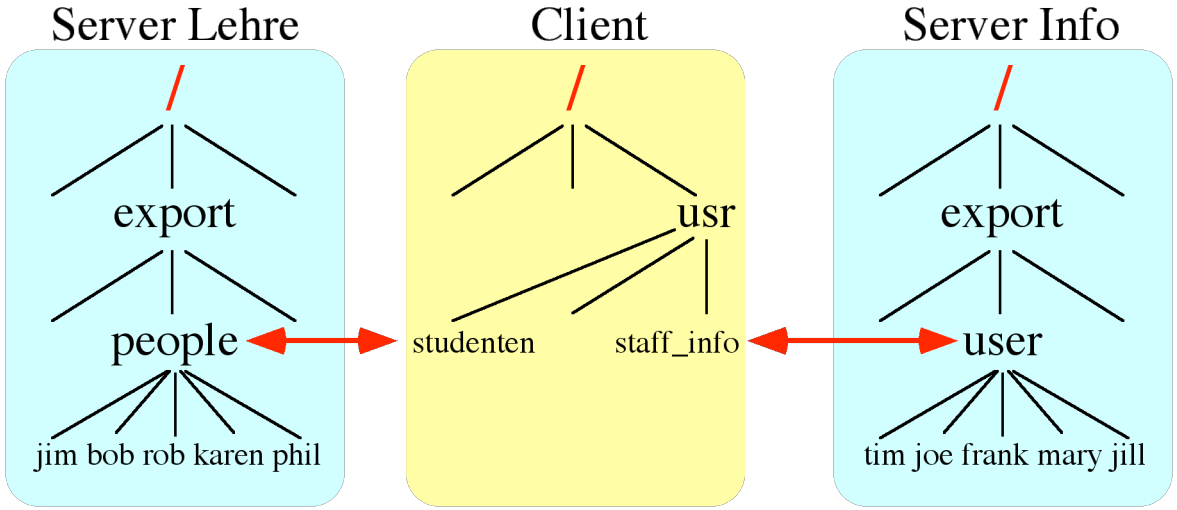
- Verzeichnis-Dienst
  - Struktur in der Dateimenge
  - hierarchisches System
  - Baum mit Wurzelverzeichnis
  - beliebiger Graph (!)
  - Verzeichnisse anlegen, löschen, wechseln, …
- Dateinamen
  - Typ im Namen als Extension (.PDF, …)
  - kompletter Dateiname: name+Zugangspfad
  - verteilt: Computer+Zugangspfad+Dateiname => URL
  - Aufgespannt durch Kombination mehrerer Namensräume
- Beispiel:

  //frodo.informatik.uni-ulm.de/test/Beispiel1.txt
  identifiziert Datei: /test/Beispiel1.txt auf frodo

UNIX-Semantik	jede Operation sofort für alle sichtbar
Session-Semantik	Änderungen erst nach close für andere sichtbar
Unveränderbarkeit	keine Updates …
Transaktionen	Änderungen entweder unsichtbar oder für alle sichtbar

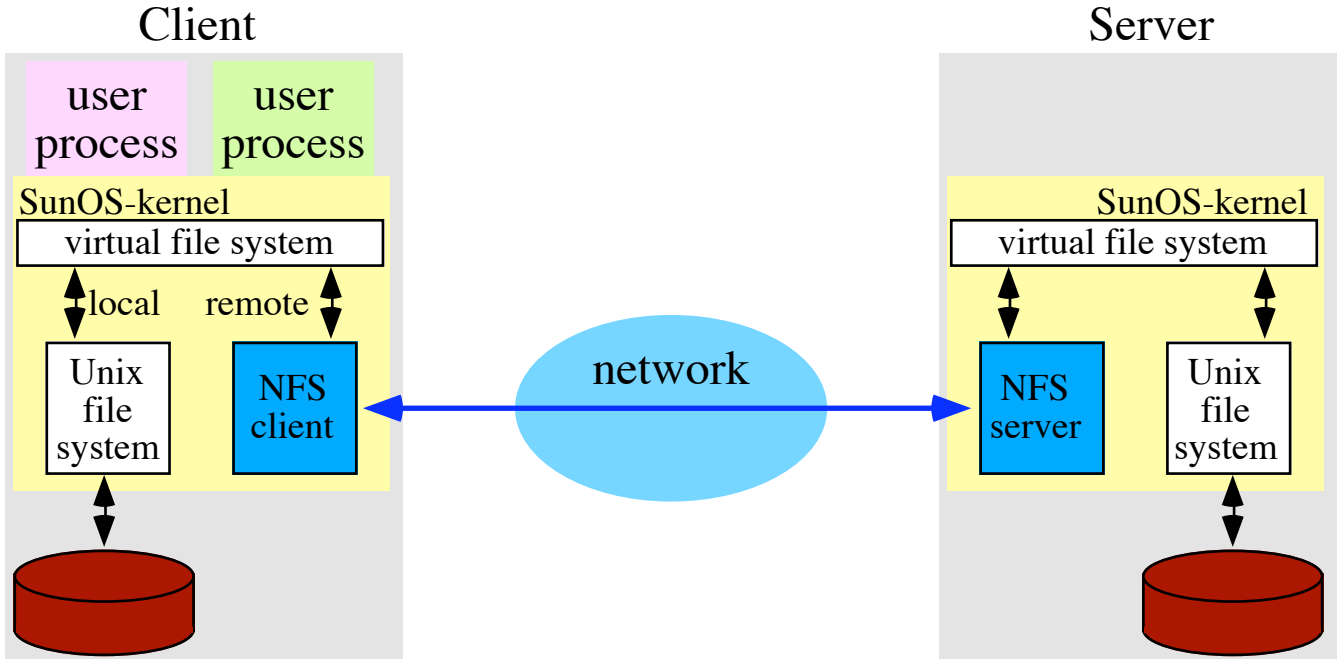
- Dynamische Zustände
  - Positionsinformation
  - Klienten und Sperren
- Zustandsorientierte Systeme
  - Server hat genaue Zugriffsinformation
  - kurze Pakete, insbes. ohne Positionsinformation
  - Prefetching möglich (Zugriffsmsuster)
- Zustandslose Server
  - jeder Request enthält komplette Zustandsinformation
  - Ausfallsicherheit Server *und* Klient
  - Locks trotzdem nötig
  - auf lokales Dateisystem abbilden

- AFS: Andrew File System [CMU, Morris, 1986]
- NFS: Network File System [SUN, 1989]
  - transparente Benutzung
  - symmetrisch: jeder Rechner als Klient und Server
  - Server- und Klienten-Komponenten im Kernel



  - kein Netzwerk-einheitlicher Namesraum
- Heterogenes System
  - UNIX, Mach
  - Server: VMS, Netware, …
  - Klienten: MS-DOS, Windows, …

- Architektur
  - SUN-RPCs
  - well-known port, port-mapper



- VFS: Virtual File System
  - file handles: (file system ID, i-node number, i-node-generation)
  - i-node-number wiederverwenden => i-node-generation++
  - v-node: (locallremote, i-nodel remotefh)

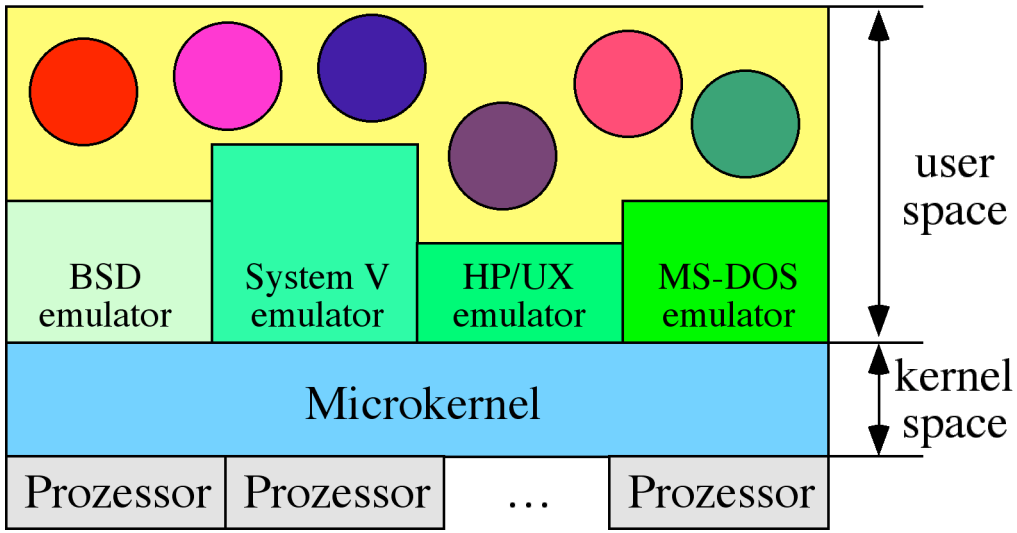
- Client caching: read
  - Zeitstempel
  - Vergleich eines 'ge-cachten' Blockes
  - beim Lesen 3 Sekunden später
  - Nachfrage beim Server: getattr
- Client caching: write
  - Asynchrones 'flush' nach dem Schreiben (dirty-bit)
  - close-file oder sync
  - bio-daemon (block input output)
- Server caching
  - write-through anstelle von delayed-write
  - ab V3: Schreiben erst nach commit für Datei
- Zustandsloser Server
  - Zustand im NFS-Client
  - Zustandsinformation in jedem Kommando
  - siehe cookie
  - Authentication in jedem RPC (Kerberos)

- RPC
  - im LAN über UDP
  - im Internet Multiplex über eine TCP-Verbindung
- Befehle
  - **`lookup(dirfh,name) -> fh, attr`**
  - **`read(fh,offest,count), write(fh,offest,count,data)`**
  - **`getattr(…), setattr(…)`**
  - **`create(dirfh,name,attr) -> newfh,attr`**
  - **`remove(dirfh,name), rename(…)`**
  - **`link(newdirfh,newname,dirfh,name)`**
  - **`symlink(newdirfh,newname,string)`**
  - **`mkdir(…), rmdir(…)`**
  - **`readdir(dirfh,cookie,count)`**
- Mount-Service
  - läuft auf jedem Server
  - /etc/exports enthält Liste mit mountbaren Filesystemen
  - modifiziertes mount-Kommando
  - RPC-basiertes mount Protokoll
  - hard mounted und soft mounted
  - /etc/rc im Klienten: mount beim Systemstart

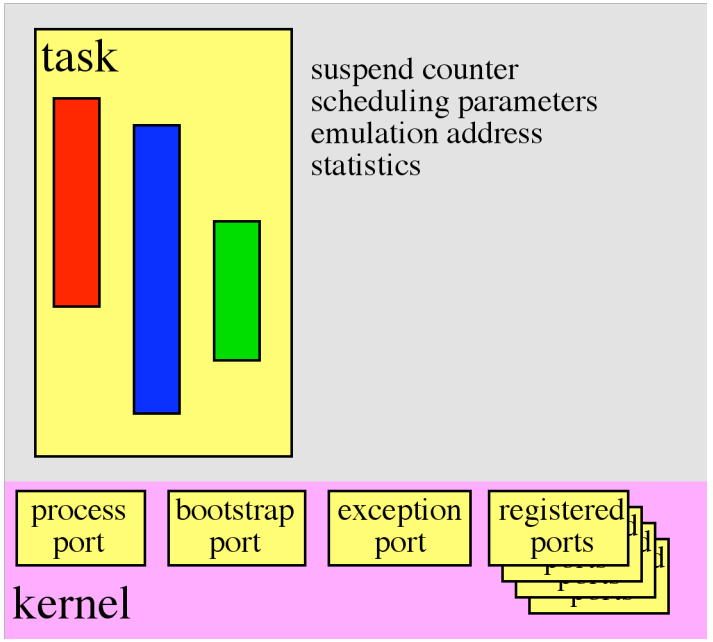
- Automount
  - mount bei Bedarf
  - beim Referenzieren eines 'leeren' mount-points
  - lokaler NFS-Server
  - sucht Filesystem: probe zu (mehreren) Servern
  - trägt symbolischen Link lokal ein
  - unmount nach einigen Minuten
  - Replikation von read-only Dateisystemen (/usr/lib)
- Transparenz
  - ~ Zugriff: VFS wie lokales Dateisystem;
    aber: UNIX-Semantik nicht komplett
  - ✓ Ort: remote file system in lokales Verzeichnis einbinden
  - ✓ Fehler: Server zustandslos
  - ? Leistung: viel caching
  - ~ Migration: teilweise durch mount und Automount
  - - Replikation
  - - Concurrency
  - ~ Skalierung: <50?

5. Microkernel: Mach

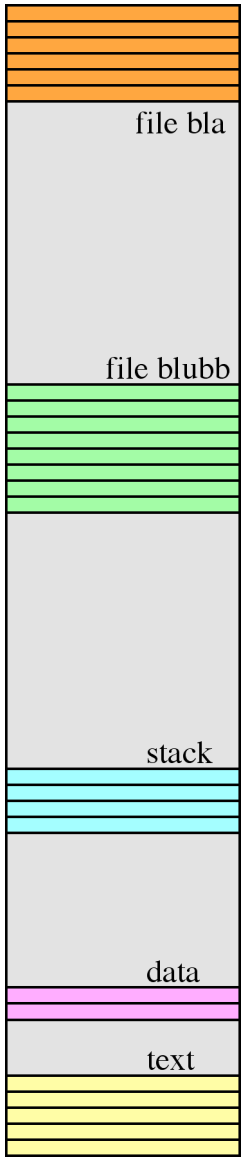
- Betriebssystemprojekt von Carnegie-Mellon (CMU)
  - 1985-1990-1994
  - UNIX-Emulation auf User-level
  - Träger für verschiedene Betriebssysteme
  - BSD, System V, AIX, OS/2, MS-DOS, VMS
  - OSF/1, Next,
  - MkLinux, Mac OS X (Darwin)
  - auf verschiedenen Hardwarearchitekturen
  - Shared Memeory Multiprozessor
- Microkernel: Mach 3.0
  - möglichst viele Aufgaben auf User-level
  - Basis zur Betriebssystementwicklung
  - Prozesse, Speicher, Kommunikation, I/O



- Prozesse: Tasks
  - Adressraum
  - Stacks etc.
  - Ressourcen: Kommunikationskanäle, …
  - Threads im Task werden ausgeführt
  - Erzeugen: Blueprint-Task kopieren
  - leerer oder BluePrint-Adressraum
  - erster Thread von außen erzeugt
  - kernel-port, exception-port
- Threads
  - ausführbare Einheit: PC, Register
  - vom Kern verwaltet
  - thread port
  - fork, exit, join, detach, yield, …; lock, trylock, unlock
- Tasks kontrollieren Thread-Ausführung
  - Threads einer Prozessormenge zuweisen
  - dynamische Lastverteilung
  - Scheduler im Kern: run queues

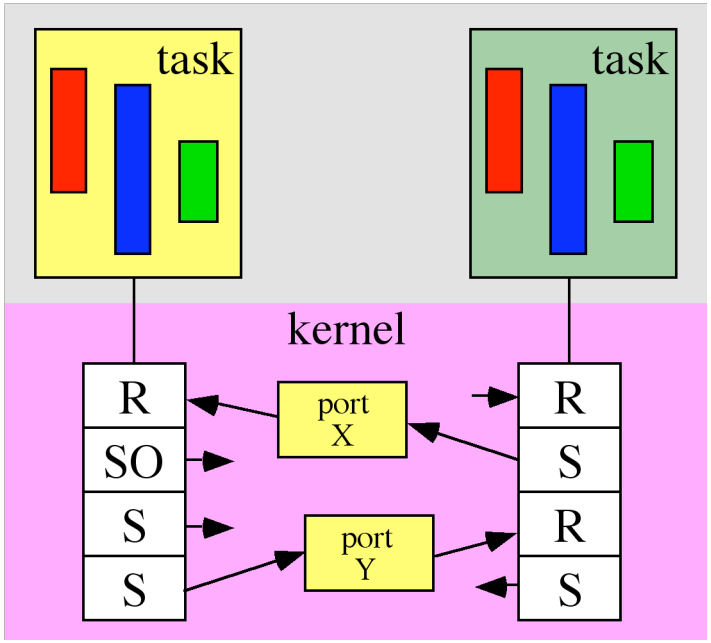


- Memory Management
  - maschinenabhängig: MMU-Interface pmap
  - m-unabhängig: page-faults, address maps, auswechseln
  - User-Prozess external pager verwaltet Seiten auf HD
- Linearer Adressraum
  - $0..2^{31}-1$ für user
  - Regions zur Effizenzsteigerung
  - Allozierung: System sucht passenden Bereich
- Speicherobjekte
  - Datenstrukturen
  - eingeblendet in Adressraum (map)
  - eine oder mehrere Seiten des virtuellen Speichers
- Dateien sind ebenfalls Speicherobjekte
  - map: einblenden
  - automatisches Schreiben auf Disk
  - unmap schreibt ebenfalls
- Objekte können in andere Tasks 'gemapped' werden

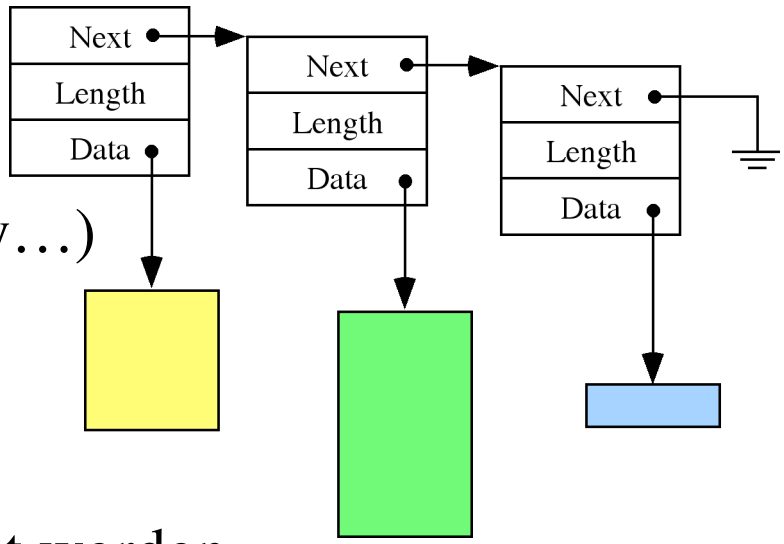


- Memory sharing
  - *shared* typisch bei code
  - *copied* typisch für Daten
  - copy-on-write semantic optimistisch (z.B. Netzwerkverkehr …)
- Beispiel `fork`
  - object-code shared
  - auch shared-libraries
  - stack, heap als Kopien
- Mach-Kern kennt nur Speicherobjekte
  - keine Dateien
  - Dateizugriff als Speicherzugriff - page-fault
  - Inhalt vom External Pager: `memory_object__data_provided`
  - Datei(-teil) wird in Adressraum eingeblendet
  - Mapping durch Kern
  - pre-fetch: spontane `memory_object__data_provided`

- Messages
  - Prozesskommunikation von Mach
  - asynchron
  - lesen evtl. blockierend
  - Transport speicherbasiert
- Port und Port-Rights
  - unicast, gerichtet (simplex)
  - Datenstruktur im Kern
  - geschützte Warteschlange mit Messages
  - ist Tasks oder Threads zugeordnet
  - Message Übertragung zuverlässig, geordnet
  - Port-Sets
- Port-Capabilities
  - Send, Send-Once, Receive
  - Capabilities Liste gehört zum Task
  - (Pointer zum Port, Recht)



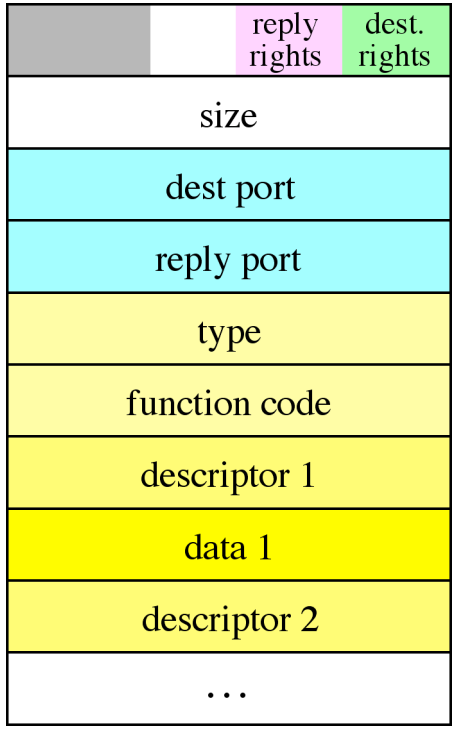
- Rechteweitergabe
  - zwischen Tasks in Messages
  - Senderecht an Port: 2 Senderechte
  - Receive-Recht an Port: Sender verliert Recht
- Ports manipulieren
  - allocate: erzeugen mit Lese-Recht
  - destroy für Lese-Port, deallocate für send, send_once
  - Managen von Port in child-tasks
- Nachrichtenformat
  - Header + {Tag,Daten}
  - Datentyp (bit, byte, int, string, float, capability…)
  - Tag: Flags,Anzahl(20), Bits(8), Typ(8)
  - evtl. Übersetzung
  - in-line: (Tag,Data)
  - out-of-line: (Tag,Pointer)
  - Speicher kann in Empfängerraum eingeblendet werden
  - copy-on write
  - scatter/gather möglich



- Senden von Nachrichten
  - Lesen und Schreiben

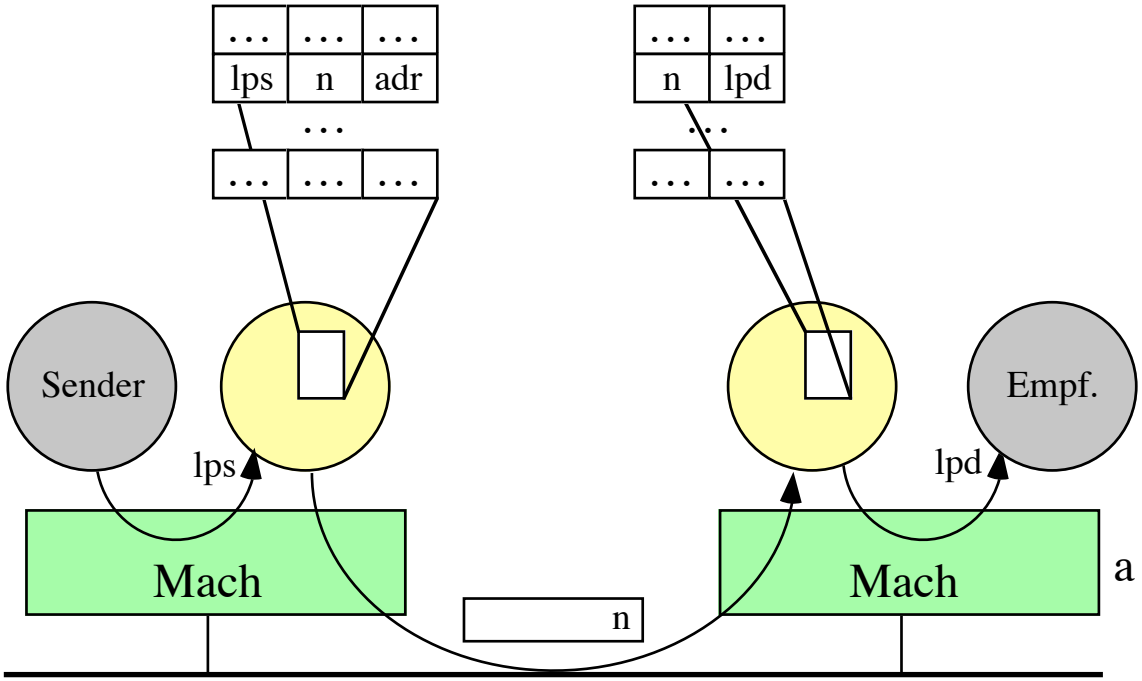
```
mach_msg(&hdr,opt,s_size,r_size,rcv_port,timeout,notify_port);
```

  - options-bits: send, receive, …
  - RPC falls send+receive
  - MIG: Mach Interface Generator -> RPC
  - nach dem Senden Datenstruktur sofort wieder verfügbar



- Besondere Kernel-Ports
  - bootstrap: andere Ports finden
  - process: messages an Kern zum Systemdienstaufruf
  - exception, registered

- User-level Prozess Network Server
  - transparenter Message-Transfer über Netzwerke



  - Tabellenmanagement anspruchsvoll
  - Portrechttransfer, besonders bei Receive

6. Konsumptive Dienste

6.1 Multimedia-Verteildienste (siehe 'Elektronische Medien')

- Audio und Video
- "Terrestrische" Verteilung
- Kabelverteilsysteme
- Satellitenverteilung

6.2 Media on Demand (siehe 'Elektronische Medien')

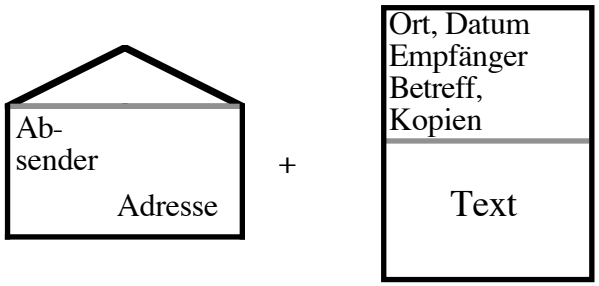
- "Individuelle" Programme
- Video on Demand
- Movie on Demand

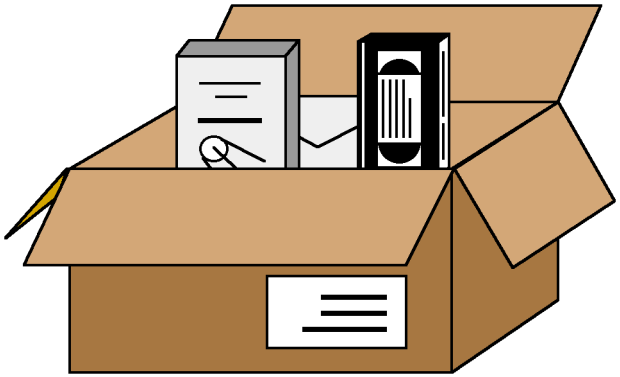
6.2.2 Movie on Demand: Server (siehe 'Elektronische Medien')

6.2.3 Movie on Demand: Set-top-box (siehe 'Elektronische Medien')

6.3 Electronic Mail

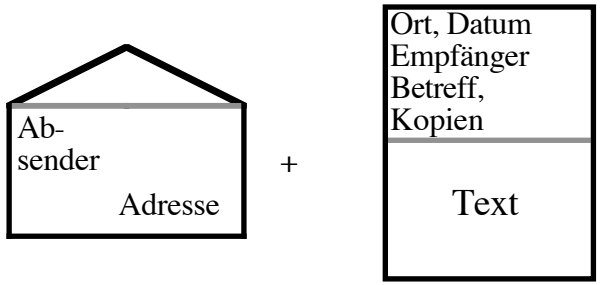
- Asynchrone, paketisierte Kommunikation
- E-Mail: Versand elektronischer Dokumente
  - Brief und Fax (Text, Grafik, Photo)
- Multimedia-Mail
  - Päckchen oder Paket
  - Text, Grafik, Photo, Audio, Video, …
- Standards über Standards:
  - X.400 (ITU),
  - SMTP (Simple Mail Transfer Protocol)
  - MIME (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





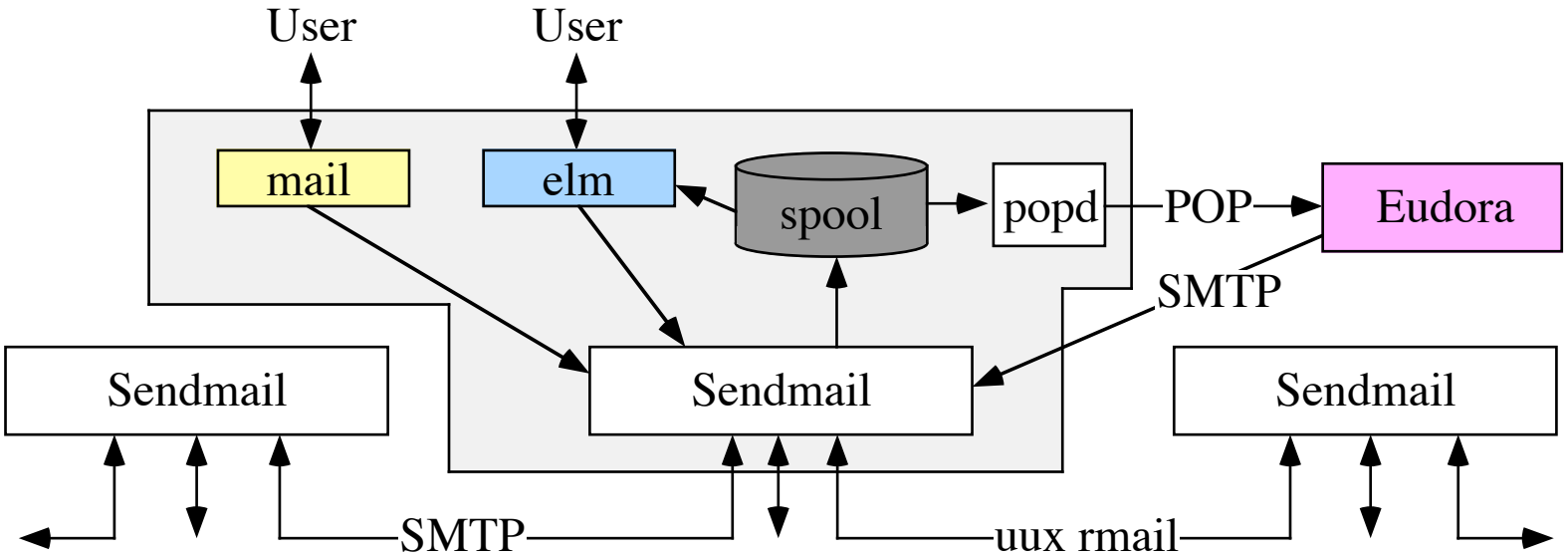
6.3.1 Internet Mail

- 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
  - big-trains@cirr.com
  - 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.  He'll be able to provide  a
              more  accurate  explanation  when  he  returns
              next week.
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

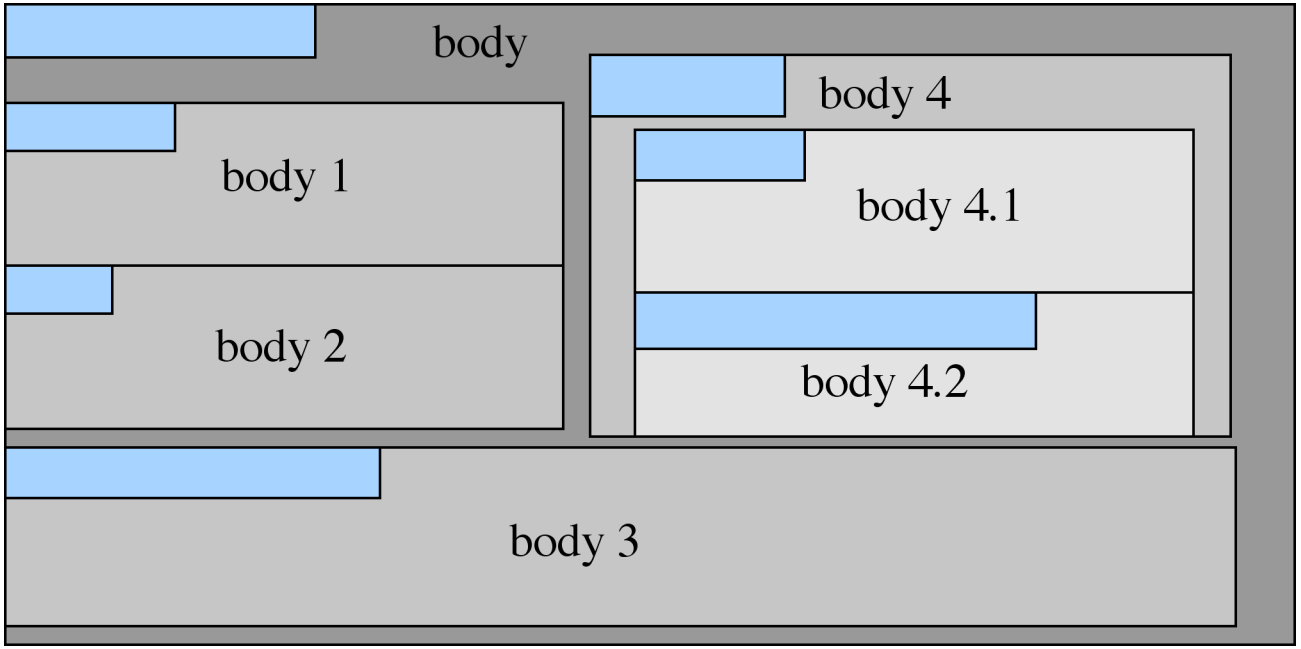
  => Server: nur Mail-Relay für bestimmte IP-Nummern-Bereiche

  => System: authenticated SMTP
- Anti-Spam heute
  - 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

6.3.2 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

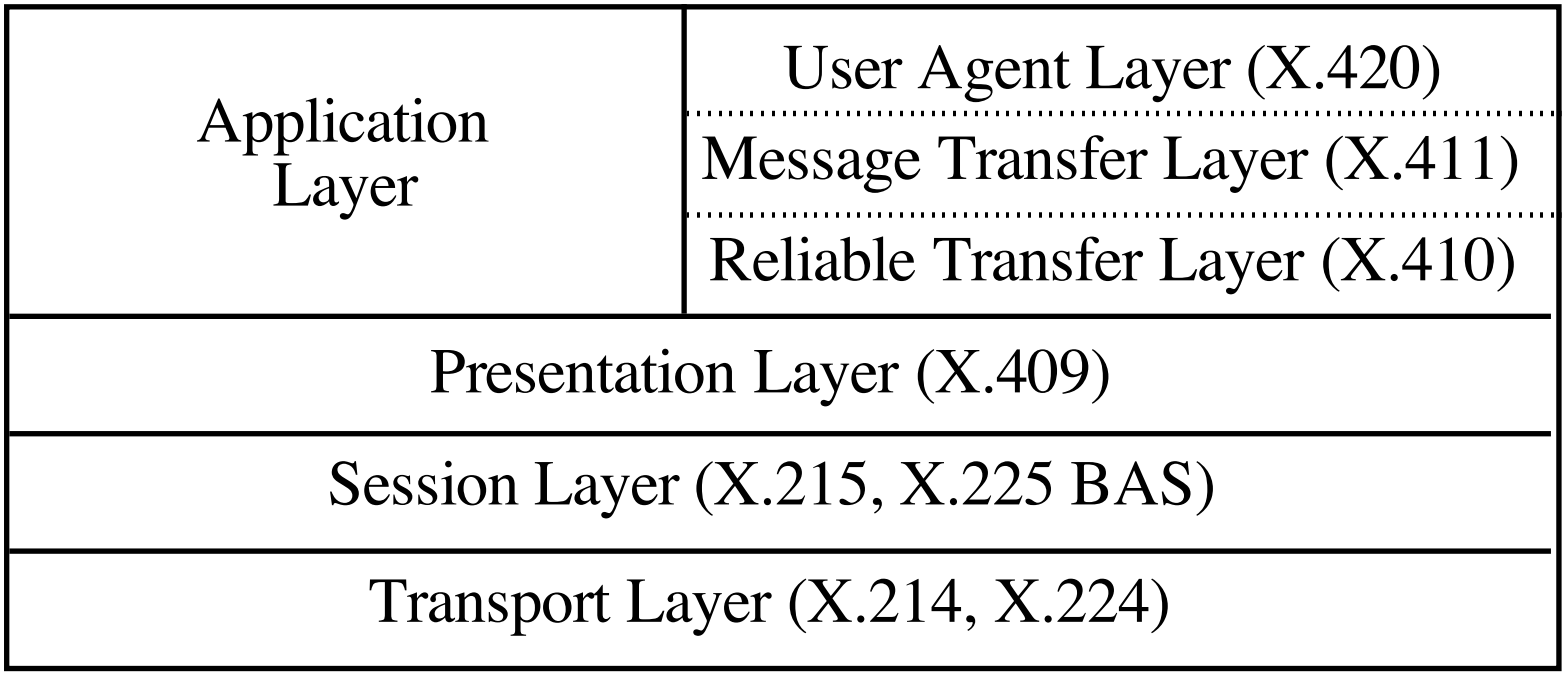
```
Content-type: text/plain; charset=ISO-8859-1
Content-transfer-encoding: base64
```

- 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
```

6.3.3 X.400

- ITU-Standard



- Message = Umschlag + Inhalt
- Message Transfer Layer: Umschlag
- User Agent Layer beschreibt Briefe (Inhalt)

- X.400 Nachricht
  - IM-UAPDU (Interpersonal Message User Agent Protocol Data Unit)
  - Header und Body
  - bestehen aus Objekten
- Formatbeschreibung von X.400: X.420 (ASN.1)
- Objekt: Tag Length Contents

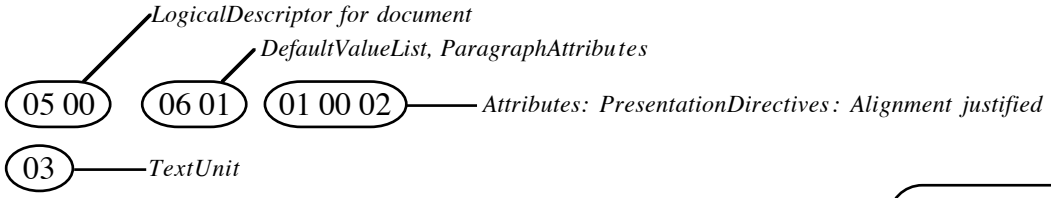
Typ	Tag	Length	Content	Wert
Boolean	01	01	FF	TRUE
Integer	02	05	0100000000	4294967296
BitString	03	03	03AF38	00..01010111100111
IA5String	16	05	5045544552	'PETER'

- Typ und Length ein Byte
  - Typ: 0..30, 31 : Escape
  - Length: 0..127 oder 128 + Länge des Längenfeldes
- Transfersyntax zur Definition und Übertragung neuer Typen

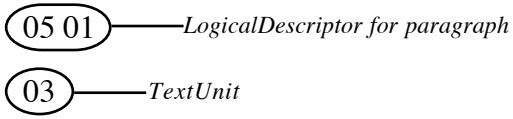
- Header:
  - Absender, den Empfänger, die Liste der Kopienempfänger,
  - Verfalldatum, Betreff, etc.
  - Reihenfolge fest
- Typen im Body

```
BodyPart ::= CHOICE {
    [0]    IMPLICIT IA5Text,
    [1]    IMPLICIT TLX,          -- Telex
    [2]    IMPLICIT Voice,
    [3]    IMPLICIT G3Fax,           -- T.4, Telefax G3
    [4]    IMPLICIT TIF0,            -- Text Interch. Format 0,
                                     -- T.73, Telefax Gruppe 4
    [5]    IMPLICIT TTX,          -- T.61, Teletex
    [6]    IMPLICIT Videotex,        -- T.100 BTX
    [7]    Nationally Defined,
    [8]    IMPLICIT Encrypted,
    [9]    IMPLICIT ForwardedIM,  -- IM-UAPDU
    [10]   IMPLICIT SFD,          -- Simple Formattable Dokument
    [11]   IMPLICIT TIF1,            -- Text Interchange Format 1,
                                     -- T.73, "Textfax"
}
```

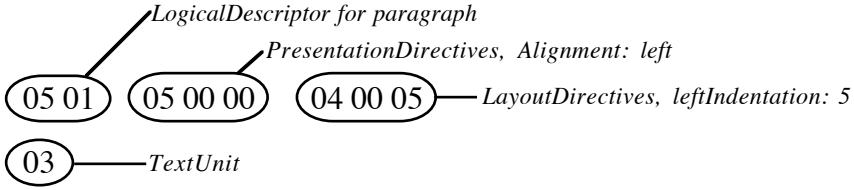
- Beispiel
  - nur einfacher Text
  - primitive Formatierung
  - vordefiniert: presentation directives, layout directives



'In der obersten Zeile wurde ein l ogischer Deskriptor
für das Dokument angegeben, der B locksatz als Voreinstellung
für alle Paragraphen vorsieht.'



'Dieser Paragraph hat keine besond eren Attribute.'

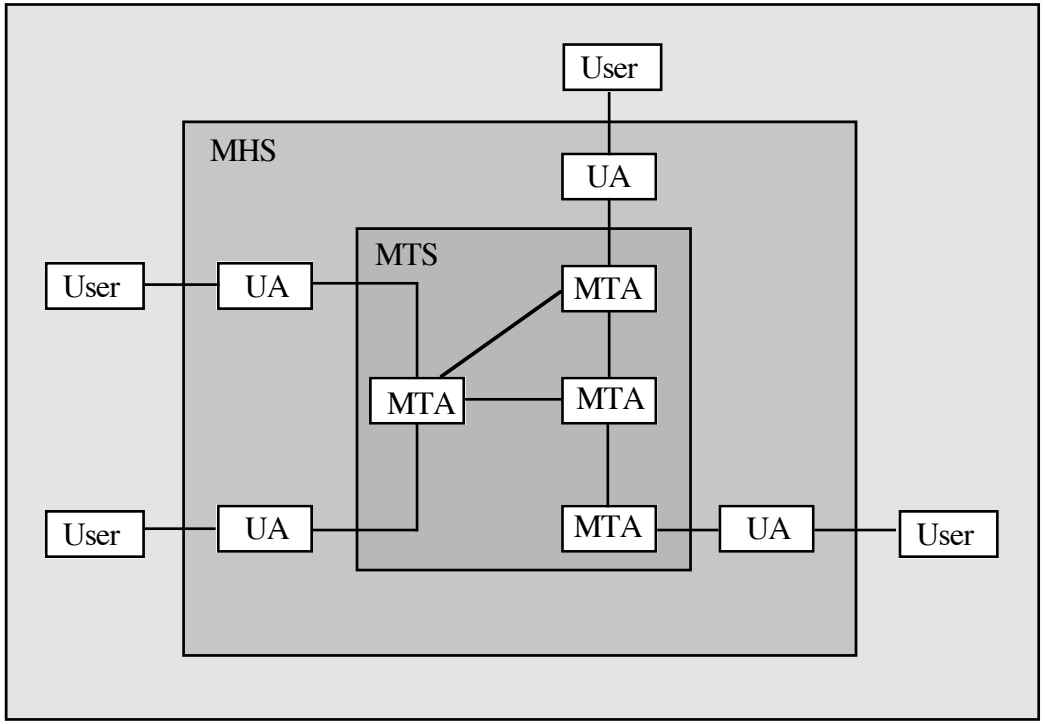


'Für diesen Paragraphen wurde link sbündig und ein linker Rand von fü nf eingestellt.'

```
In der obersten Zeile wurde ein logischer
Deskriptor für das Dokument angegeben,
der Blocksatz als Voreinstellung für alle
Paragraphen vorsieht.
Dieser Paragraph hat keine besonderen
Attribute.
     Für diesen Paragraphen wurde
     linksbündig und ein linker Rand von
     fünf eingestellt.
```

- X.400 und Multimedia?
- X.400 1988 Body Part Types:
  - ia5text, teletex, encrypted, message
  - voice
  - g3-facsimile, g4-class1, mixed-mode
  - videotex
  - externally-defined (ASN.1 Macro)
- keine Struktur-Information
  - zwischen Body-parts
  - weder räumlich noch zeitlich
  - Anordnung der Multimediaelemente?

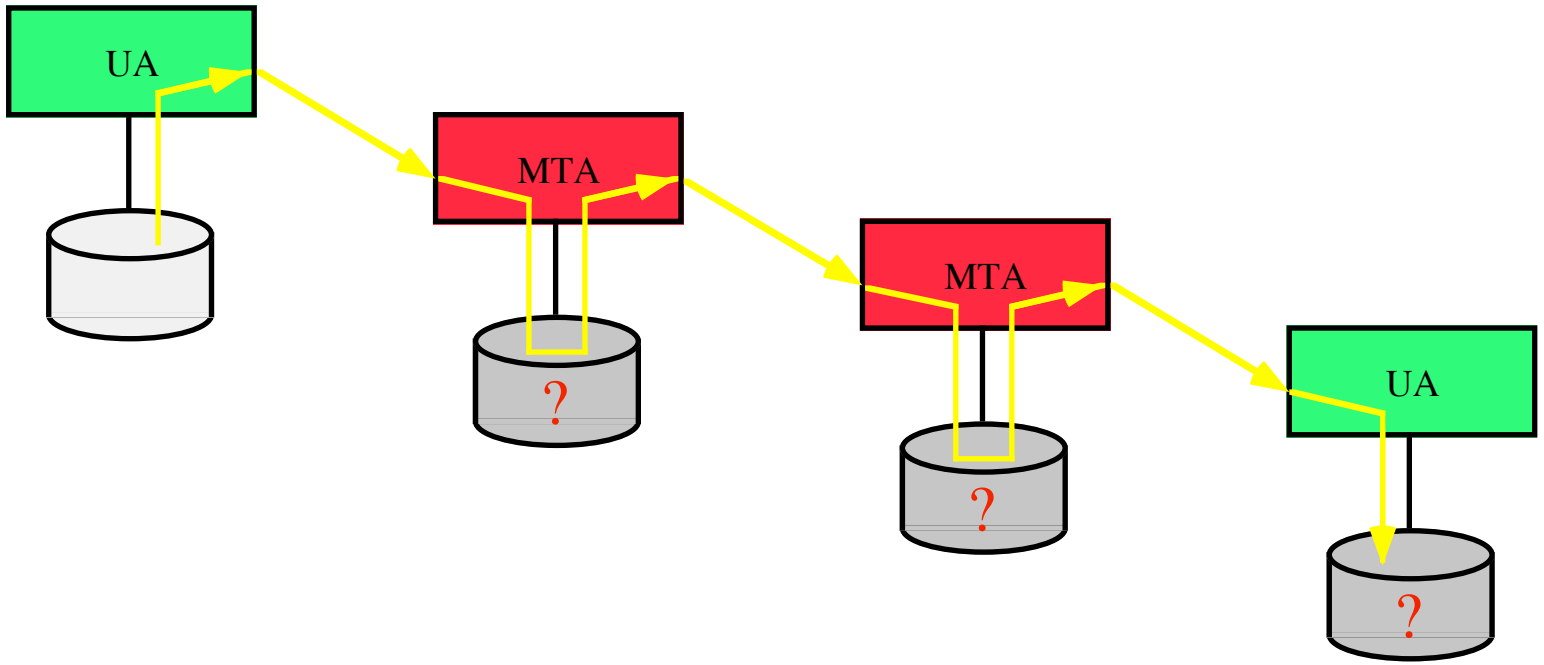
- Verteilungsmodell: zentral
  - Zwischenspeicherung im Netz



- Bewertung
  - mehr Funktionen als SMTP
  - Schwerpunkt des Dienstes im Netzwerk
  - vom Aussterben bedroht

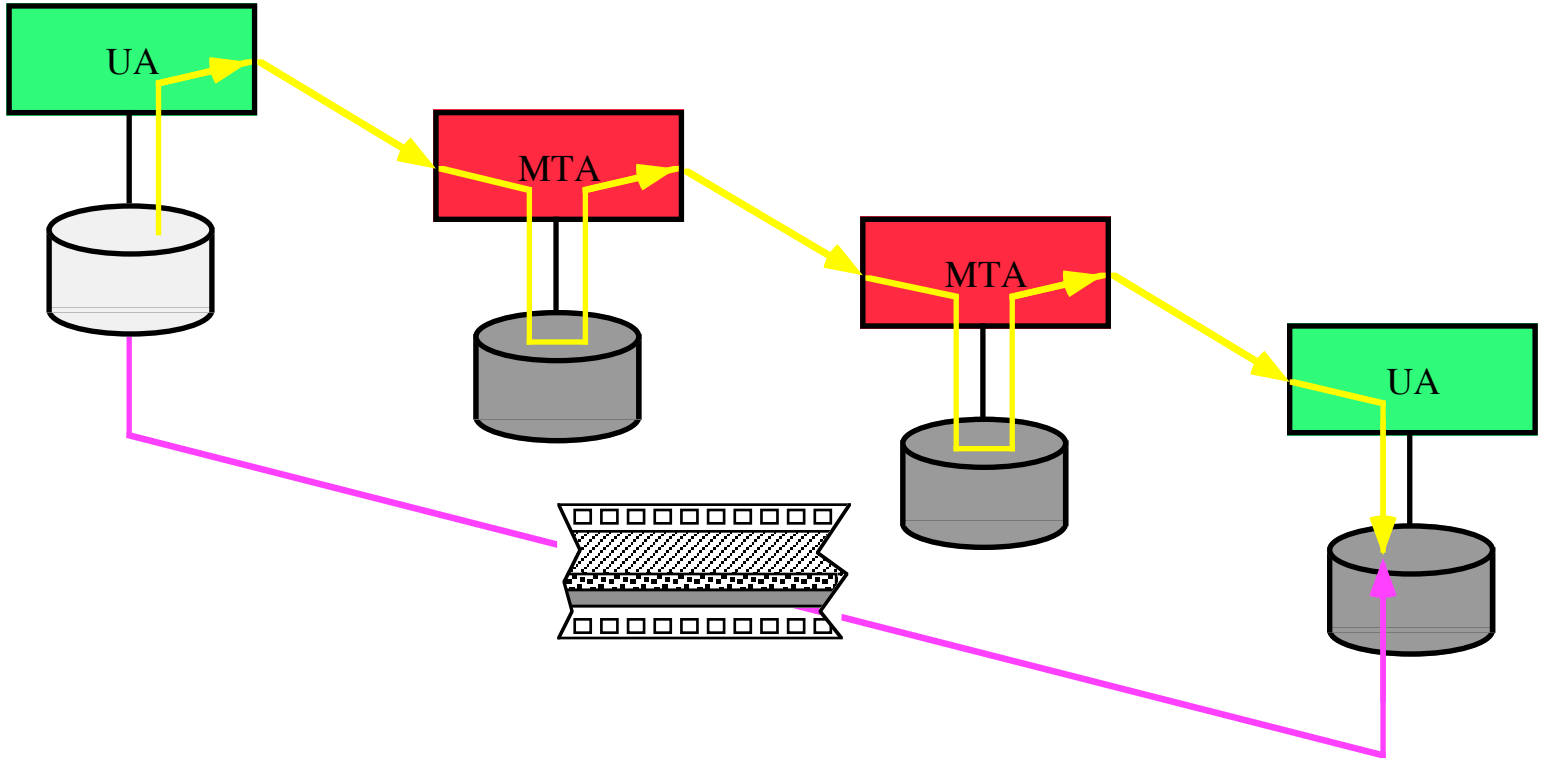
6.3.4 Multimedia Mail

- Zwischenspeicherung ist Teil des Konzeptes von Mailsystemen
  - X.400: im Netz
  - Internet-Mail: in Mailrechnern bzw. PCs



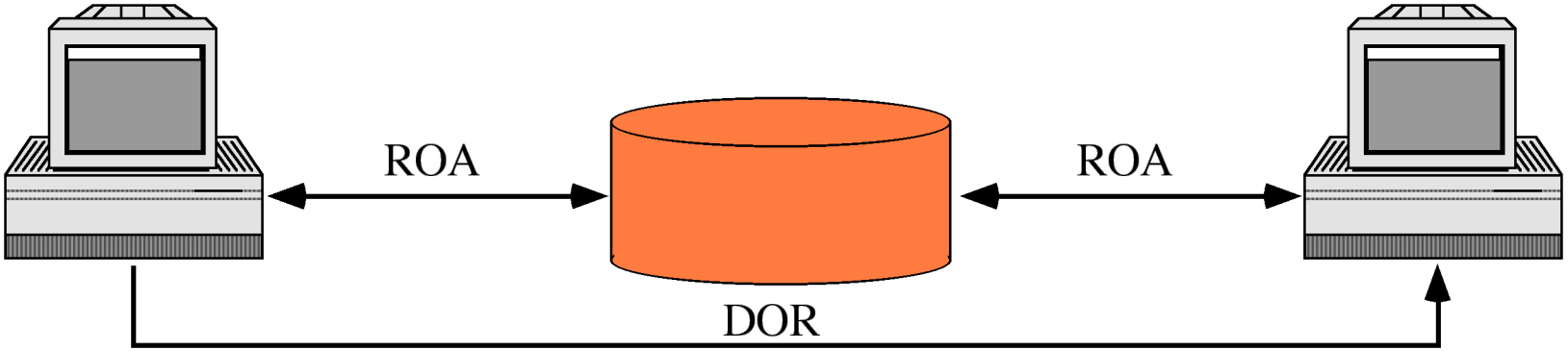
- Multimedia impliziert große Datenmengen
- Mailserver und Empfangssystem haben eventuell nicht genug Speicher

- Synchrones Abholen der Multimedia-Komponenten
  - konventionell wird nur eine Liste von Referenzen verschickt
  - während der Präsentation der Nachricht Referenzen auflösen



- Echtzeitbedingungen vs. Speicherplatzaufwand
- eventuell 'Global Store'
- URL oder DOR

- DOR - Distinguished Object References
  - ISO 10031-2; 1991



- Komponenten einer DOR

Komponente	Inhalt	vergleichbar
AE-Identifier	Application Entity im Speicher	
Local reference	App. abhängig	2. Teil der URL
Data object type	ASN.1 external type	MIME-type
QoS	Aktualitätsstufe	expiration

6.4 WWW

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

6.4.1 Grundlagen

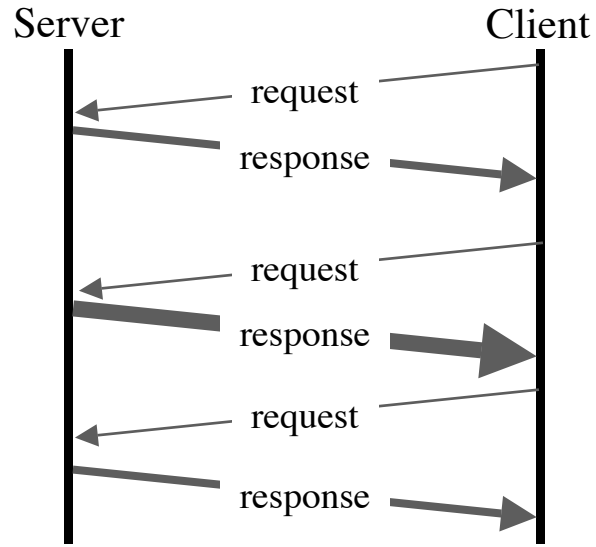
- Internetdienst wie eMail, FTP, gopher
  - Protokoll HTTP
  - eMail: **SMTP** <->WWW: **HTTP**
  - Dokumentenformat html

6.4.1.1 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
- Relative URLs: /test/Beispiel0.html, Beispiel0.html, ../movies/

6.4.1.2 HTTP

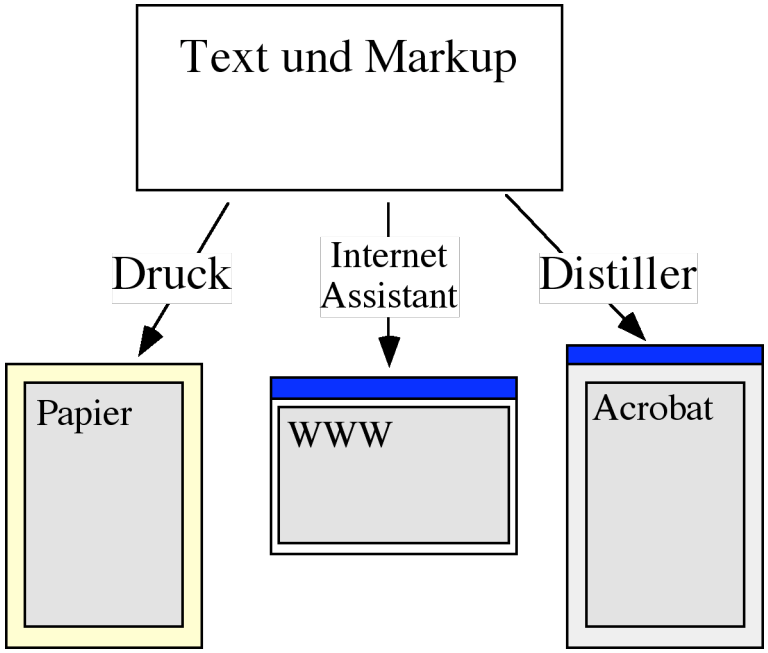
- Client-Server Modell
- Request-Response Mechanismus:
  - Request: Typ, Attribute (Header fields / Request fields), Objekt
  - Response: Typ, Attribute (Object Metainformation), Objekt
- Request-Typ (Methode)
  - **GET**, PUT (integrierte Dokumenterstellung)
  - POST, DELETE, ...
- Response-Typ / Code
  - **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
```

6.4.1.3 HTML, die 'Sprache' des WWW

- Hypertext Markup Language
  - Berners-Lee 1989
  - Zweck: 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

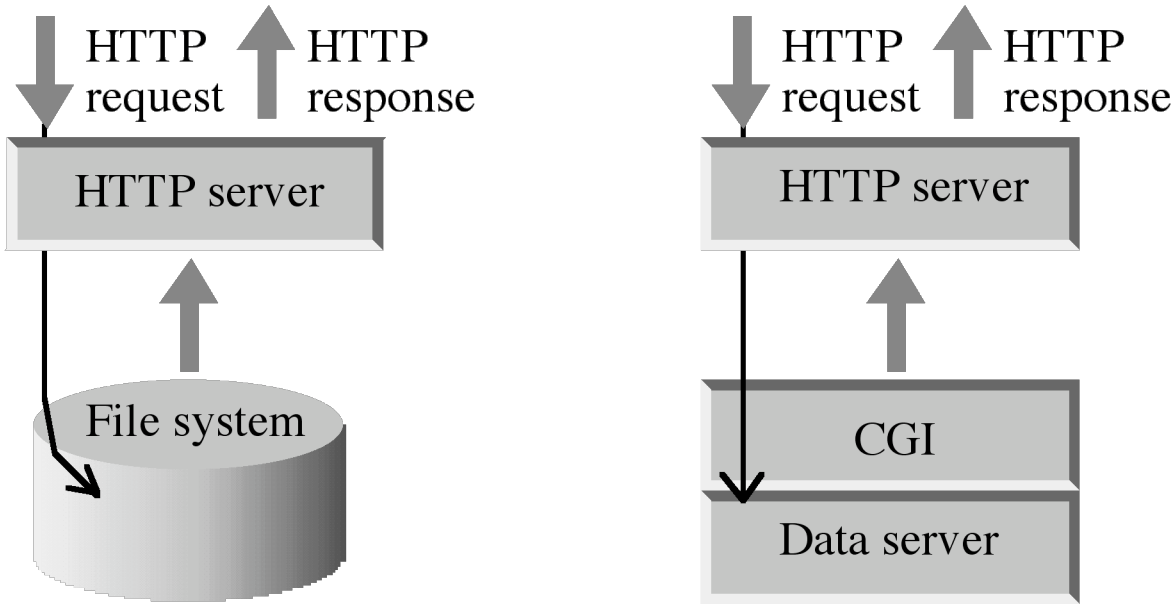
- Elemente:
  - Stile
  - Listen
  - Formatierung
  - Links

- 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)
  - MacOS: Dateityp + Creator
  - auch Heuristiken im Klienten: Namenserweiterung, Inhaltsanalyse
- XML, CSS, …

6.4.2 Details

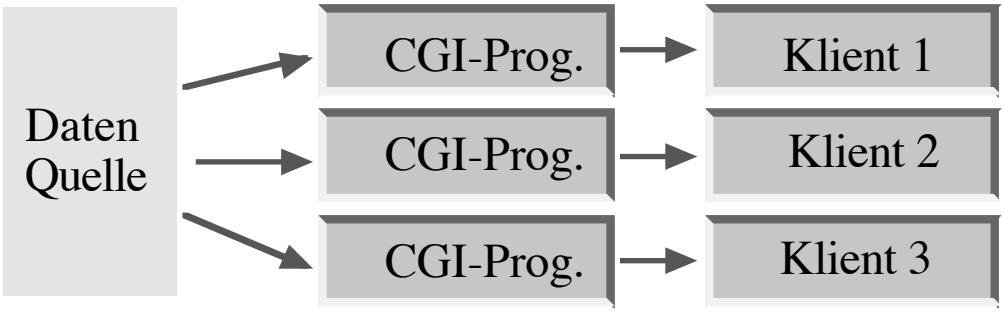
6.4.2.1 Erweiterungen auf Serverseite

• 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)



6.4.2.2 Andere Dienste im WWW

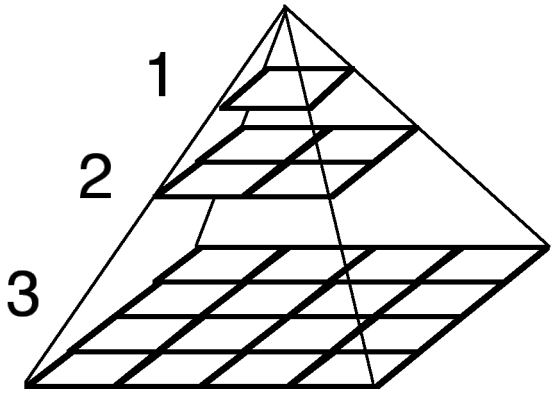
- Dienst identifiziert durch Dienstkennung im URL
  - URL = Dienstkennung : dienstspezifischer Teil

    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

6.4.2.3 Antwortzeit Optimierungen

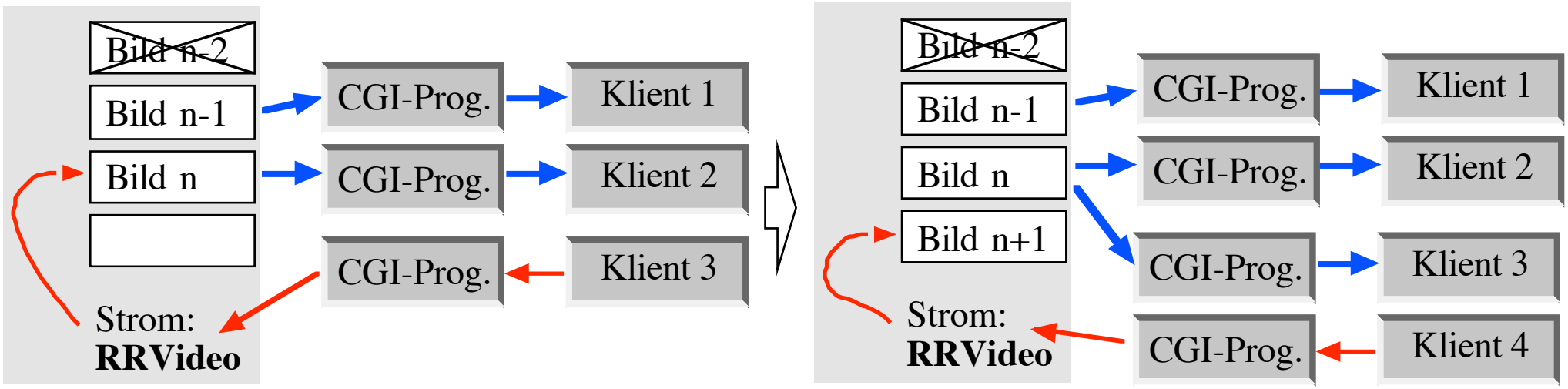
- Caching von Objekten (URLs)
  - lokal: Arbeitsspeicher, Massenspeicher
  - Proxy-Server: transparenter Mittler/Cache zwische Klient und Server
    auch mehrstufig: Fakultät, Universität, Provider, Kontinent ?
  - Expiration-Date für Cache-Management
- Hierarchisch organisierte Objekte:



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

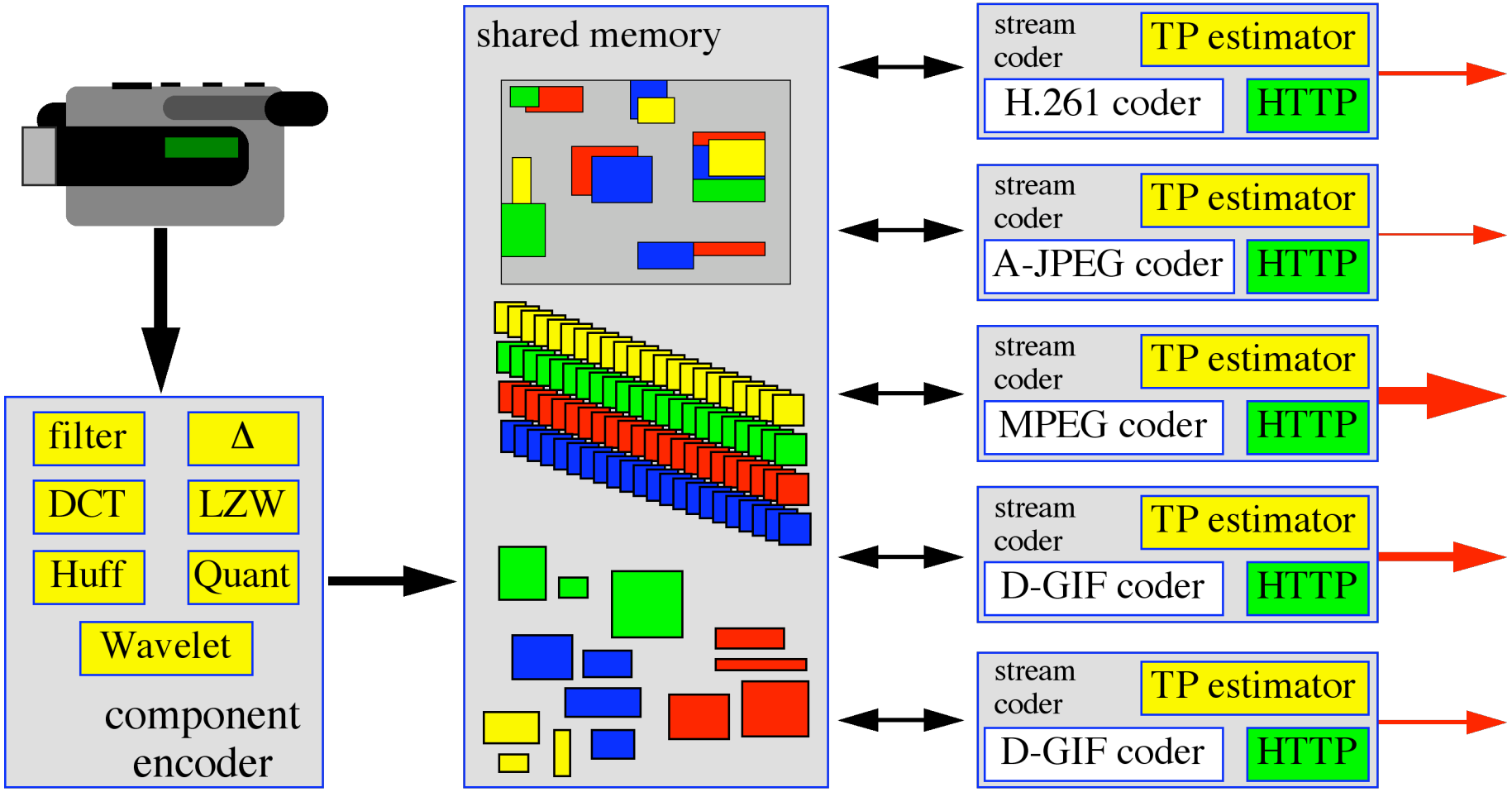
6.4.2.4 Video / Animation

- abgelegte Daten: animated GIF
- live Video
  - Datenquelle: Videostrom
  - Bildentnahme entspricht Datenbankabfrage
  - Daten im Shared Memory - addressiert durch Stromnamen
  - Zugriff immer auf aktuelles Bild
  - alte Bilder werden verworfen, wenn letzte Referenz verschwindet



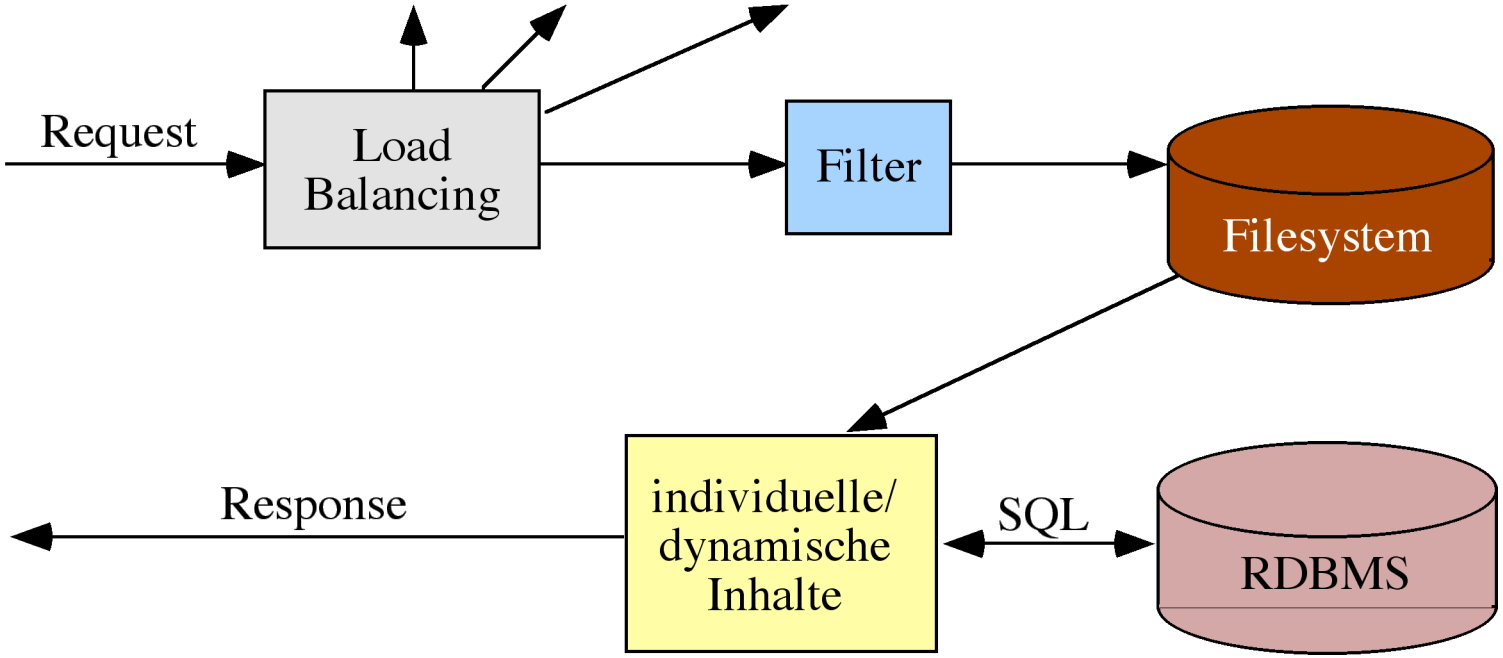
  - alle Klienten haben unterschiedliche Transferraten
  - Bereitstellung mehrere Formate: GIF, JPEG, MPEG

- Component Encoding - Stream Construction (CESC)
  - Internet-QoS heterogen und dynamisch

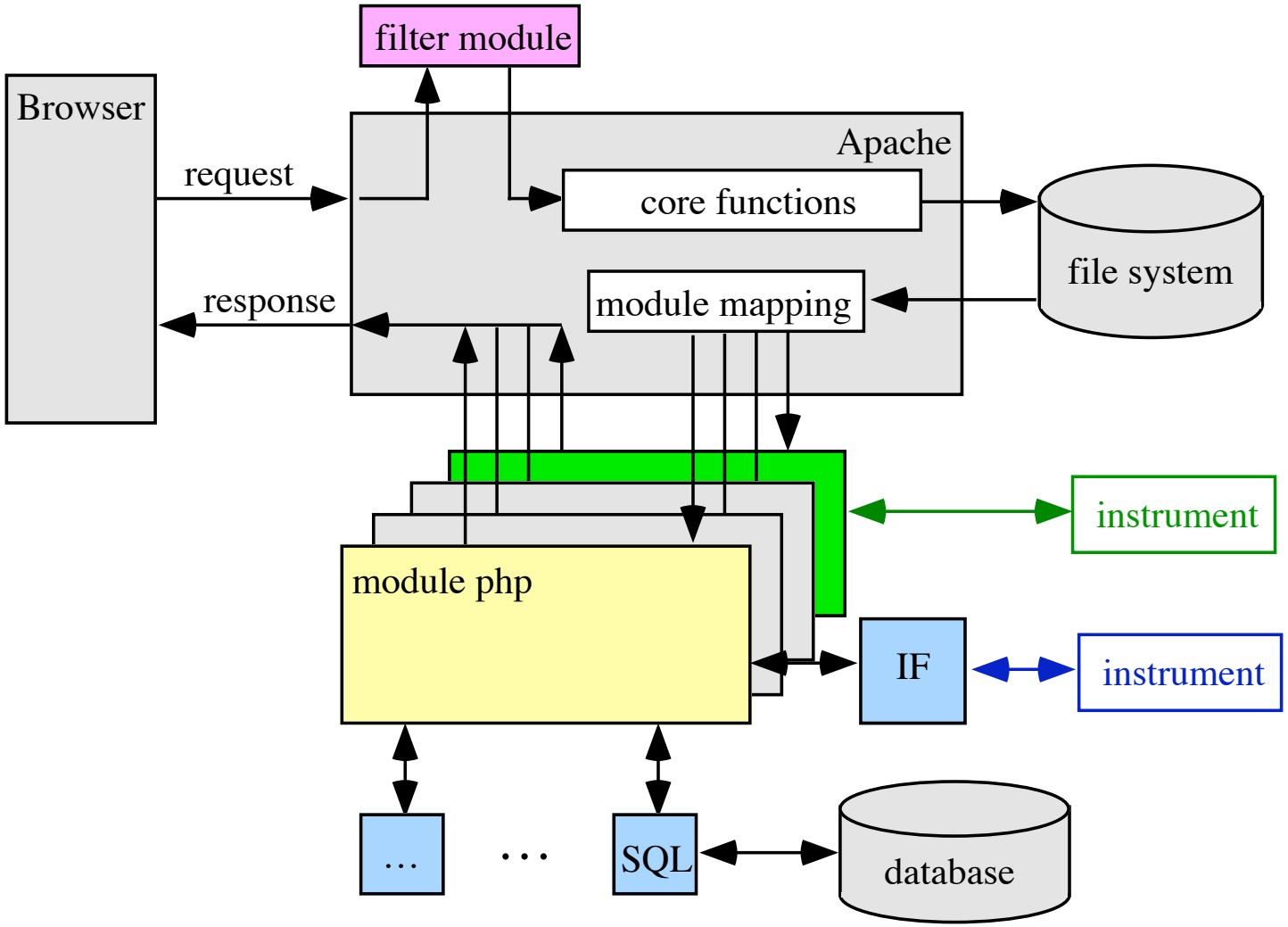


6.4.3 Struktur eines Webservers

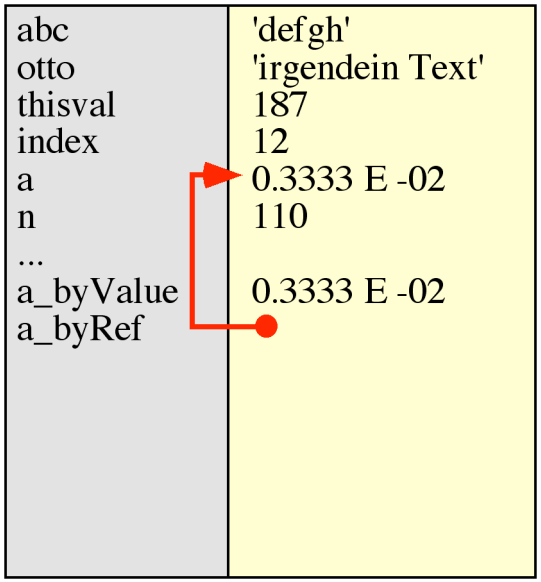
- Sites mit viel Verkehr
- Inhalte an individuellen Besucher anpassen
  - Shopping System
  - Abonnent, Suche, …
- Systematische Integration weiterer Datenquellen
  - Datenbanken
  - beliebige Programme



- 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

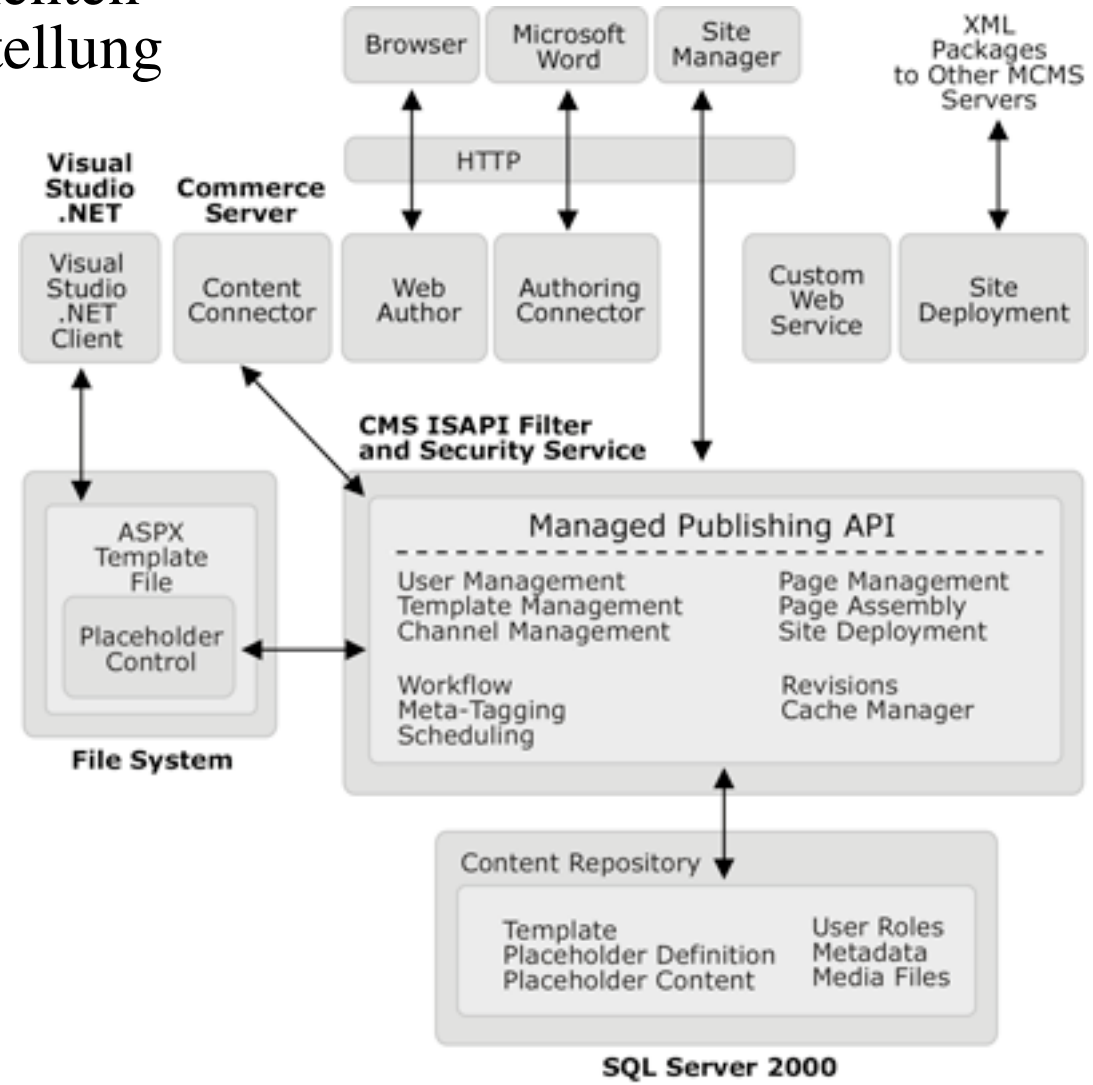
```
<html> <head> <title>Example</title> </head>
    <body>
        <?php
         if (strstr($_SERVER["HTTP_USER_AGENT"], "MSIE")) {
           echo "<p> You are using Internet Explorer </p>";}
        ?>
    </body>
</html>
```

  - 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:

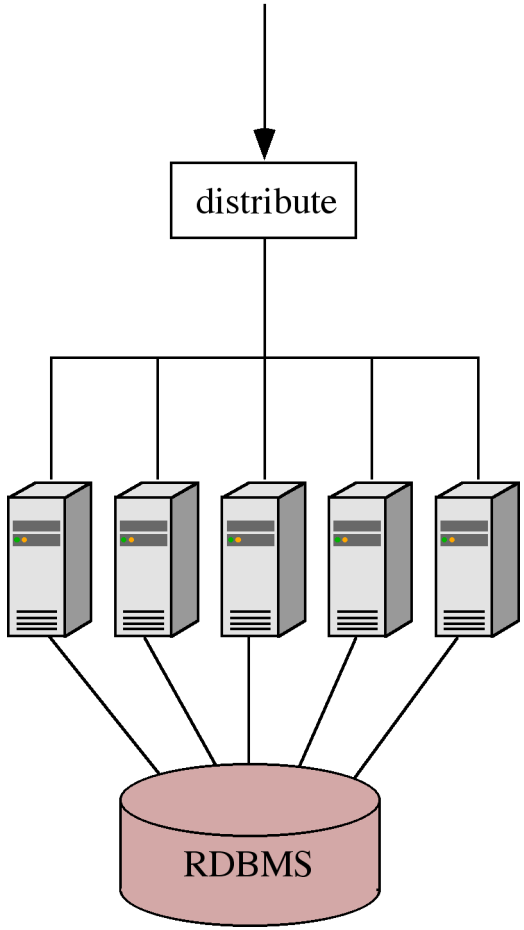
```
<html> <head> <title>Example</title> </head>
    <body>
        <p> You are using Internet Explorer </p>
    </body>
</html>
```

- 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

6.4.4 JavaScript

- Programmfragmente in HTML
  - Verbesserung von HTML-Seiten auf der Klienten-Seite
  - von Netscape
  - Fenstergröße und -Gestaltung
  - Menus, Effekte, …
  - Beispiel: http://ara.informatik.tu-freiberg.de
- 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;
```

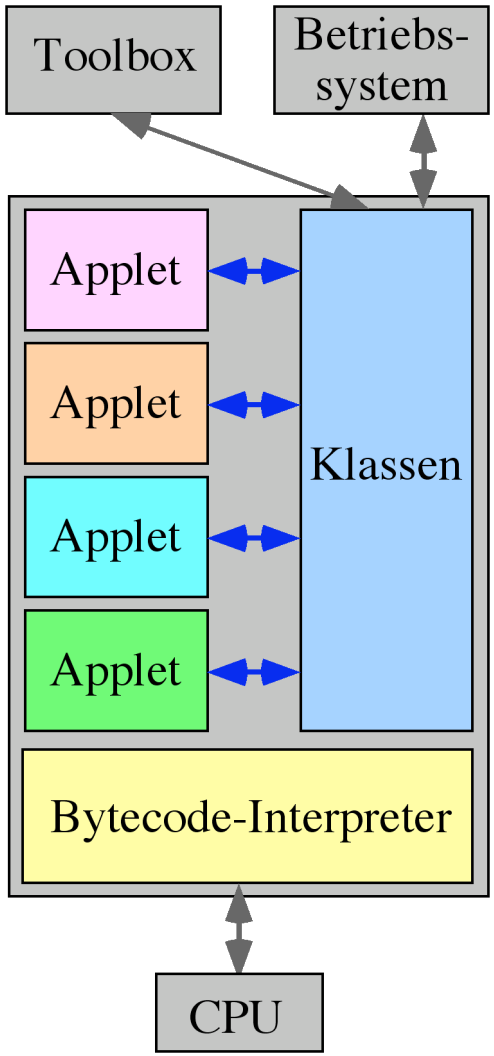
  - Methoden zur Manipulation

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

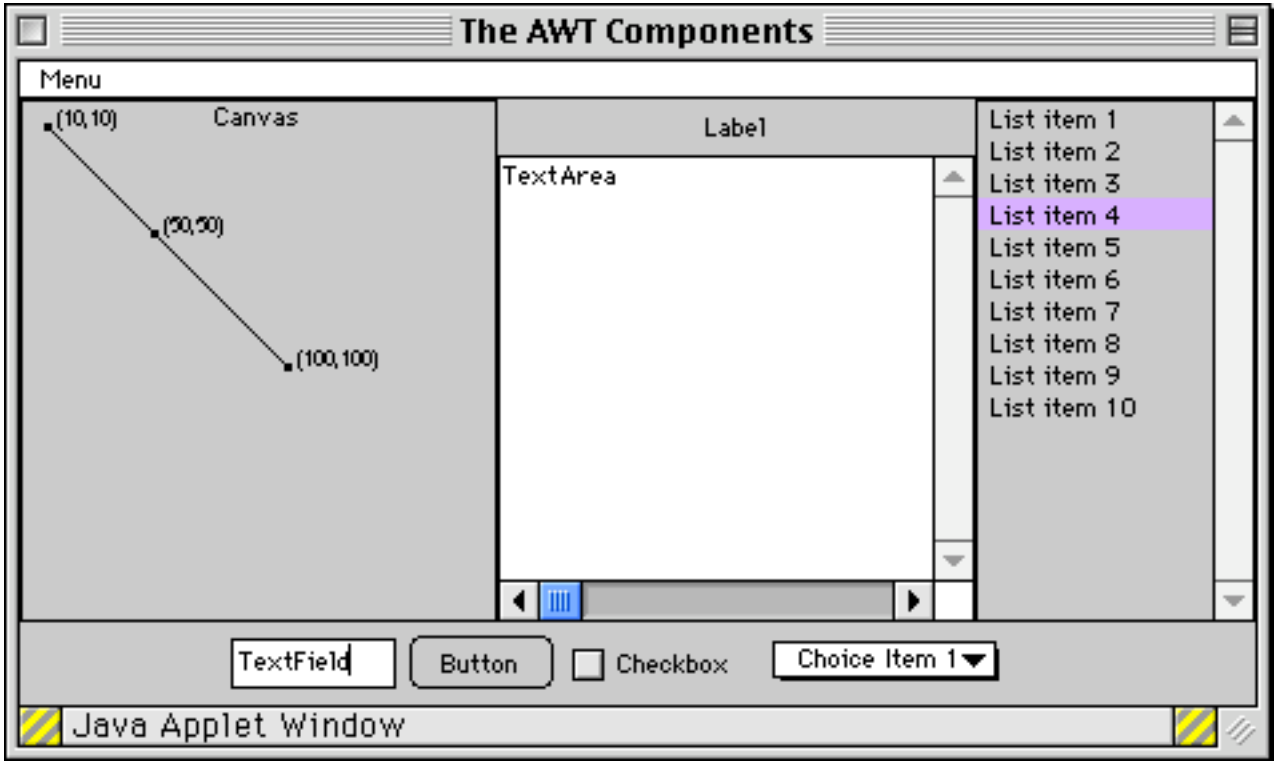
- Funktionen
  - Anweisungsblock mit Variablen
  - Aufruf aus anderen Funktionen und in Eventhandlern

6.4.5 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



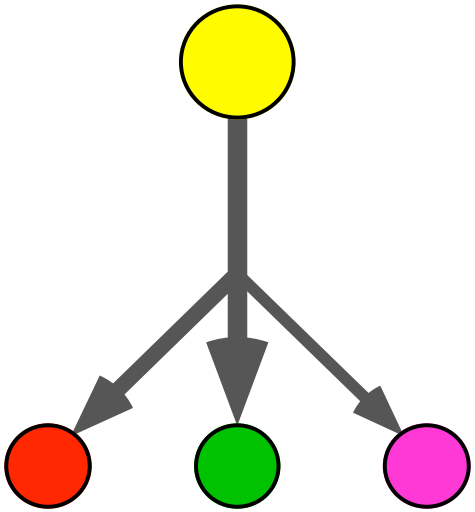
- Java Runtime Environment
  - Virtuelles Betriebssystem mit Toolbox
  - Klassen mit häufig gebrauchten Funktionen
  - Strings, Netzwerk, …
  - AWT: Buttons, Checkboxes, Choices, Lists, Menus, Text Fields
    Windows, Panels, Scroll Panes
    Grafische Objekte



- 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)

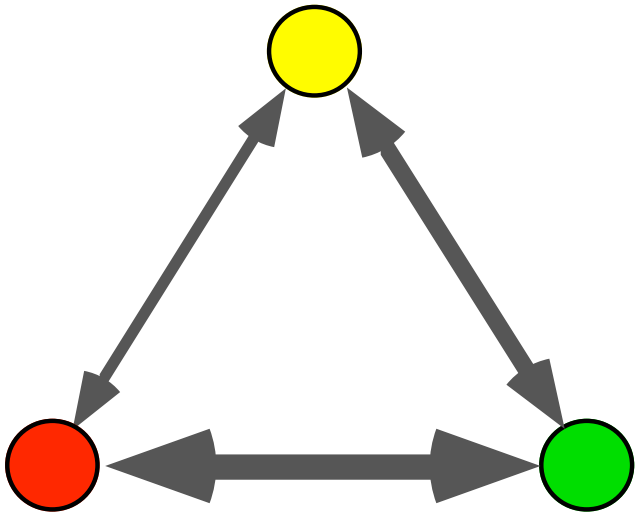
6.5 Stichwort Web 2.0

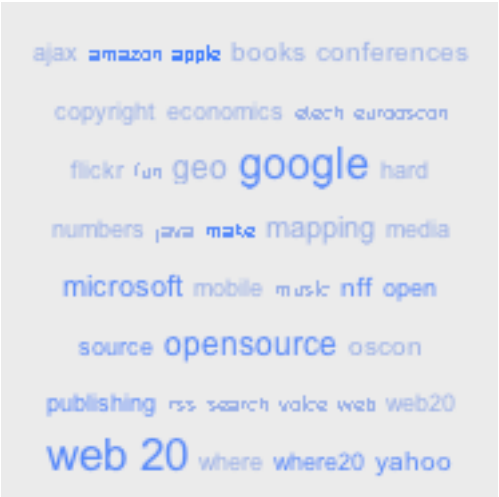
- Das WWW der Wissenschaftler
  - Publizieren
  - Links sammeln, Nützliches anbieten
- Web-Präsenzen
- Meta-Seiten
  - Yahoo's Linksammlung
  - Suchmaschienen (Altavista, Google, …)
- Statische Webseiten
  - request-response
  - makro-interaktiv
  - Inhalte ändern sich (sehr) langsam
- Datenbank-Webseiten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



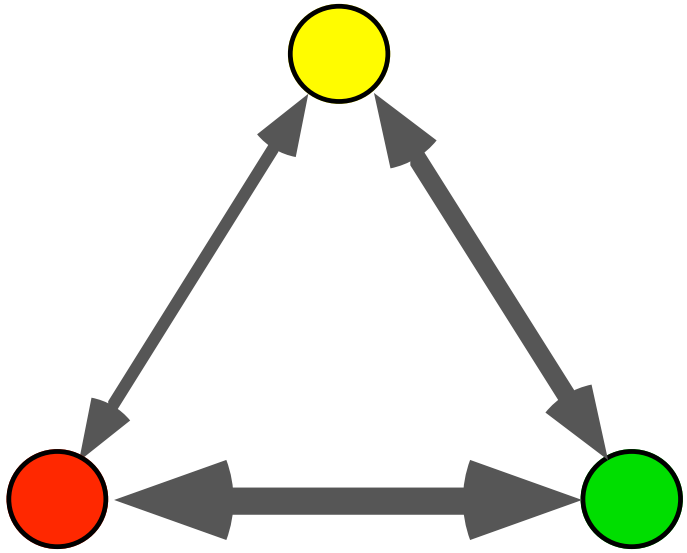
- Erfolgreiche kommerzielle Webdienste?
- Ebay
  - Transaktionssystem
  - Handelsplattform
  - Regeln
  - Kassieren
- Amazon
  - ISBN und Inhalt vom Buchhandel
  - Logistik- und Zahlungssystem
  - Handelsplattform
  - Benutzerkommentare
  - Auswertung des Kaufverhaltens
- Amazon Web Services
  - E-Commerce 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



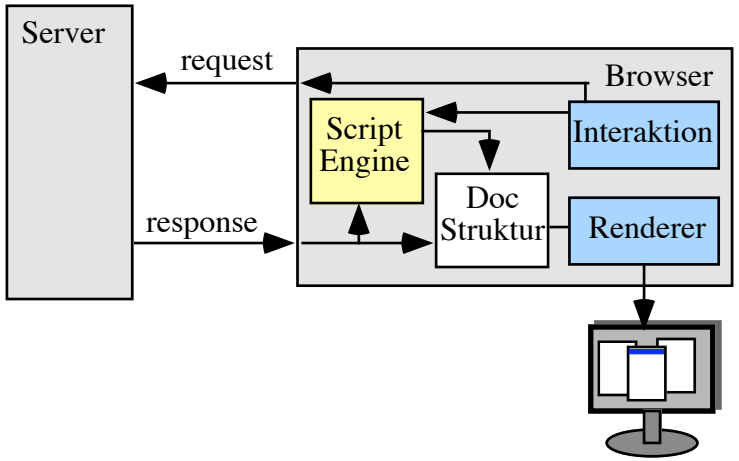


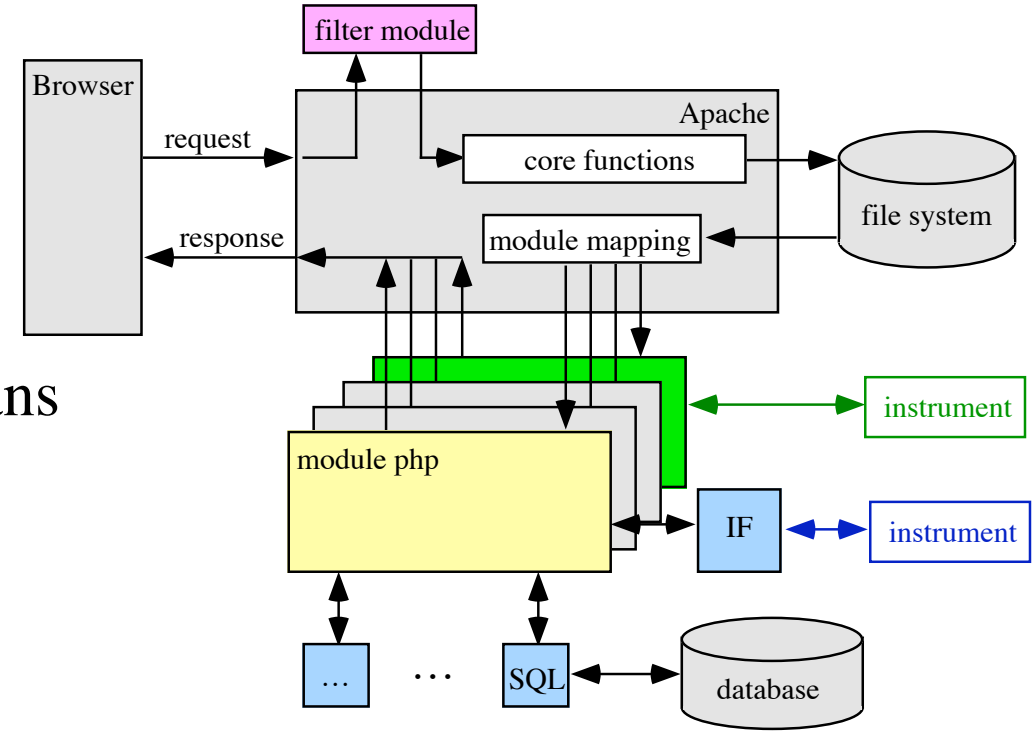
- Gemeinsamkeiten erfolgreicher Webdienste
- Infrastruktur für Benutzer
  - Benutzer erstellen den eigentlichen Inhalt
  - Benutzer schaffen die Werte
- Benutzer arbeiten lassen
  - Auktion erstellen
  - Artikel beschreiben
  - Rezensionen schreiben
  - bloggen und kommentieren
- Benuterbeiträge kanalisieren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



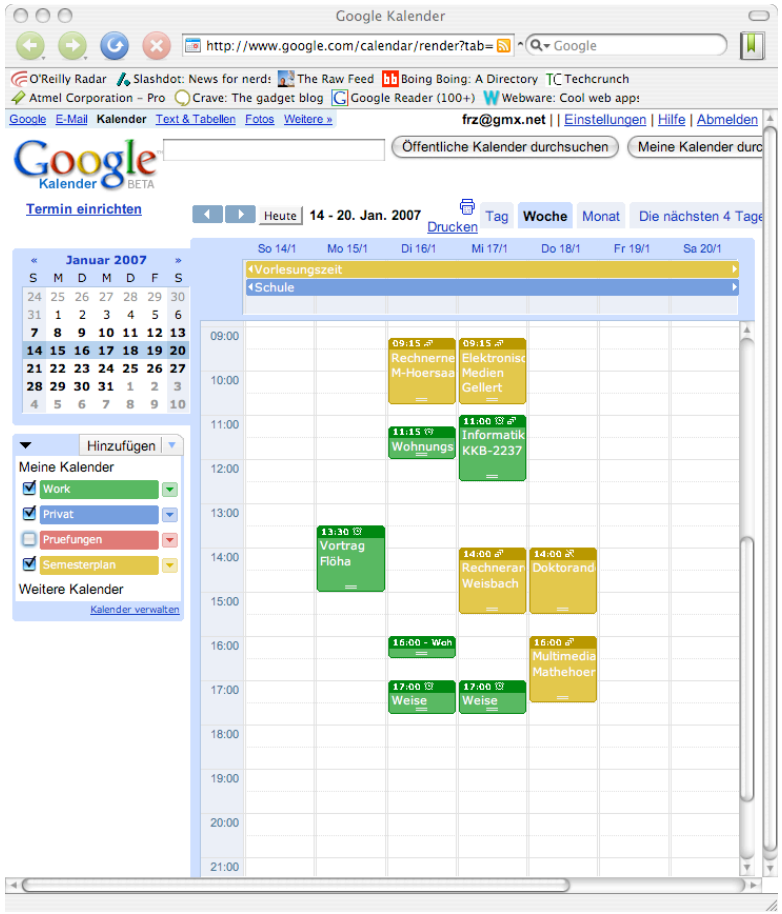
- 7 Thesen von Tim O'Reilly
- Einzigartige Daten
  - explizit und implizit vom Benutzer
  - Beispiel Amazon
- 'kollektive Intelligenz' 'bändigen'
  - Wikipedia, Google, Sourceforge, Blogs
  - tagging
- Services anstelle von Softwarepaketen
  - Google statt Netscape
  - Web-Plattformen: Amazon, Google, …
- 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

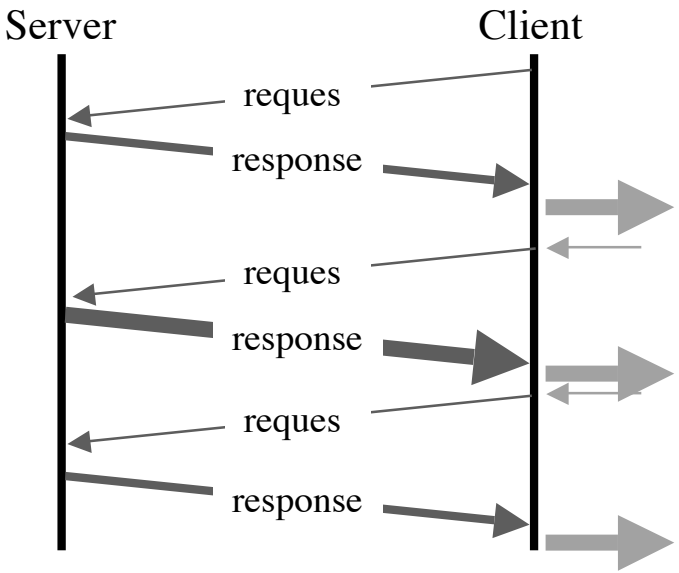


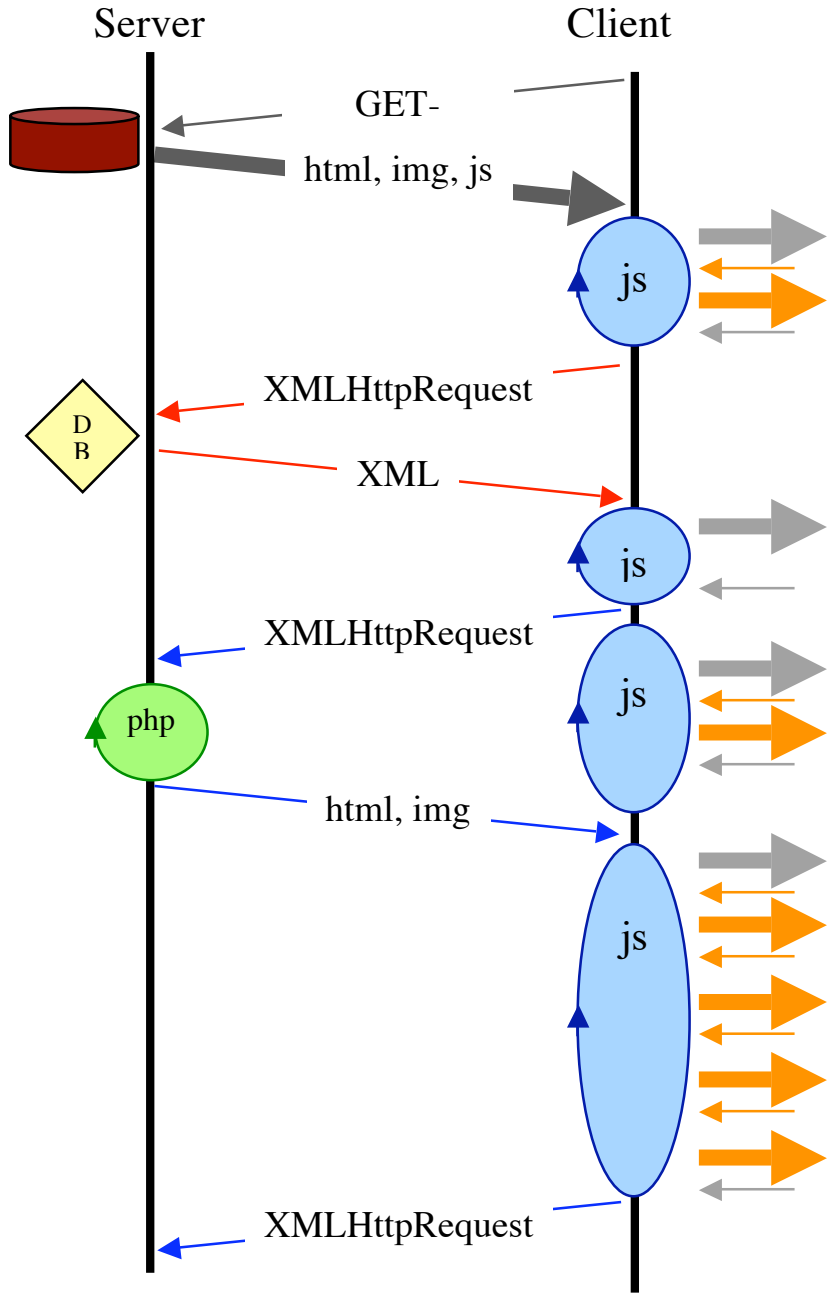


- Programmierparadigma
- Infrastruktur
  - Server:(Apache+phplJavaBeans)
  - Server:Datenbank
  - Client:(InternetExplorerlFirefoxlSafari)
- Krümelware
  - kleine, vorgefertigte Bauteile (-> Objekte)
  - abgeleitete (erweiterte) Objekt
  - inkrementelles Programmieren
  - Einfügen und Erweitern statt 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

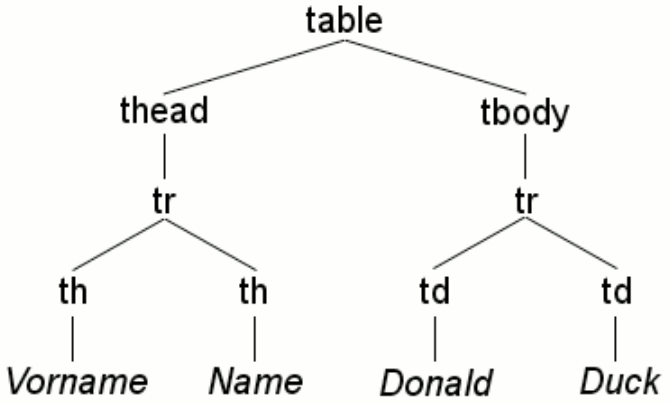
```
function createXMLHttpRequest() {
 try {return new ActiveXObject("Msxml2.XMLHTTP"); } catch(e) {}
 try {return new ActiveXObject("Microsoft.XMLHTTP"); } catch(e) {}
 try {return new XMLHttpRequest(); } catch(e) {}
 alert("XMLHttpRequest not supported");
 return null;}
```

- 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

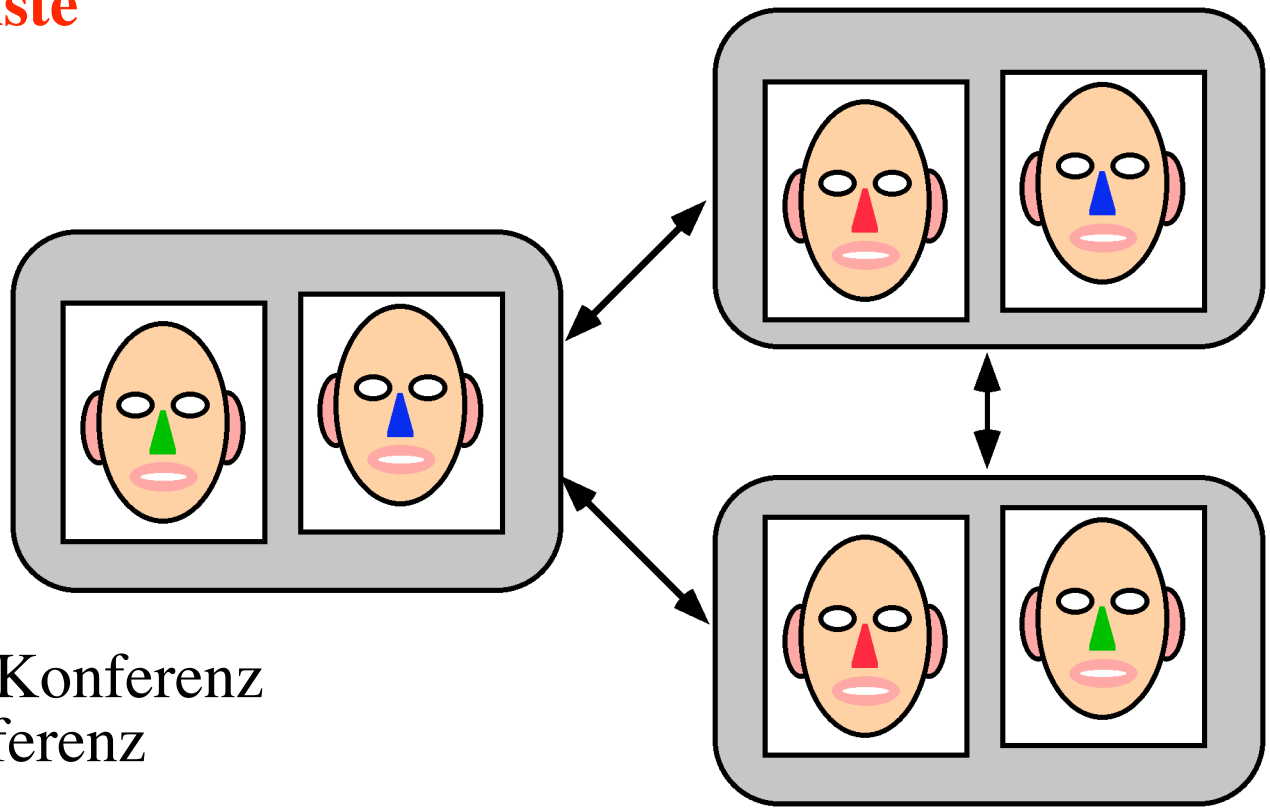
```
<table>
   <thead>
     <tr>
       <th>Vorname</th>
       <th>Name</th>
     </tr>
   </thead>
   <tbody>
     <tr>
       <td>Donald</td>
       <td>Duck</td>
```



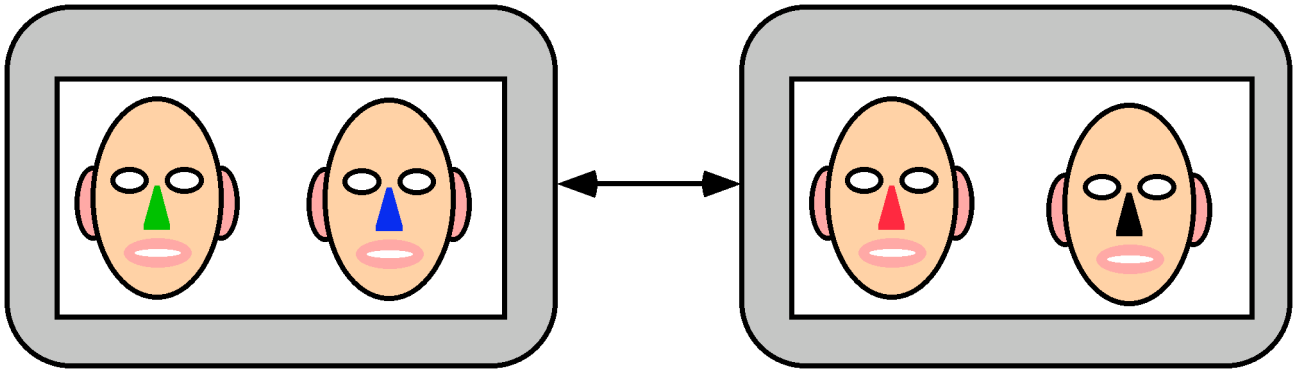
Quelle: de.wikipedia.de/wiki/Document_Object_Model

7. Kooperative Dienste

- (Video-)Telefon
  - Punkt-zu-Punkt
  - Mehrpunkt

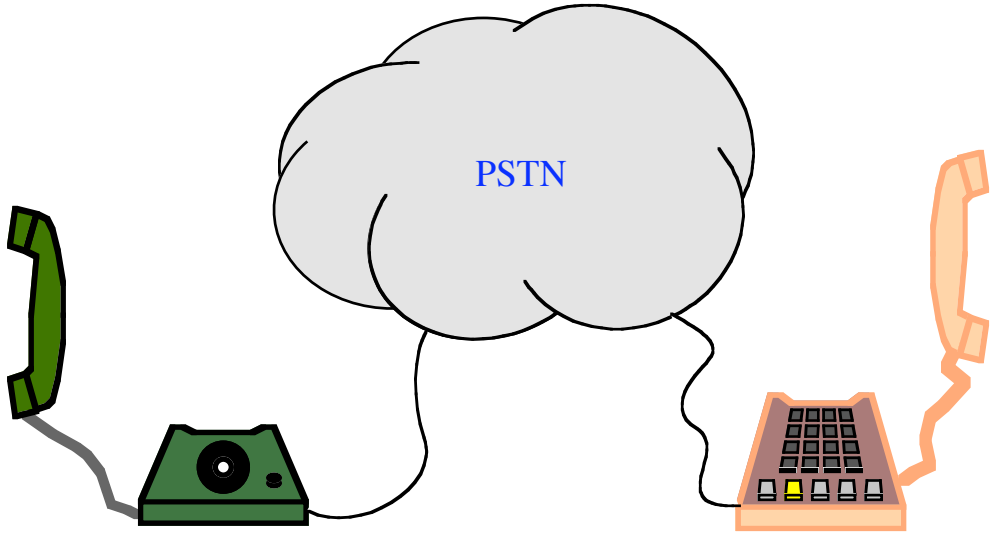


  - Punkt-zu-Punkt-Konferenz
  - Mehrpunkt-Konferenz
- Telepräsenz
- Unterricht
- Arbeitsprozesse (CSCW)



7.1 Kommunikation Person-zu-Person

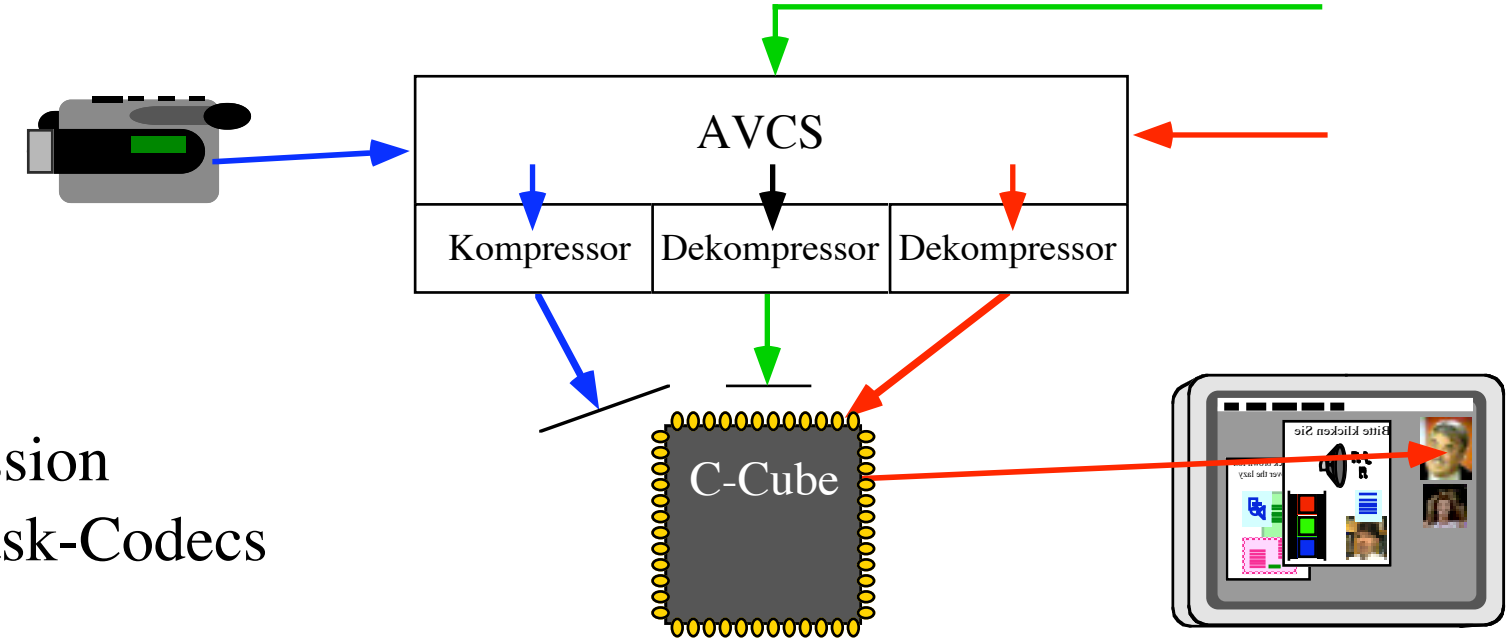
7.1.1 Telefondienst



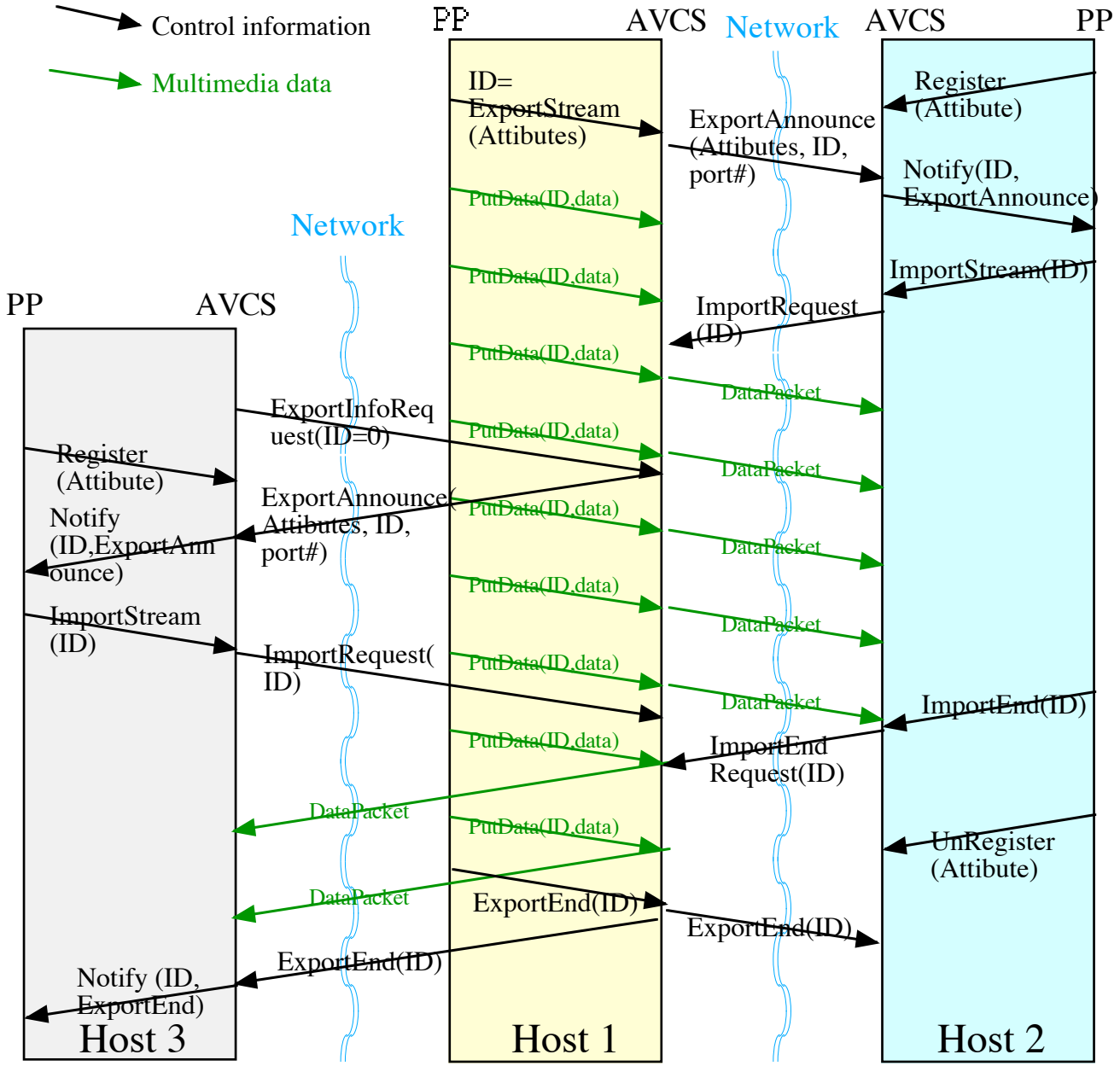
- 10^9 Teilnehmer?
- Endgeräte standardisiert
  - Mikrofon
  - Lautsprecher
  - Bedienprozedur
- Netzwerk heterogen

7.1.2 CIO-Phone

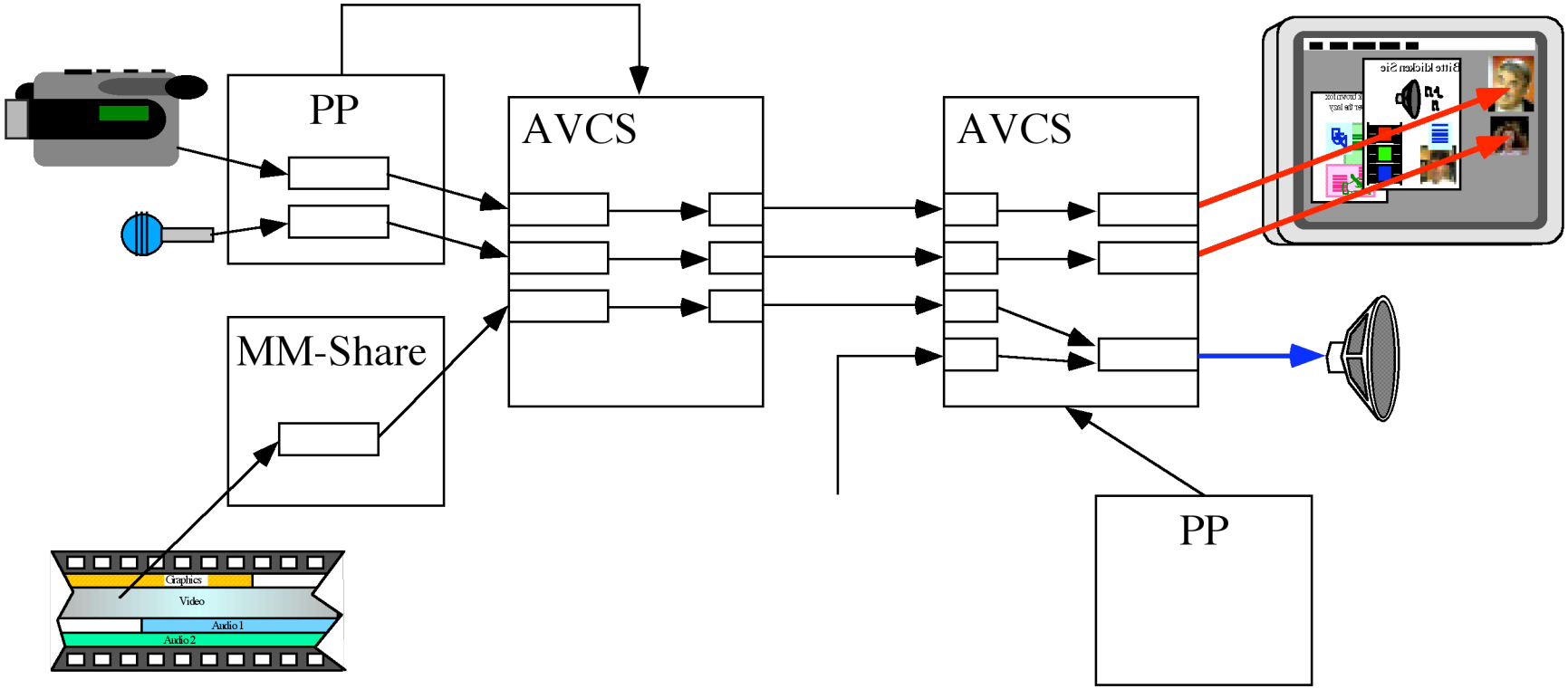
- Einfache Videotelefonapplikation
- Audio-Video Communcation Service
  - für VideoPhone und Multimedia-Ströme der Applikationen
  - UDP-basiert
  - Punkt-zu-Punkt
  - eventuell Datenreplikation
- AVCS: Echtzeitorientiert
  - kein Delay absichtlich eingefügt
  - Pakete rigoros verwerfen:
    Ankunft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



• AVCS Verbindungsmanagement



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



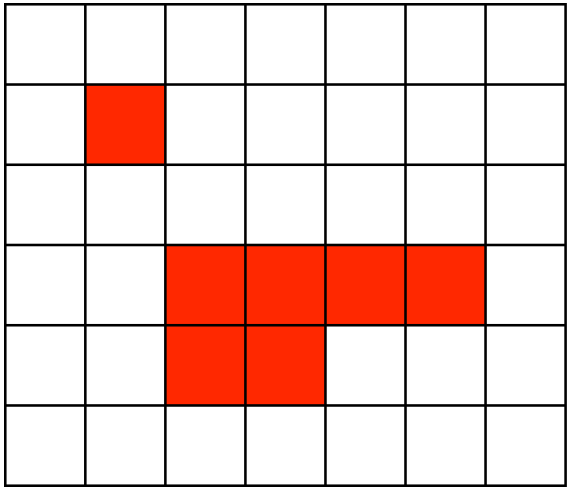
- Verzögerungen bei der Datenmanipulation
- Senden
  - PP öffnet Digitizer
  - PP bekommt Daten
  - PP übergibt Daten an AVCS
  - AVCS -> Ethernettreiber
- Empfangen
  - Ethernettreiber empfängt Paket
  - wartet auf Paketende!
  - Ethernettreiber -> AVCS
  - AVCS dekomprimiert
  - AVCS mischt
  - AVCS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

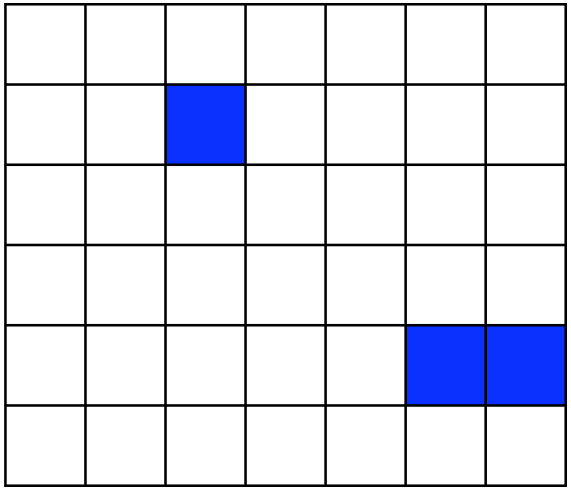
7.1.3 vat

- 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

7.1.4 vic

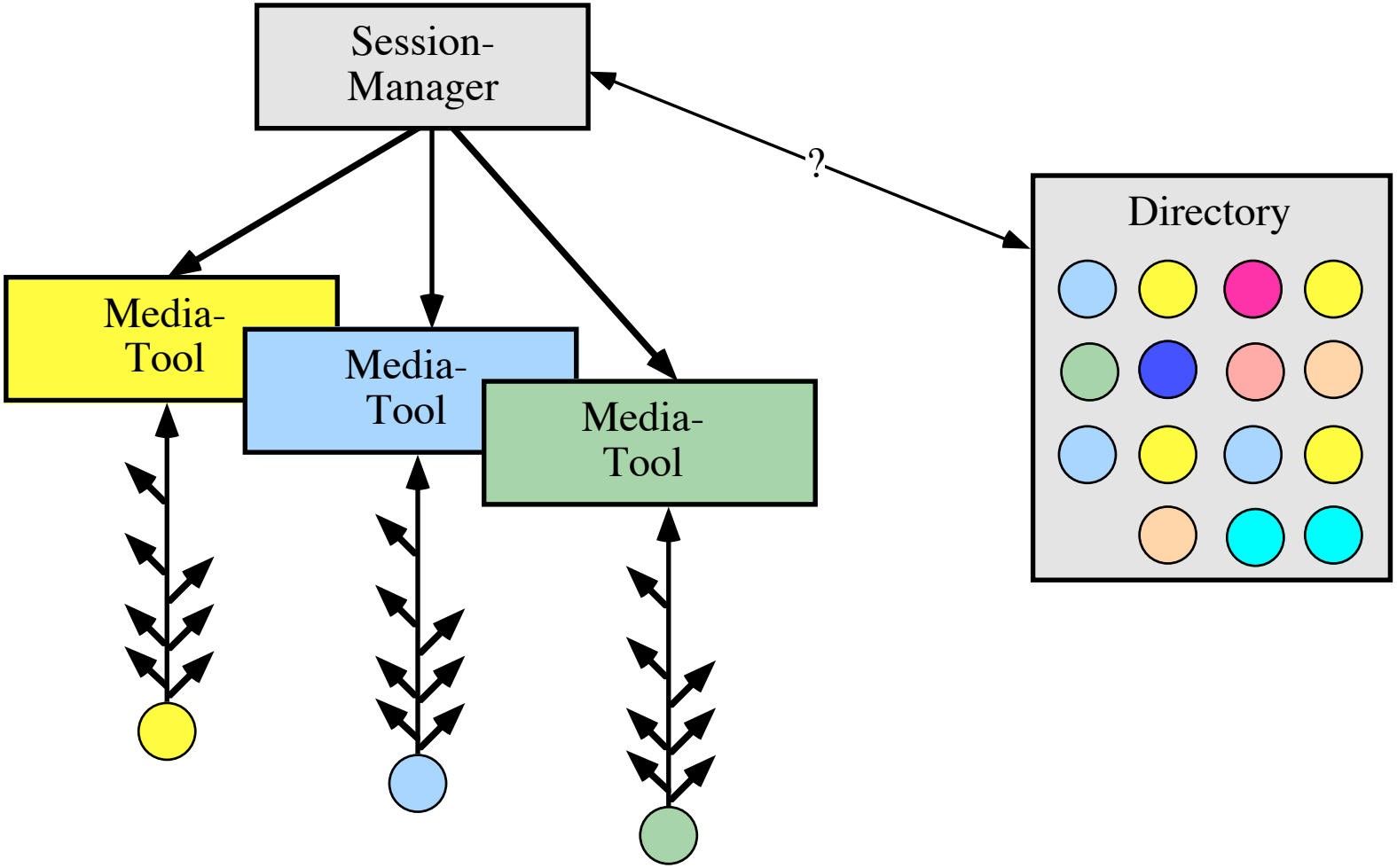
- Video conferencing
- Video-Formate
  - M-JPEG
  - Cell-B
  - nv
  - Intra-H.261
- Intra-H.261
  - Conditional replenishment: nur Änderungen
  - H.261 Mechanismen, um Blöcke zu überspringen
  - keine Differenzkodierung (intercoded blocks)



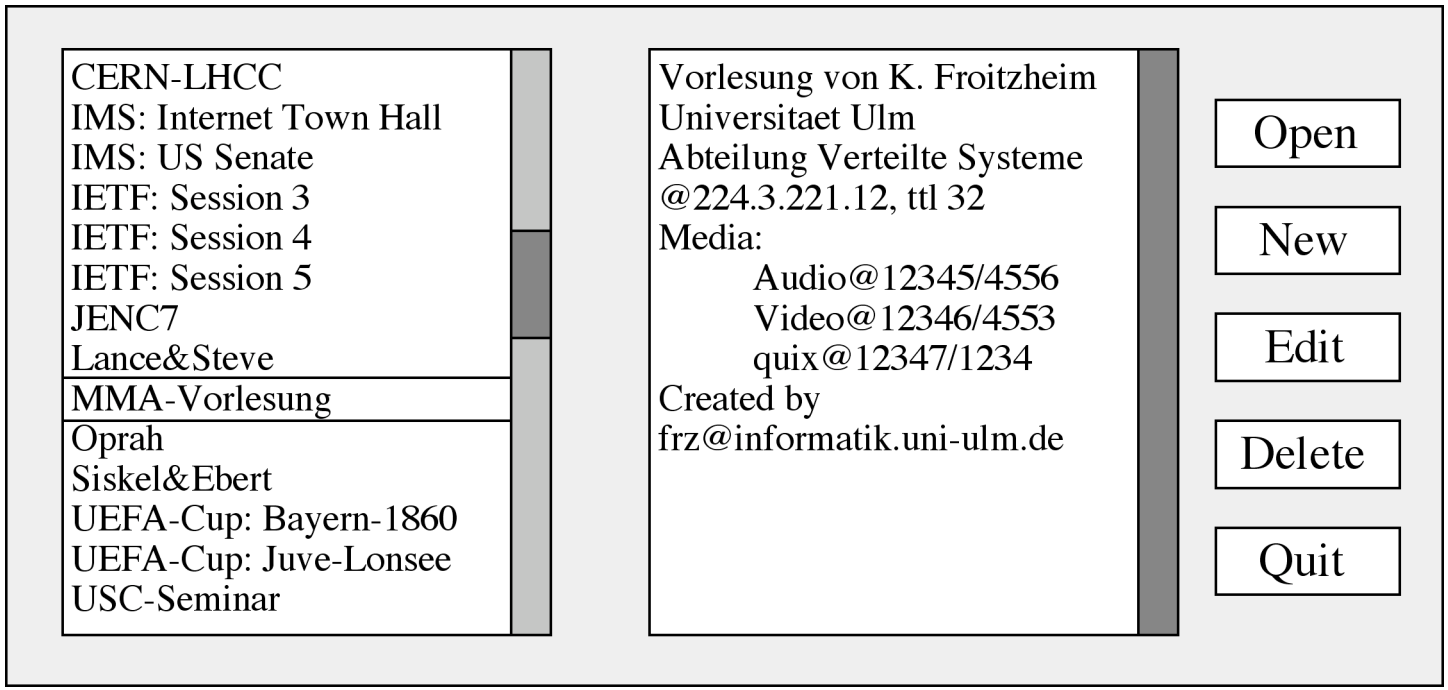


7.1.5 Management der MBone-Tools

- Multicast ≈ Rundfunk
- Media-Tools sind eigenständige Programme

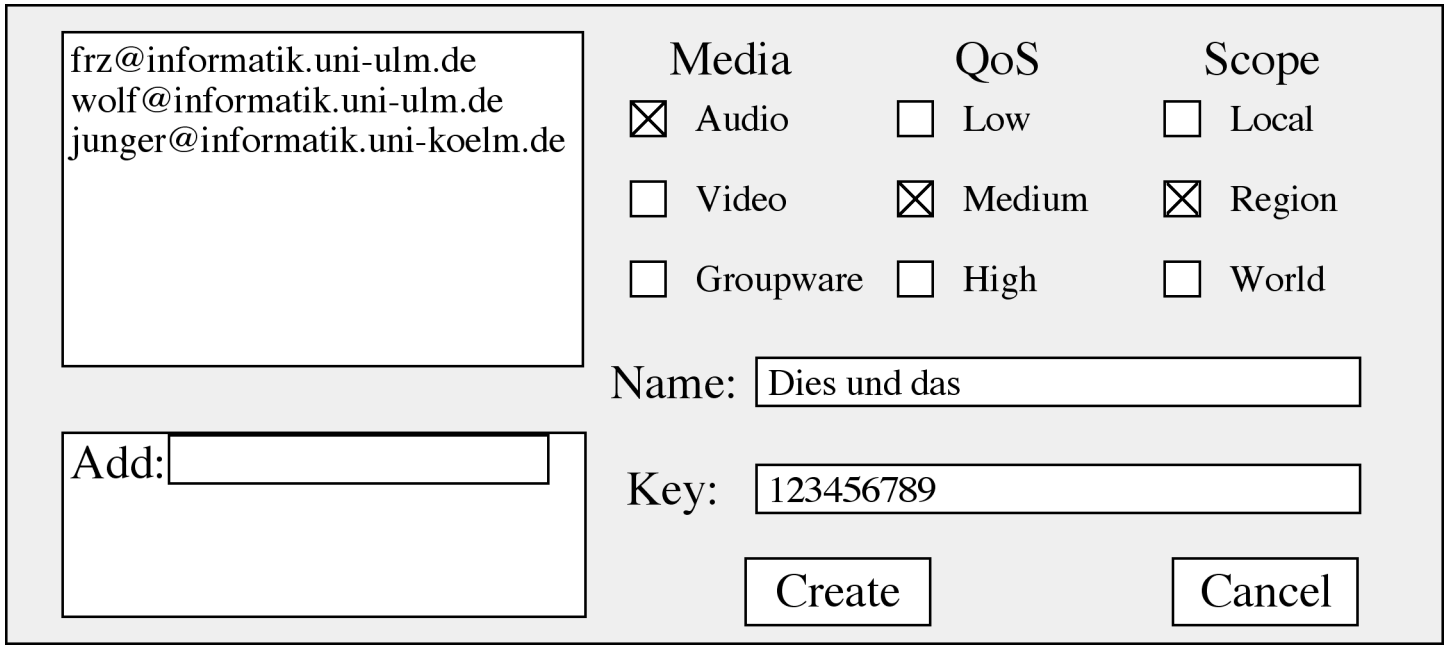


- sd: Session Directory
- Ankündigung von Konferenzen
  - Multicast-IP-Nummern-Vergabe
  - Strombeschreibung: port, ID, scope, Medium, Kodierung
  - Kommentare
- Programmzeitung als Paradigma
- Import: Start des entsprechenden Tools



- 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*



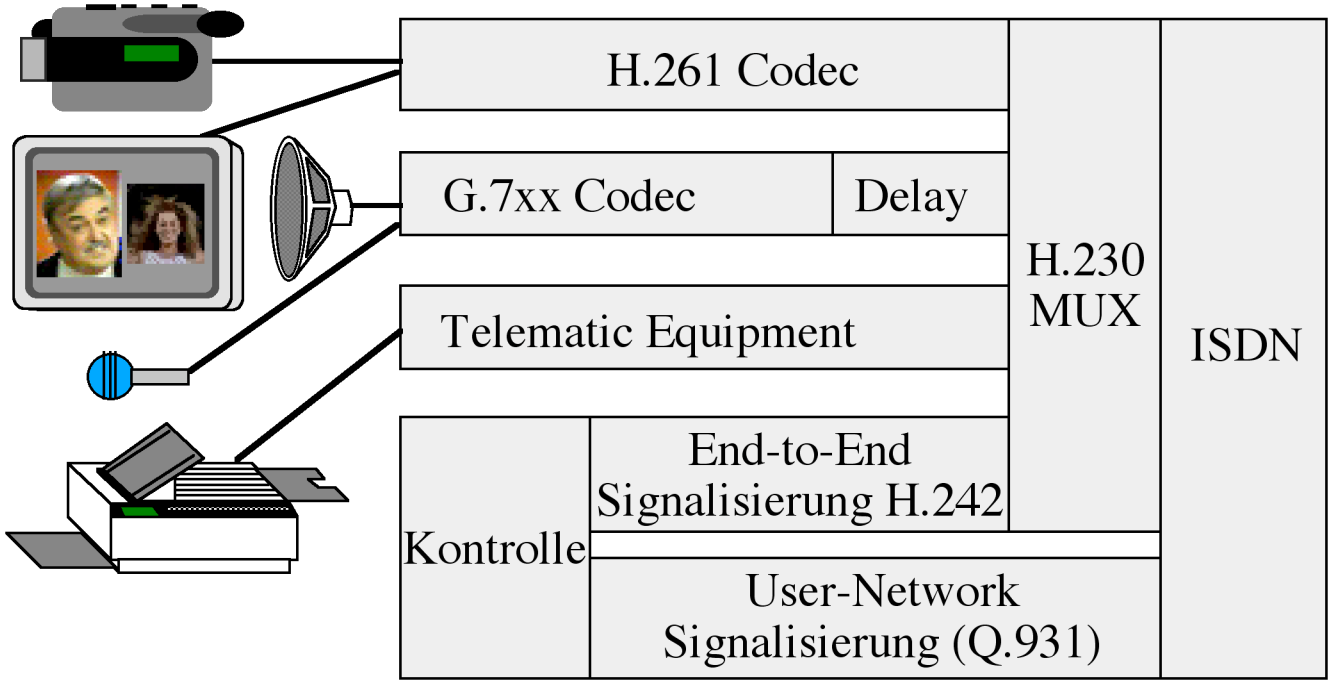
7.1.5 Kontrolle der Mehrpunkt-"Verbindung" im Mbone

- 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

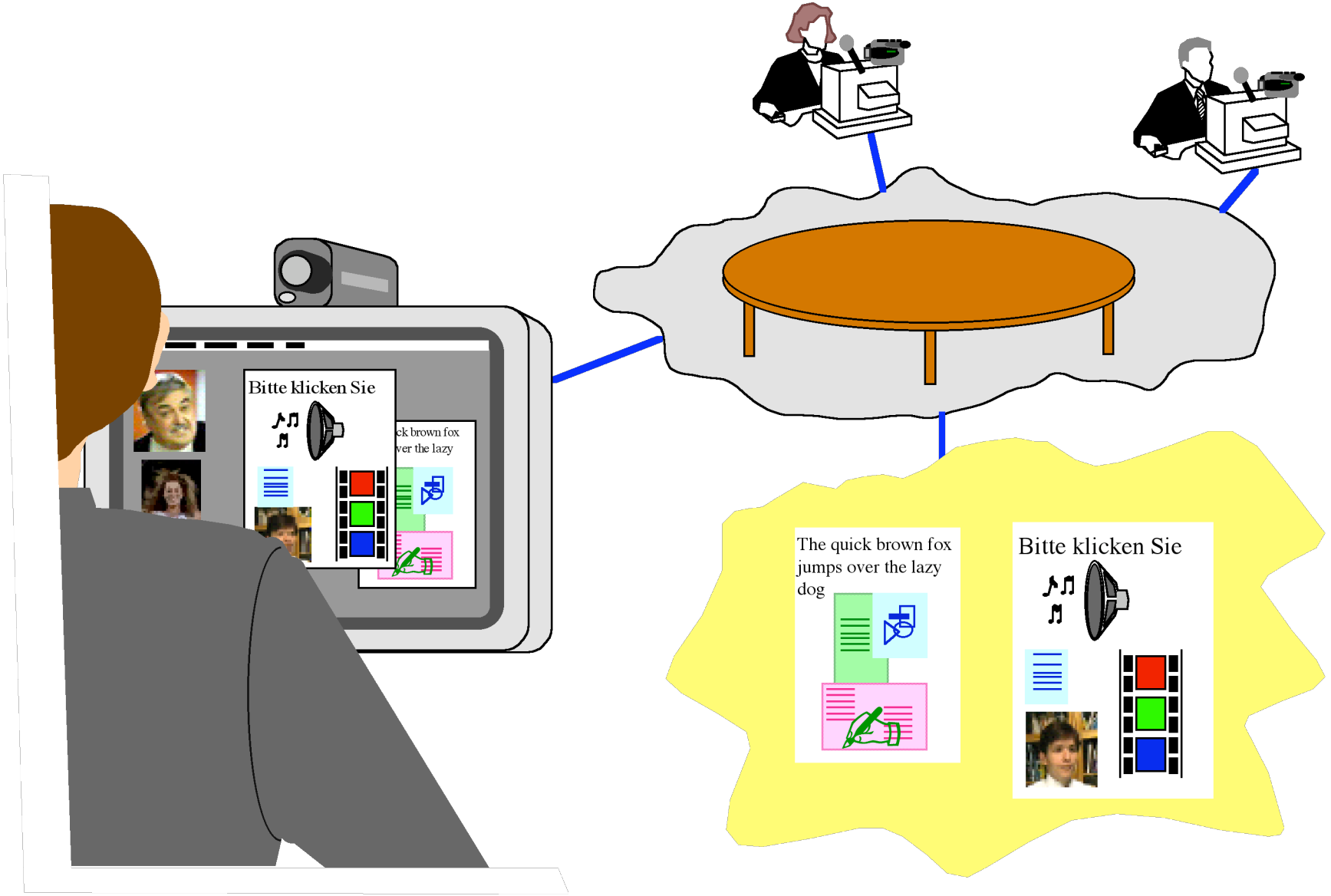
7.1.6 H.320 Videotelefonie

- ITU-Standard
- Zusammenfassung existierender Standards



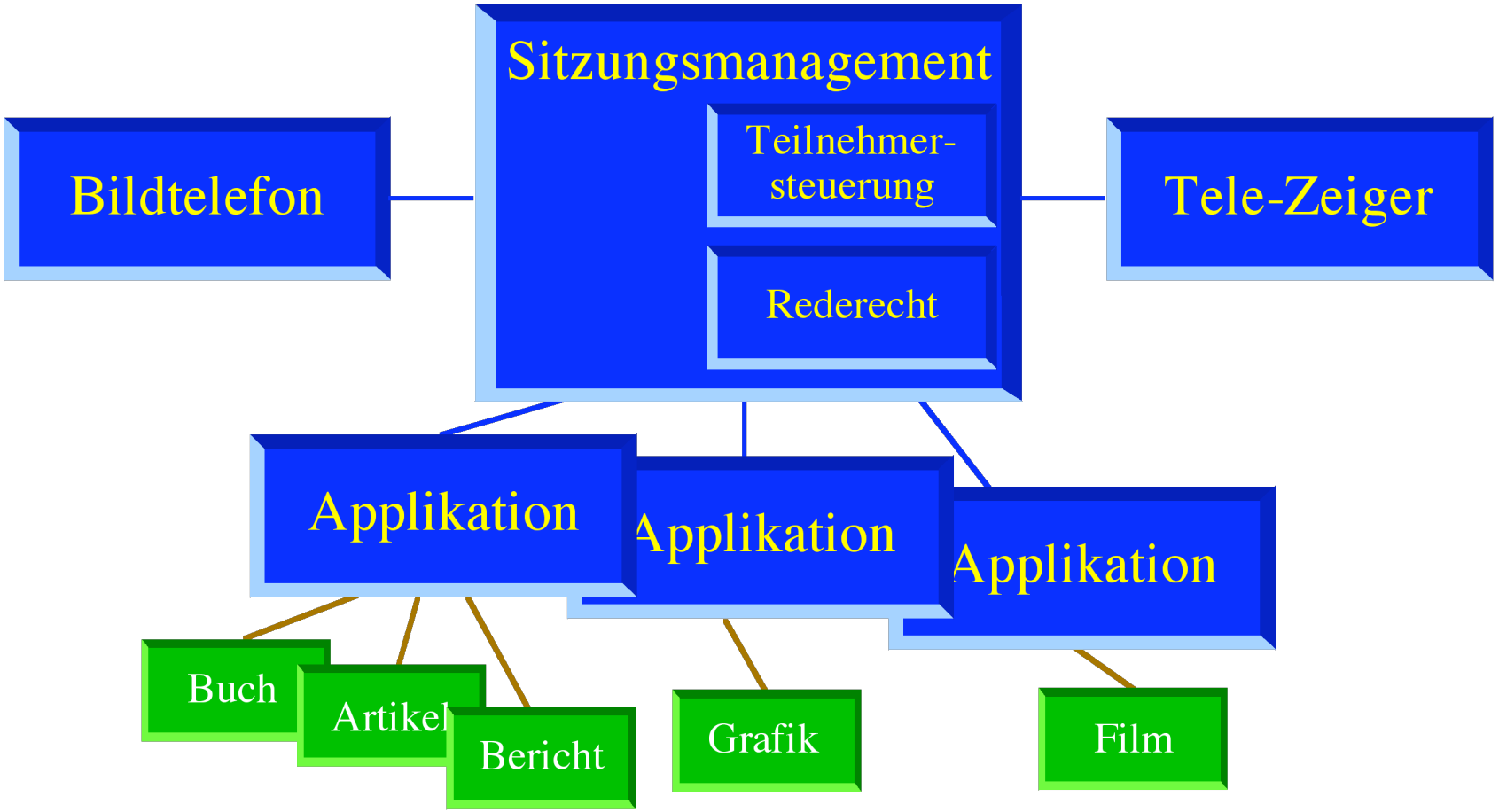
- H.320 ist Rahmenspezifikation
  - End-to-End Signalisierung (H.242)
  - Multiplexen von Audio und Video: H.221
  - Multiplexen von Audio,Video und ITU-Daten: H.230
- Erfolgreich zur Verbindung von Konferenzstudios

7.2 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

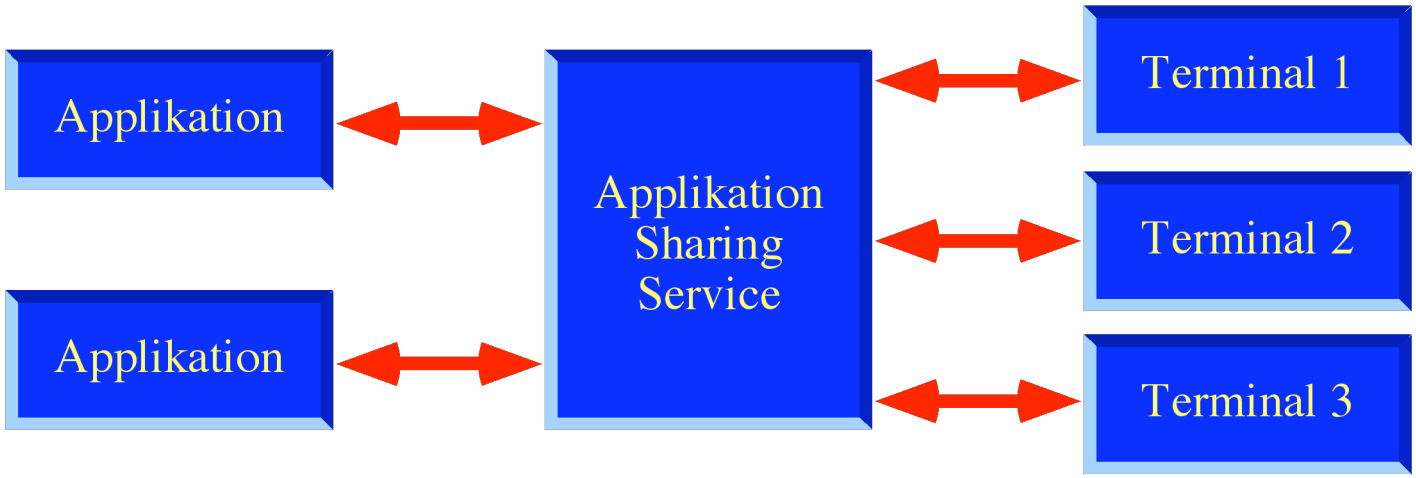
	Konferenzsystem	Benutzer
Hardware	homogen	heterogen
Betriebssystem	homogen	heterogen
Applikation	Groupware	Einzelplatz

- 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

7.2.1 Applikation-Sharing

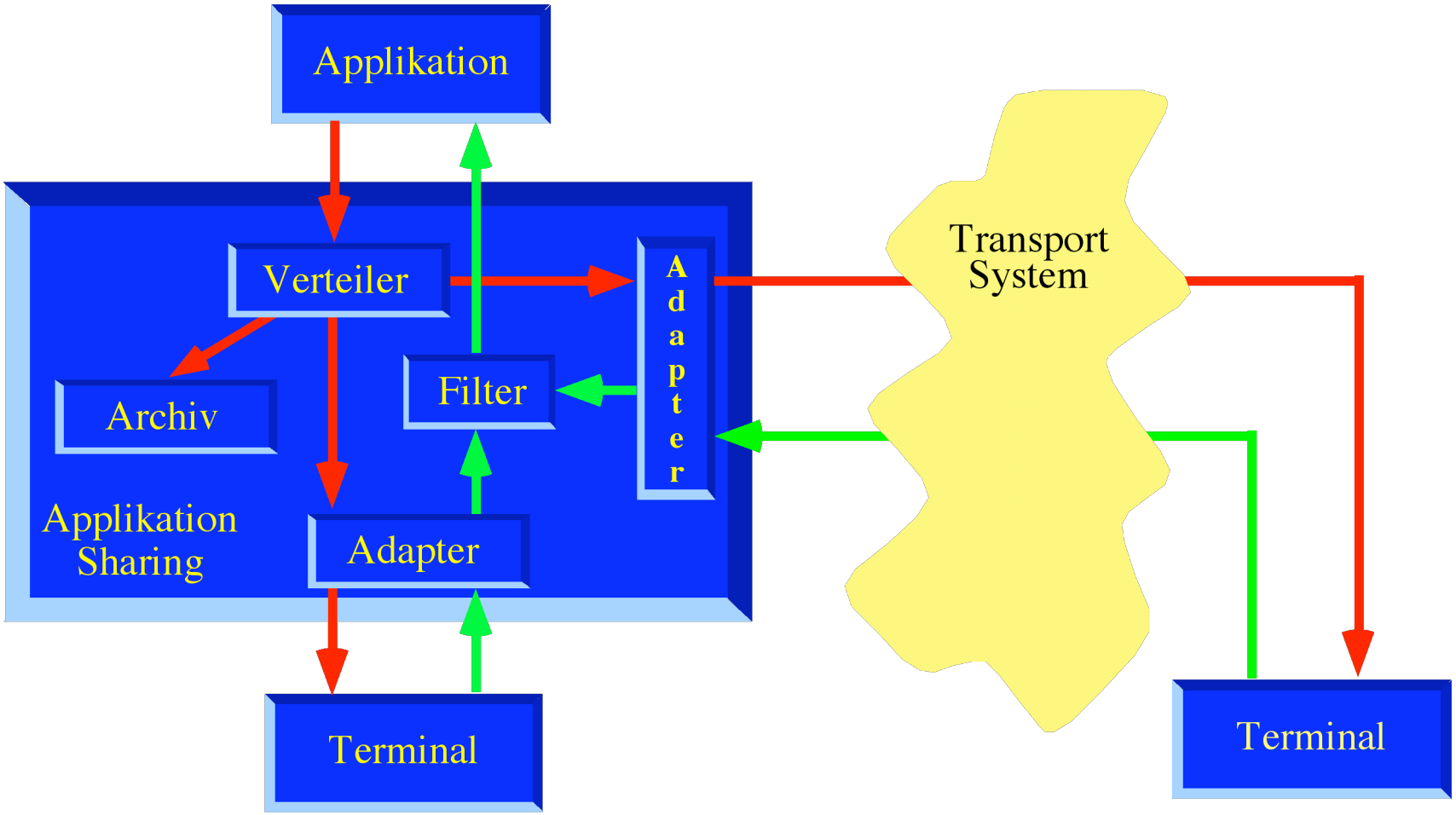
- Dienst: Vermittlung zwischen Programm und Terminal



  - Timbuktu [Epard; 1985]

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

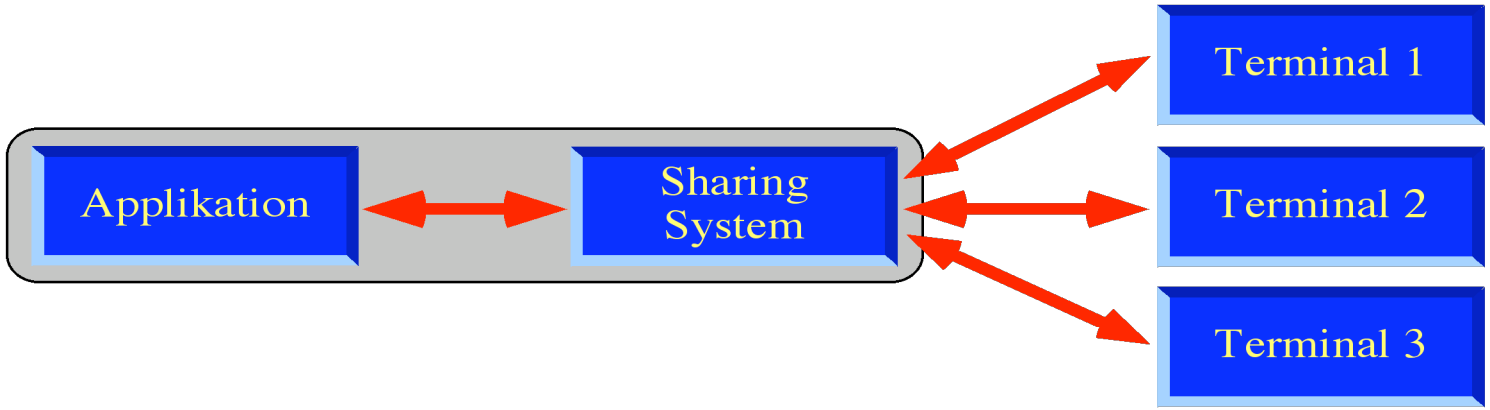
- Funktionen des Sharing - Dienstes (homogen)



7.2.2 Sharingkonzepte

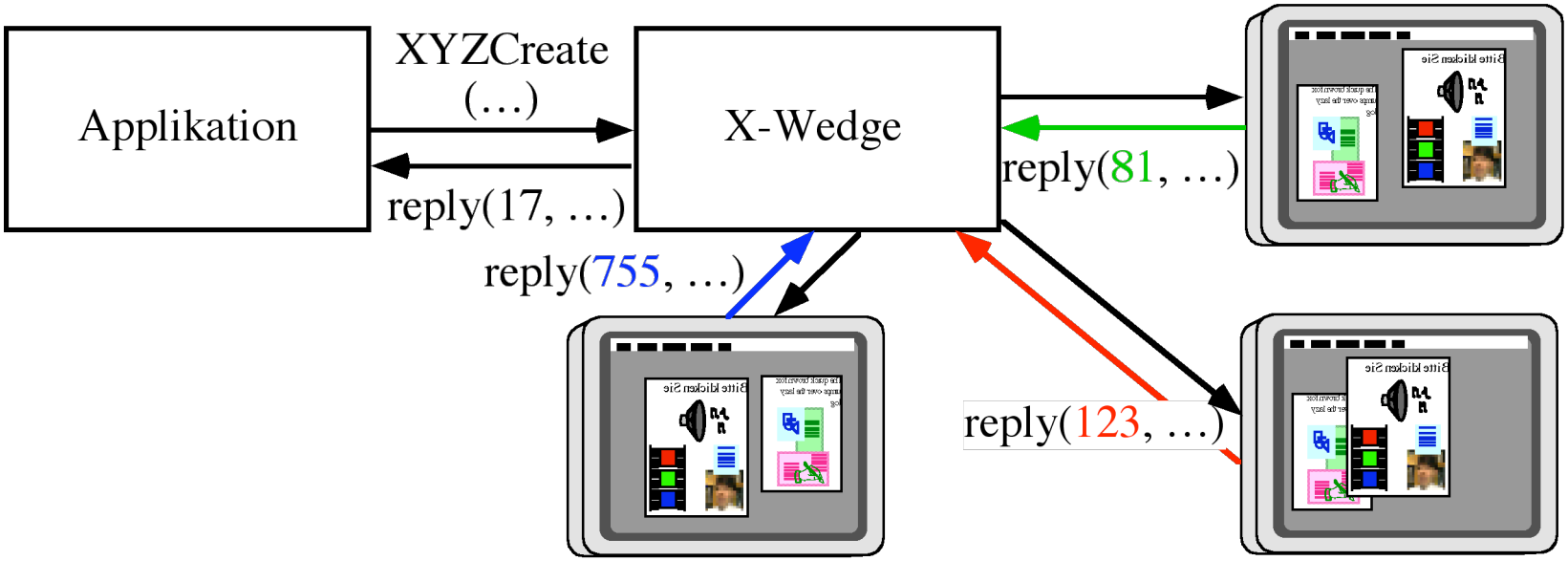
7.2.2.1 Verteilung der Grafikinformation

• Zentrale Architektur



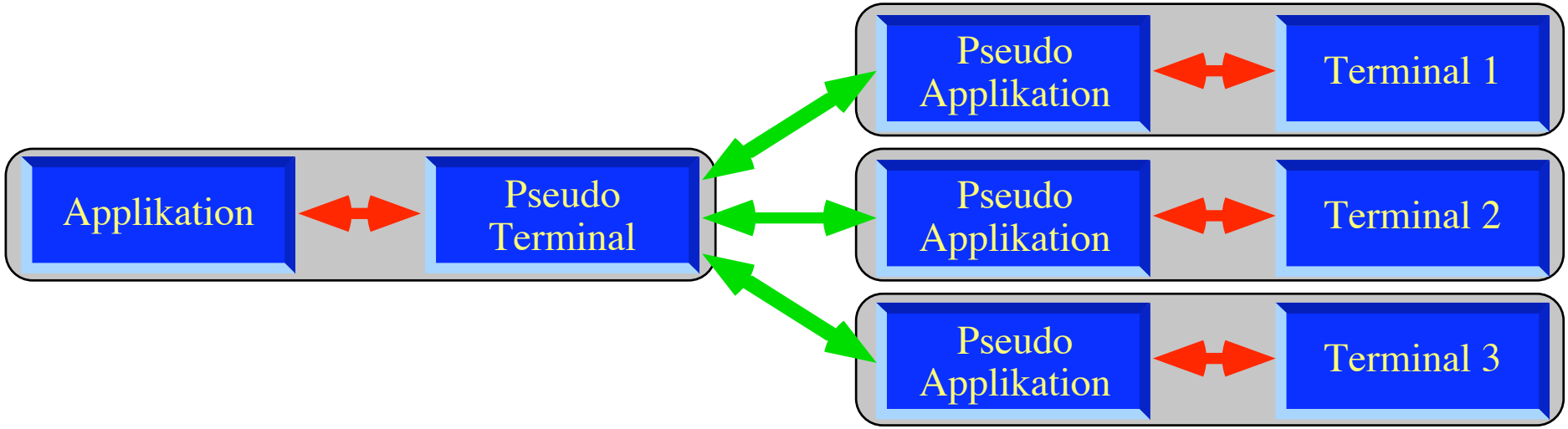
- Standardprotokolle
- Besonders geeignet für X Window
- XTV [Abdel-Wahab], SharedX [Altenhofen]

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

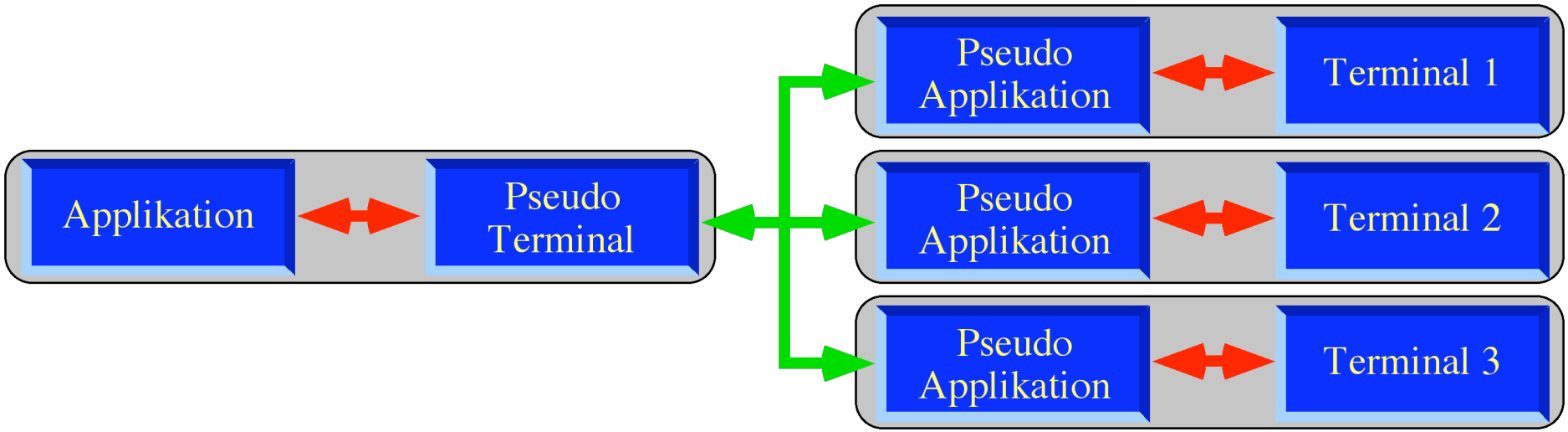


- Einheitliches Imaging-Modell
  - Farben
  - Fonts
- Bit-Order und Byte-Order
- Fensterüberlappung und Redraw konsolidieren

- Verteilte Architektur

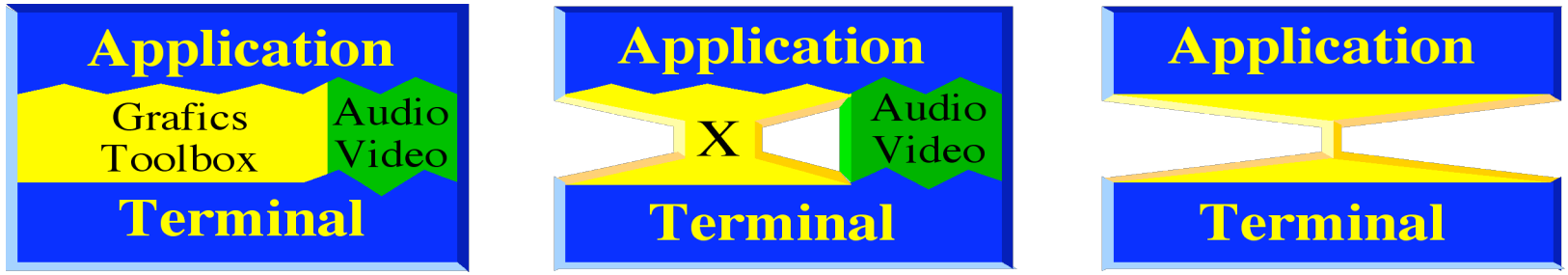


Eigene Verbindung zwischen Verteilungs-Komponenten
- 'eigene' Verbindung: Multicast, Verschlüsselung, Kompression
- X-Wedge [Gutekunst, CIO-JVTOS; 1994]



7.2.2.2 Redirector

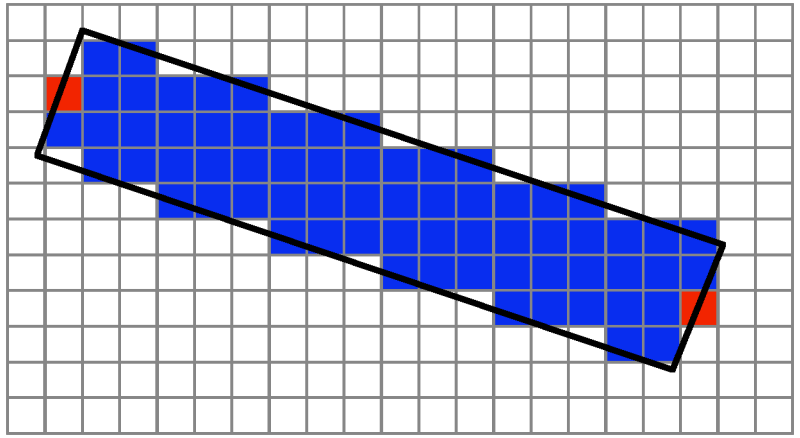
- Abhören von Grafikoperationen
- Events einfügen (Tastatur, Maus, Verwaltung)
- Interface Applikation-Toolbox-Terminal



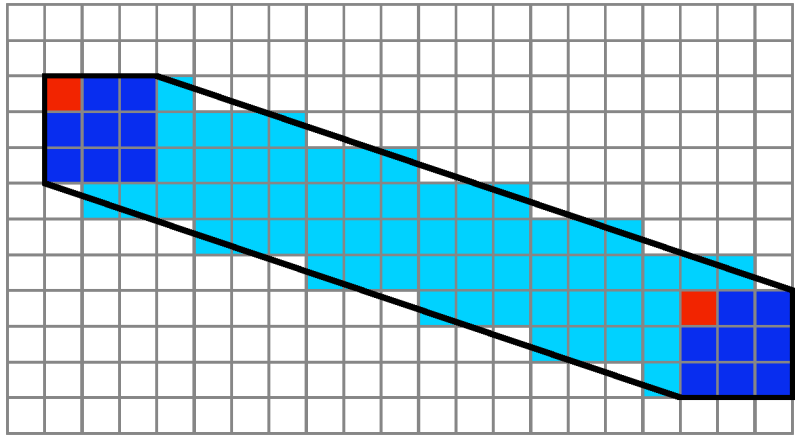
- Betriebssystemkomponente simulieren
  - toolbox
  - transport system
  - device driver
  - memory
  - low-intercept vs. high-intercept
  - Datenmenge vs. Prozedurmenge
  - Semantik

7.2.2.3 Übersetzung

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

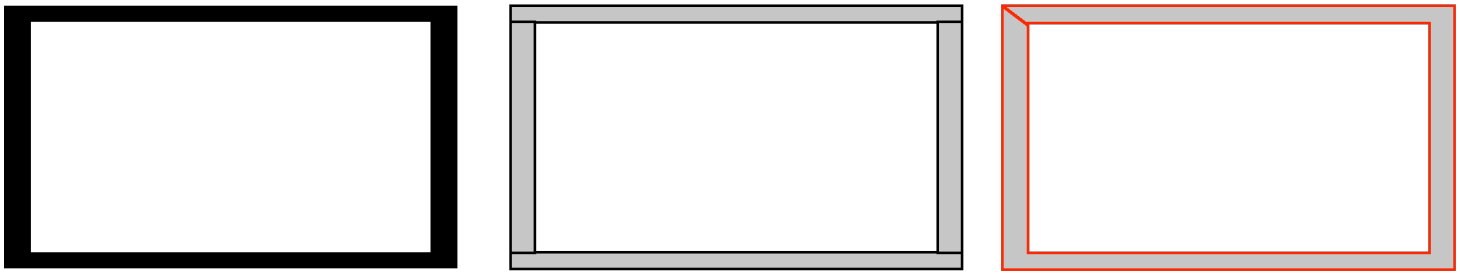


X-Window

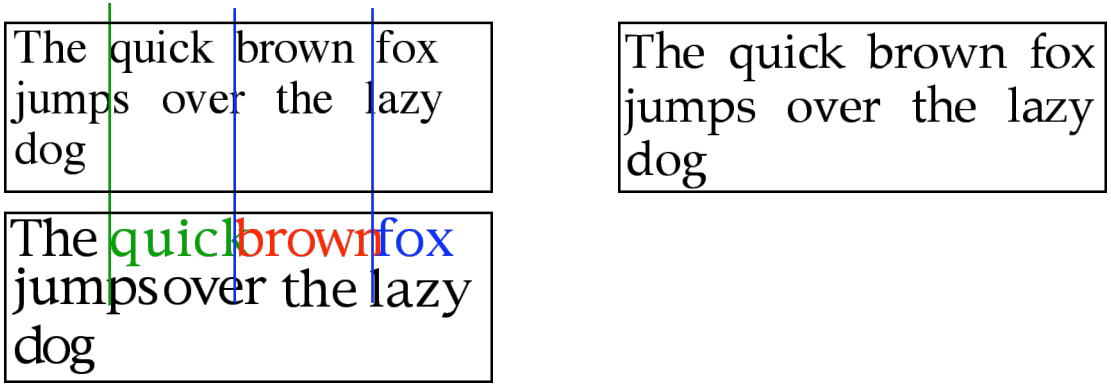


QuickDraw

- nicht-quadratische Stifte



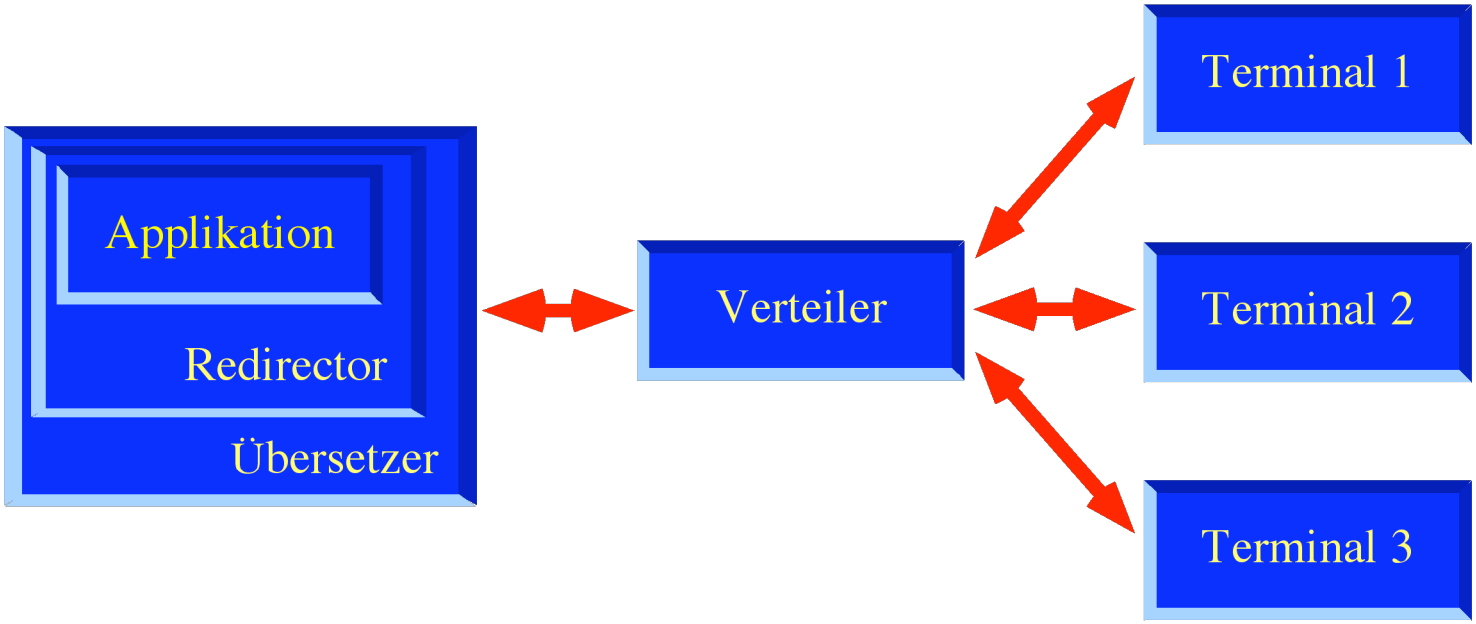
- Text
- Zeichensatzanpassung
  - Ä, Ö, Ü und ihre ASCII-Werte
  - Übersetzungstabelle
  - fehlende Zeichen (□)
- Fonteigenschaften
  - Zeichenhöhe
  - Zeichenbreite
  - Randausgleich



7.2.2.4 Anordnung der Komponenten

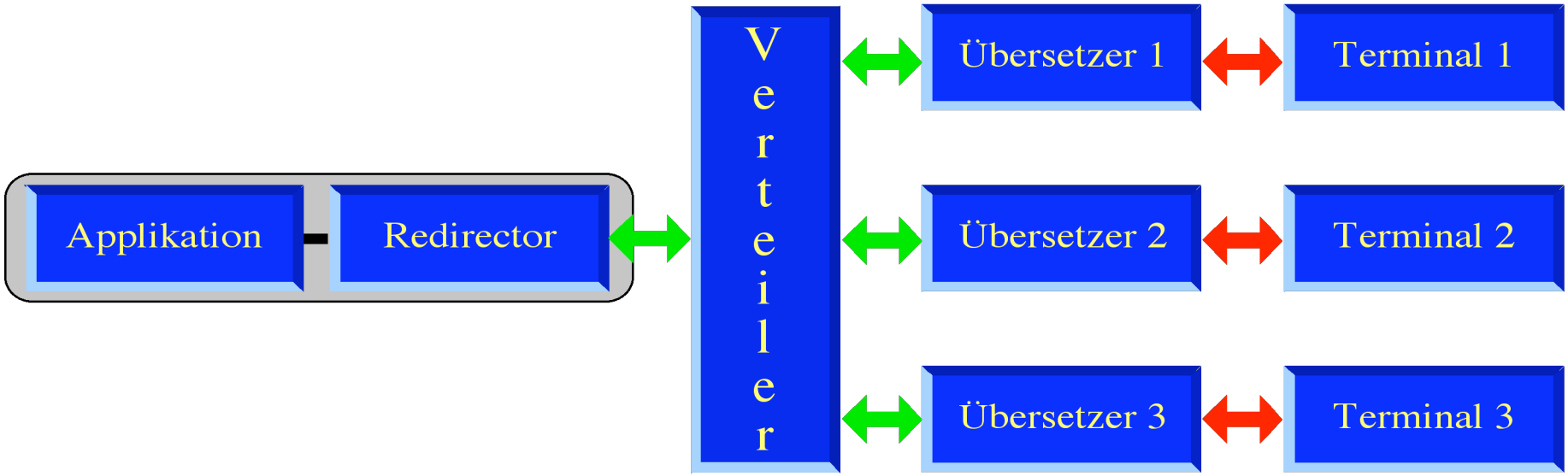
• Kombination mit Verteilung

Vor der Verteilung



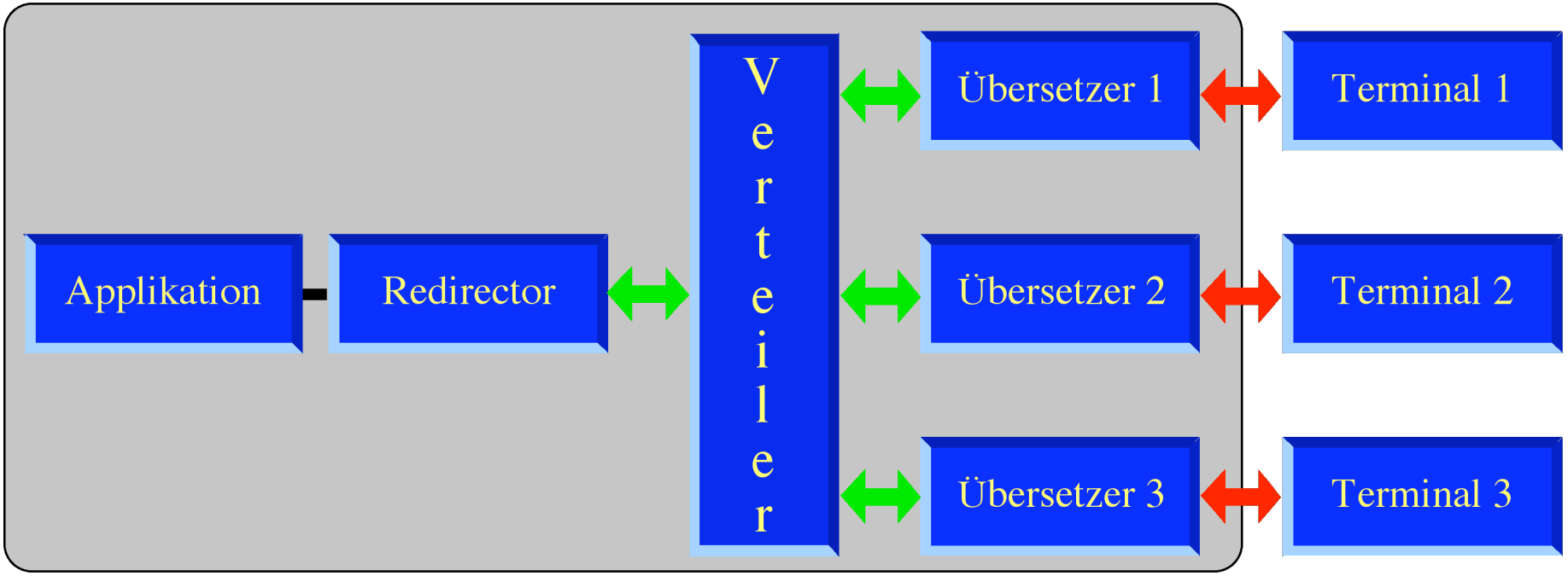
+ Komponentenwiederverwendung (QuiX + X-Wedge)
– Gleichbehandlung der Terminale (Farbtiefe, Zeichensätze, …)
– Struktur unflexibel

- Übersetzung nach der Verteilung



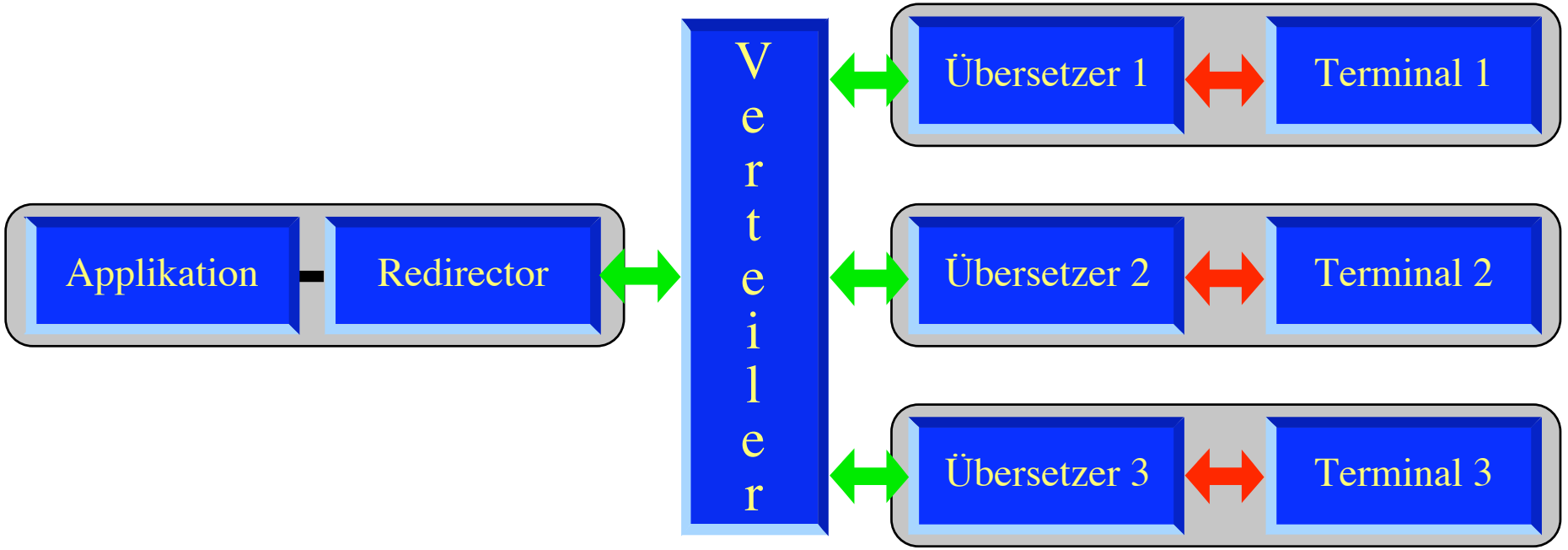
+ Flexible Verteilung der Funktionen
+ Konverter gut anpassbar an Terminaleigenschaften
  QuickDrawServer statt X-Server
  GDIServer statt X-Server
+ Objektorientierter Ansatz macht Verteiler einfach
– Mehrfache Übersetzung

- QuiX-M [Froitzheim, Wolf; 1995]



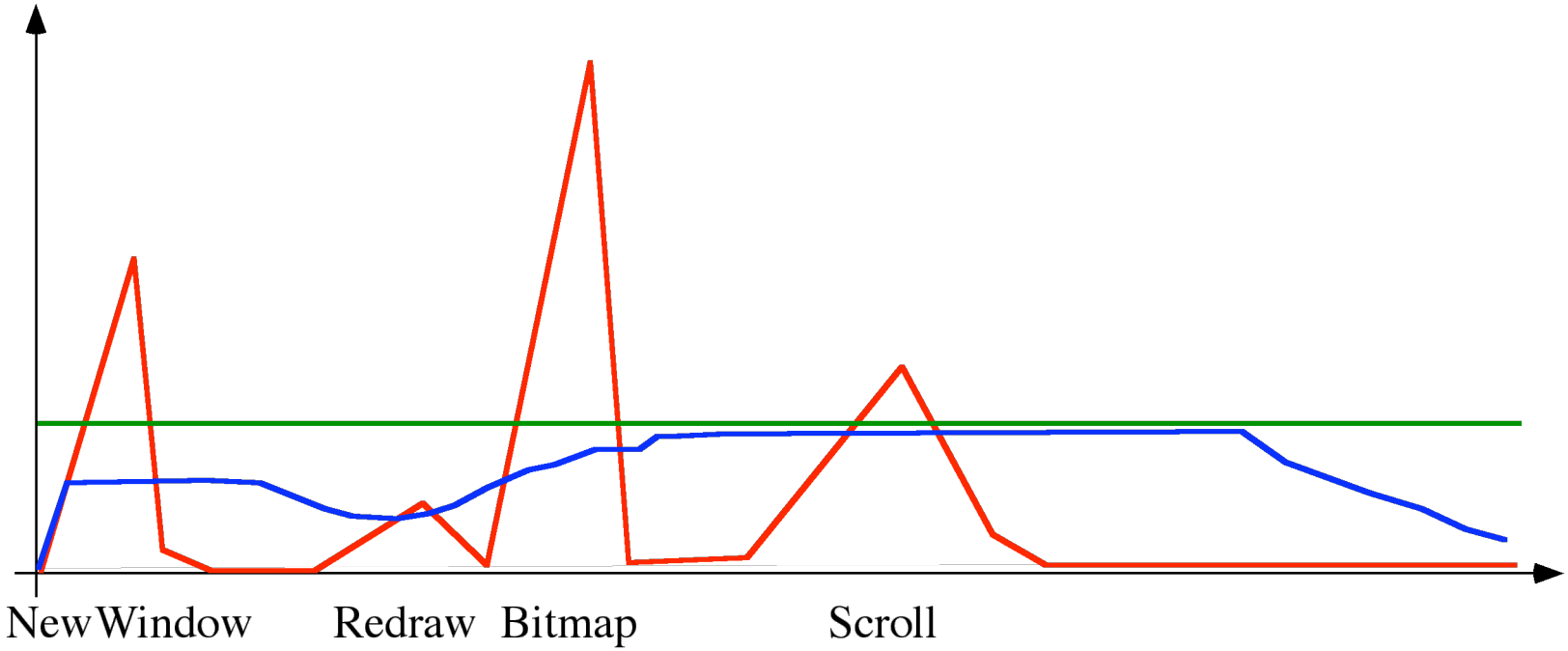
- Migration vom einfachen QuiX
  - Instanziierung des Übersetzers
  - Verteiler bietet nach oben Übersetzerschnittstelle
  - Seiteneingang zur Verteilungs-Steuerung
- Performanceprobleme
  - X-Server
  - Übertragung, Pufferstrategie, etc.
  - ab 1:4 Übersetzung

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



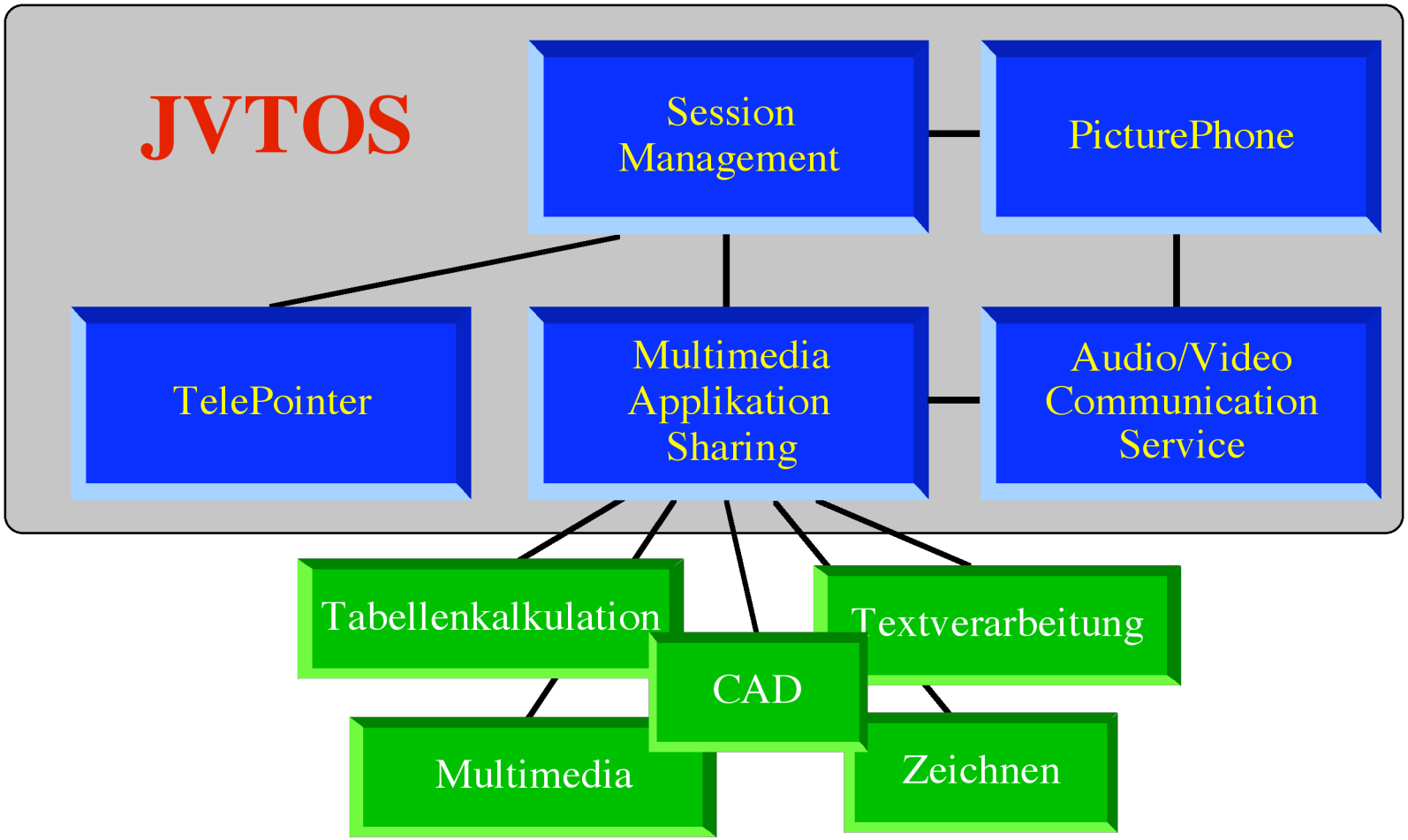
- Mehrpunktscenario
  - MBone, UDP
  - einfache Fehlerkorrektur (r=0,5) mit Bitpaar-Manipulation
    Q1 = P1; Q2 = P2;
    Q3 = P1+P2; (+ aus XOR)
    Q4 = P1 • P2 (• aus + und *; * Tabelle)

- Traffic-Shaping notwendig



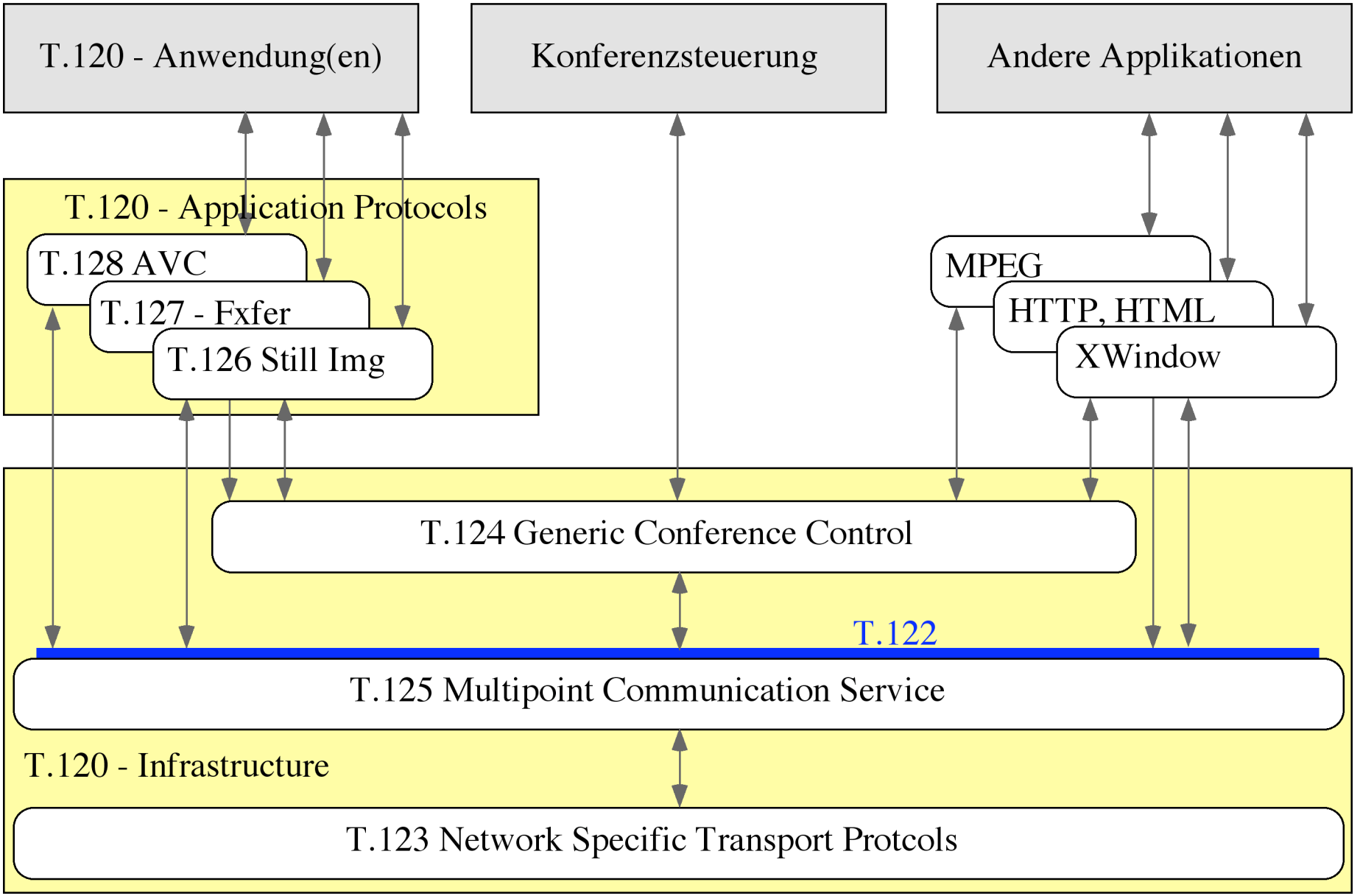
- Anständige Programme
  - Adobe PhotoShop, Premiere, Tools und Utilities
- Feinde
  - Menus, Dialogboxen, Popup-Menus, Scroller
  => Redirection an der Toolboxschnittstelle
  - DIY-Toolbox (Microsoft, Claris)
  - direkter Aufruf von OS-Routinen
  - Framemakers 'Turbo-Text'

7.2.3 Joint Viewing and Teleoperation Service JVTOS



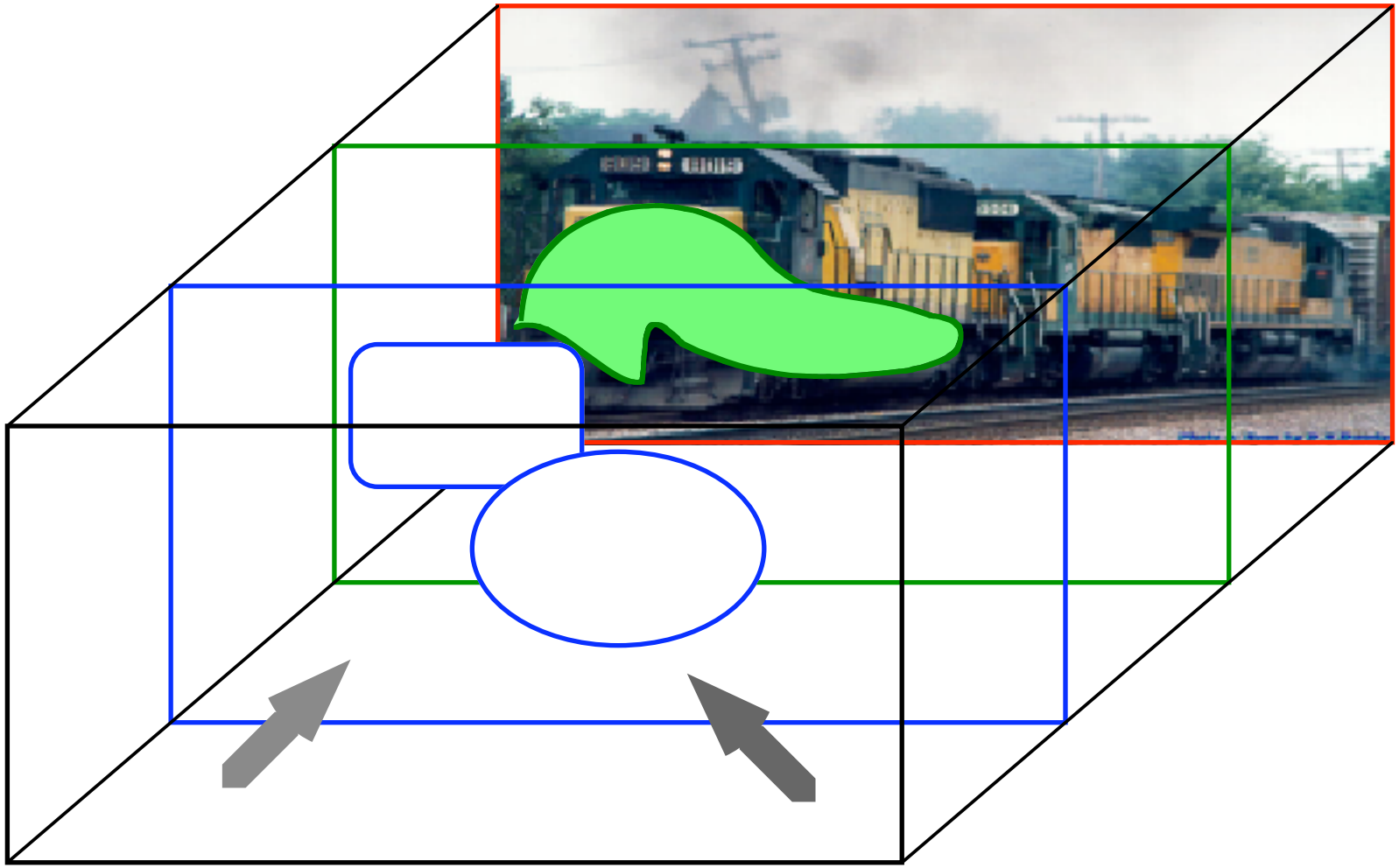
- RACE Projekt 2060, CIO
  - CET, ETHZ, Intersis, Siemens, Uni Stuttgart, TU Berlin, Uni Ulm

7.2.4 T.120: Videokonferenzstandard der ITU



- T.124
  - Abstraktion Konferenz
  - Create, Invite
  - Join, Leave
  - Application-Keys
- Mehrpunkt-Kommunikation
  - Interface in T.122, Protokoll in T.125
  - zuverlässiges Mehrpunktprotokoll
  - Kanal-Paradigma
  - globale Synchronisation zwischen Endgeräten
- T.123: Transportprotokolle
  - PSTN, ISDN
  - CSPDN, PSPDN
  - IPX, TCP/IP
  - nach oben X.214/X.215
- T.126 Still Image Exchange and Annotation
  - Bitmaps
  - einfache Grafik zur Annotation
  - Telepointer

- Workspace
  - 1 bis 256 Ebenen
  - obere Ebenen überdecken untere
  - Ebene entweder für Bitmap oder Annotation
  - virtual pointer plane



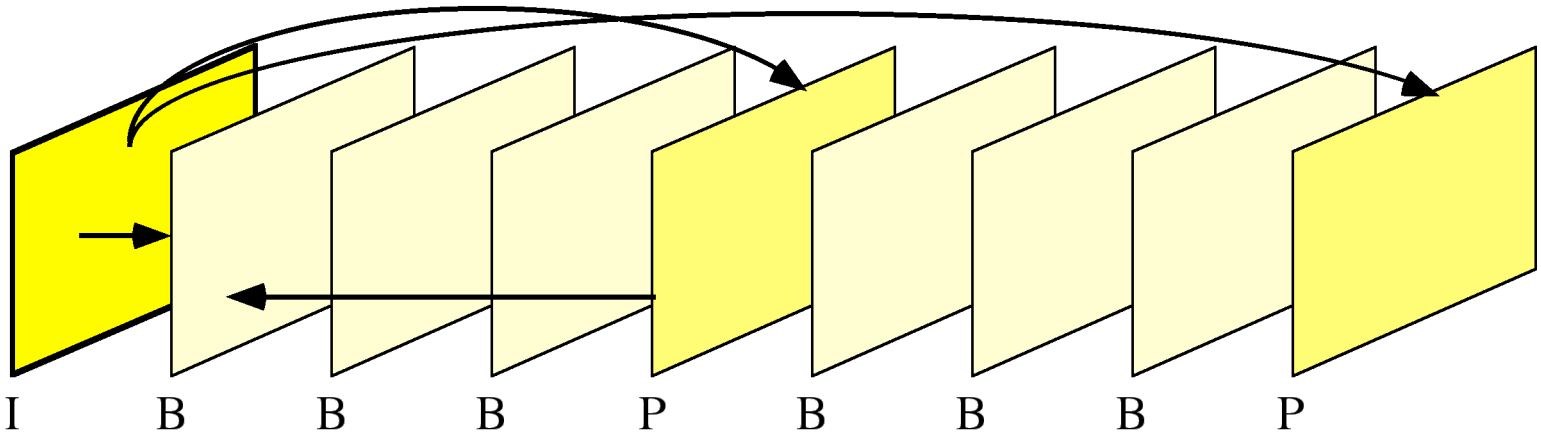
- Bitmapformate
  - Unkomprimiert
  - Fax G.3 (T.4), G.4 (T.6)
  - JGEP (T.81)
  - JBIG (T.82)
- Annotation planes
  - permanent: Grafik wird zu Bitmap
  - editable: Objekte bleiben erhalten
- Grafikbefehle
  - Punktsequenzen (Freiformlinien)
  - Polyline (offene Polygone)
  - Rechteck
  - Ellipse
- Grafikparameter
  - Stift: Form, Farbe, Dicke, Stil
  - Bounding Rect, Z-Wert, Rotationswinkel
  - sample-rate, Selektionszustand
- Events: Tastatur, Zeigegerät

7.3 Synchrone Kooperation im WWW

- kontinuierliche Medien
  - live: Audio, Video, Chat
  - Clips: Audio, Video, Animation
- Dateiübertragungsparadigma
  - Übertragen, Speichern, Anzeigen
  - Helper-Applikationen
  - Streaming-Decoder für GIF
  - Plugins für Quicktime, Shockwave
- WWW => http => TCP
  - Fehlerkorrektur
  - Vegas-Algorithmus
- Bandbreite knapp
  - Kompression
  - Verstopfung
- schlecht für Filme
  - Übertragungszeit >> Abspielzeit

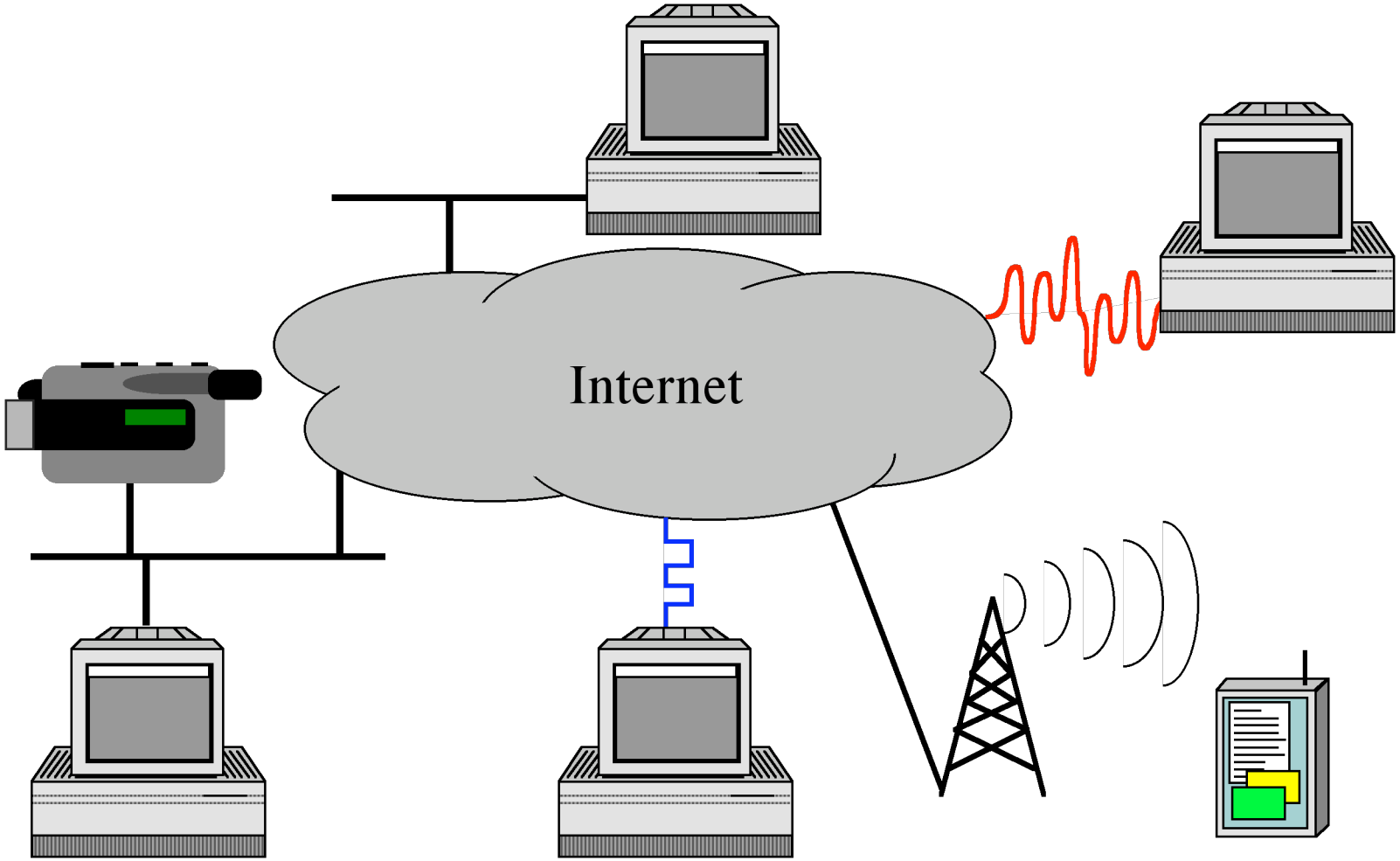
- Kompression und das WWW
  - MPEG, H.261
  - MUSICAM, ADPCM, GSM, LPC
- Kompression sehr rechenintensiv
- Filtern im Netzwerk
  - Frames, Farbe, Auflösung verringern
  - Stream dekoderien, De-Huffman
  - Frames entfernen, ohne Abhängigkeiten zu zerstören
  - Huffman, Stream bauen
- Beispiel
  - MPEG [Nick Yeadon et al, Lancaster University]
  - IBBPBBPBBPBBPBB -> IPPPP -> I
  - 1210 K 551 K 201K
  - 100% 45% 17%
  - B-Frames 55% => Film schlecht komprimiert
  - 1I, 4P, 10B (B = 66%)

- Datenstromskalierung am Beispiel MPEG
  - Farbe entfernen: max. 20%

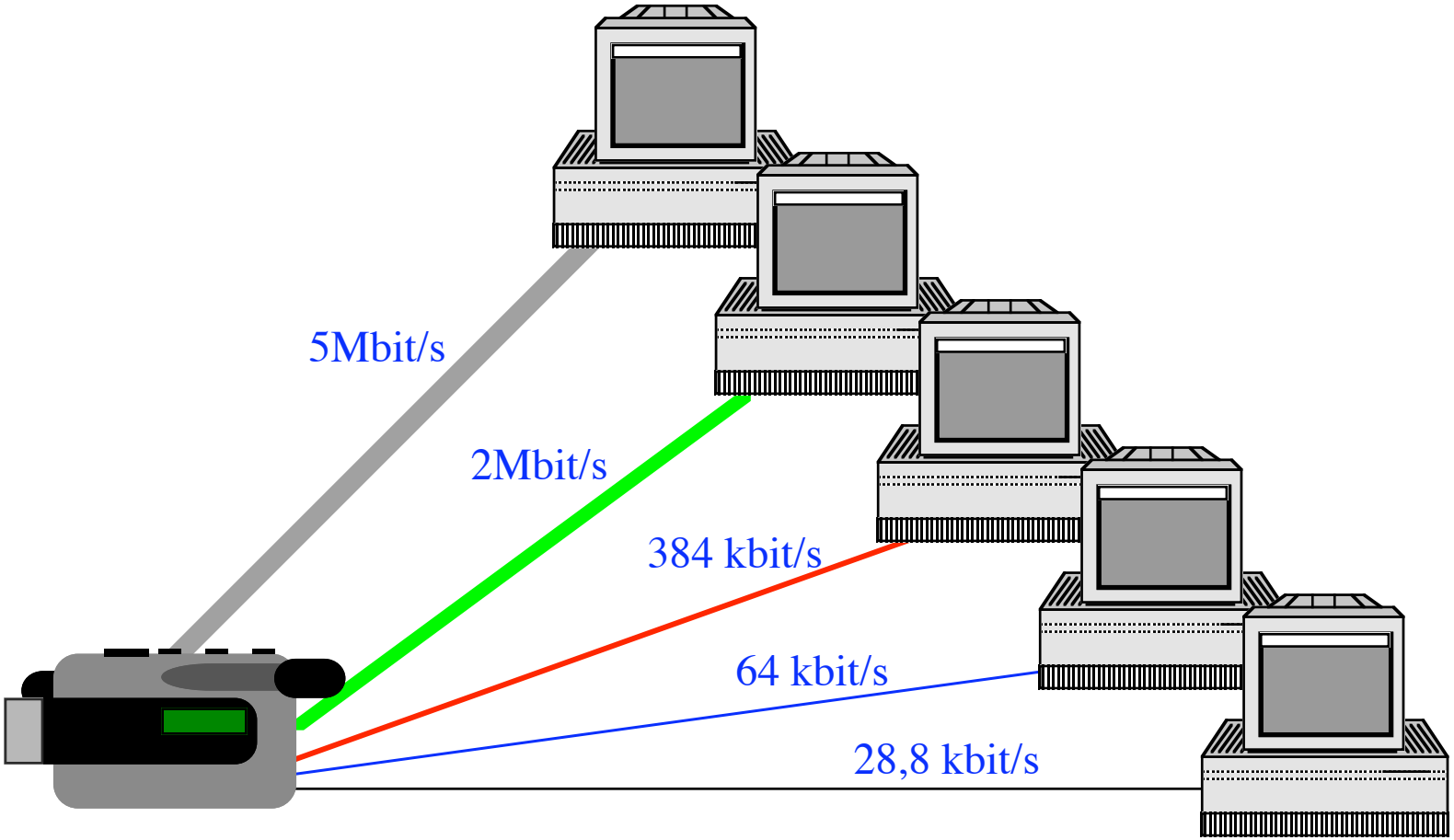


  - 1 Sekunde: 3*I, 4*P, 18*B, I:P:B = 40:30:30
  - Wiederholrate 7 statt 25 reduziert Datenrate um 30%
  - Resultat: schlecht komprimierter Strom
- Was ist eigentlich problematisch an MPEG
  - Vorwärtsbezüge
  - komplexe Bewegungsvektorsuche
  - räumliche Auflösung schwer zu reduzieren
  - nichttriviale Bewegungsvektoren sonderfall (Kameraschwenk)
  - Zoom, Drehung, Bewegung

- Mehrpunktszenarien im WWW und Internet
  - Fernsehen, Rundfunk, Konferenzen
  - Endgeräte
  - Netzwerke



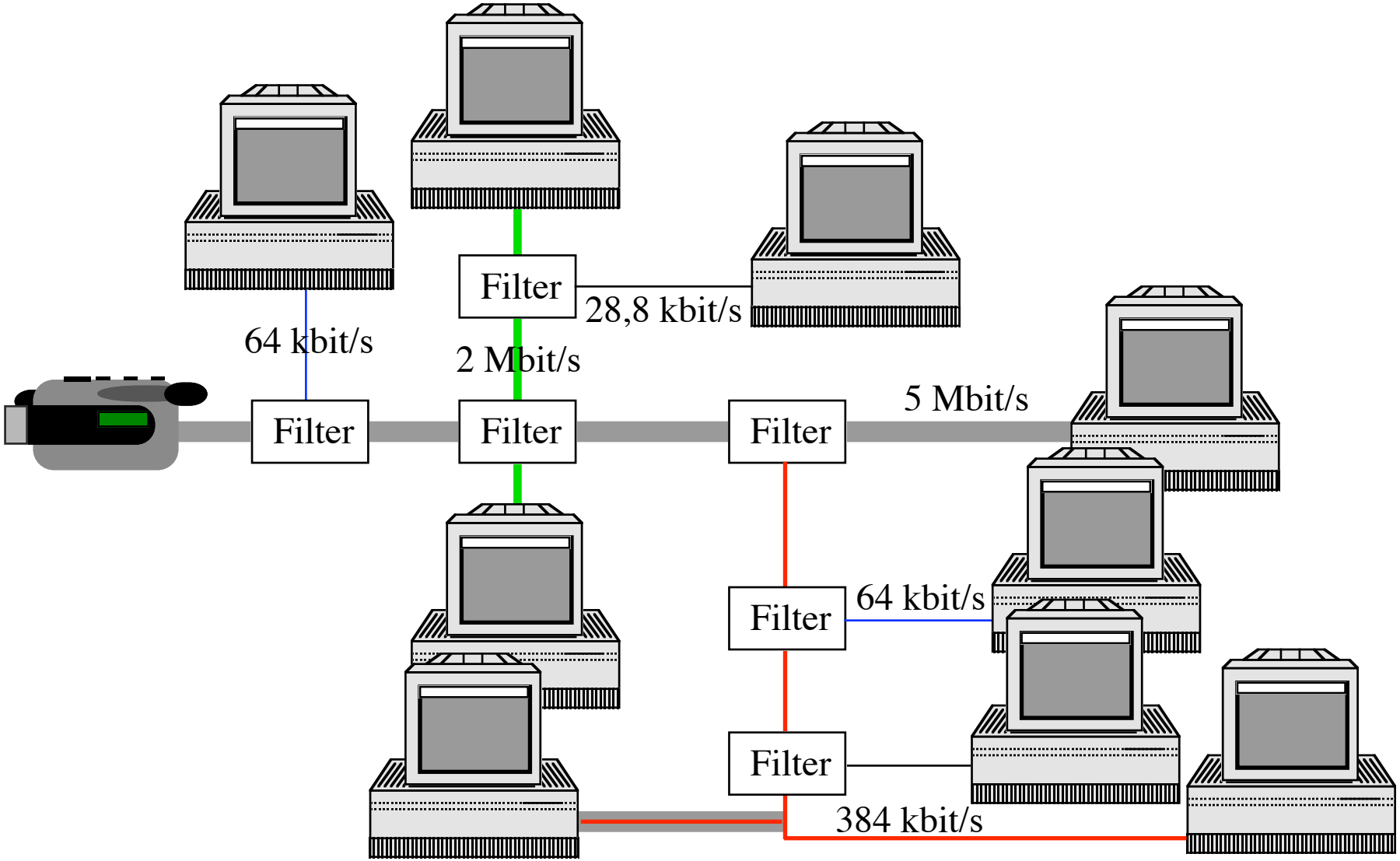
• Internet ist durchsatz-heterogen



• Echtzeitpräsentation: Leitungen dürfen nicht 'verstopfen'

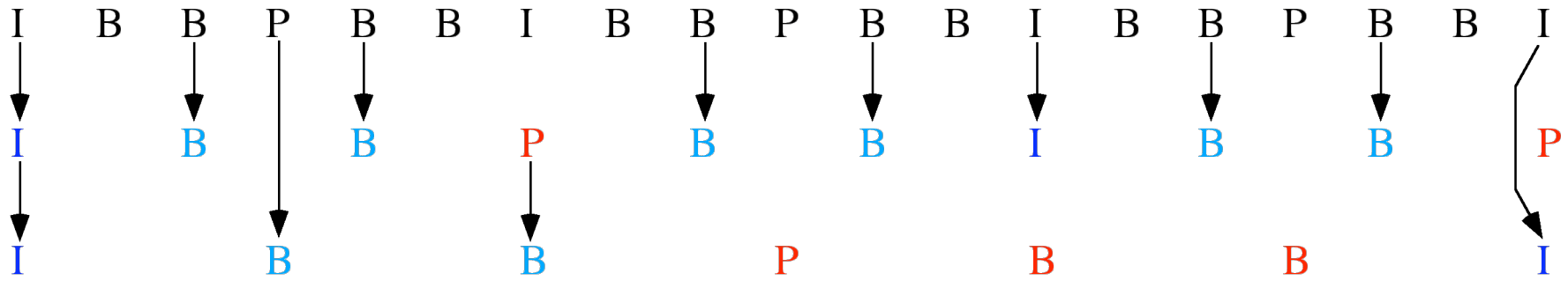
=> Teilnehmer brauchen individuelle Datenrate

• Filtern komprimierter Datenströme in Mehrpunktszenarien?



=> Skalierung an der Quelle

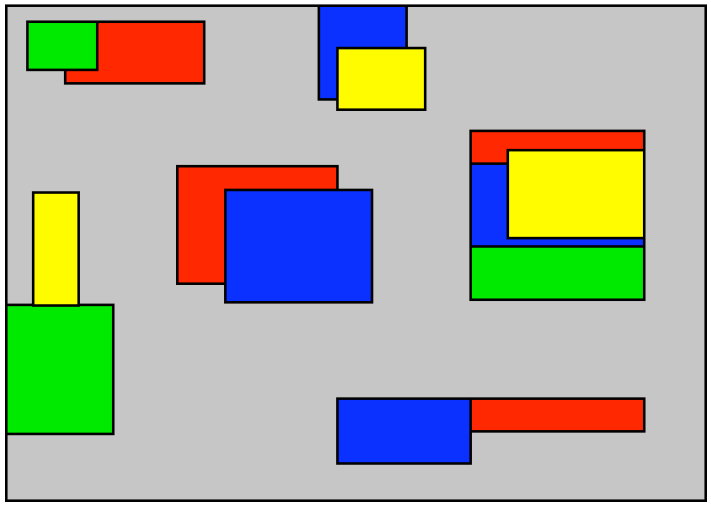
1. Idee: hierarchische Medienkodierung
   - mehrere Ströme mit unterschiedlichen Datenraten
   - jeder ausreichend gut komprimiert



- gemeinsame Kodierung
  - Wiederverwendung von Bewegungsvektoren
  - filmabhängiger Mehraufwand
- Nur Rückwärtsvektoren
  - höhere Wiederverwendbarkeit
  - schlechtere Kompression
- Aufwand 1 : 1,8 : 2,5
- nicht sehr flexibel
- viele Stufen nötig

2. Idee: dynamisch Medienströme konstruieren

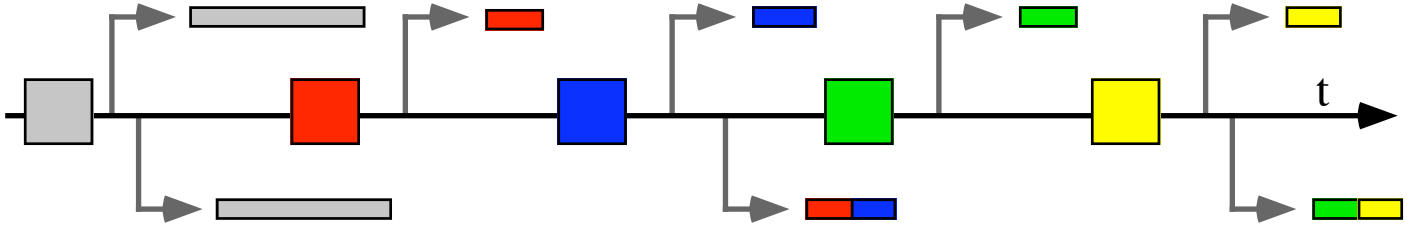
- Bereitstellung der Kodierungsbasis
  - Rohinformation
  - transformationskodierte Daten
  - Änderungsmatrix mit Generationsinformation



- Vergleichsmethode
  - Filter: CCD-Rauschen
  - Kamera-Wackeln?
  - Wie goß muß die Differenz sein?
  - Wie groß muß ein Unterschiedsblock sein?

  => Parameter abhängig vom Bild optimieren

- conditional Replenishment (Aktionsstrom)
- Erzeugung der Updateinformation beim Bildabruf



- $Bild_t := Bild_{t-m} + \sum_i Block_{i|x>t-m}$
- Synergieeffekte zwischen Medienströmen?
  - DCT
  - Änderungsinformation
- Einbettung in Standard-Videoströme
  - GIF
  - H.261 (McCanne: Intra-H.261)
  - MPEG nur mit trivialen Bewegungsvektoren
  - Problem Stützbilder (keyframes)

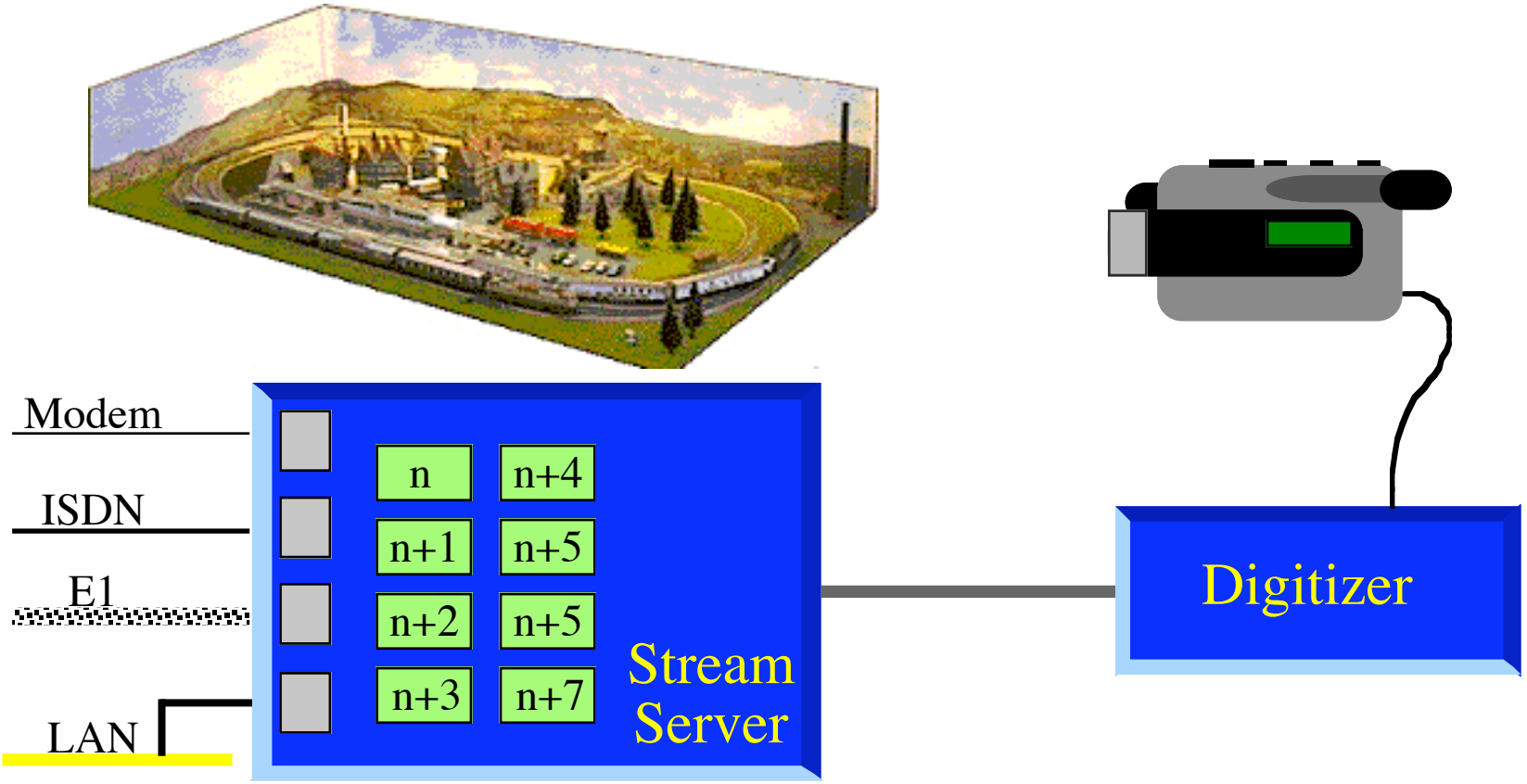
7.3.1 Webmedia

- serverbasiert
- 'Messung' der Bandbreite Server->Verbraucher
  1. kleiner Schreibpuffer + TCP
  2. Rückmeldungen

3.3.1.1 WebVideo

- Medienverteiler
  - konstruiert komprimierte Medienströme
  - verschiedene Formate
  - Bandbreitenanpassung

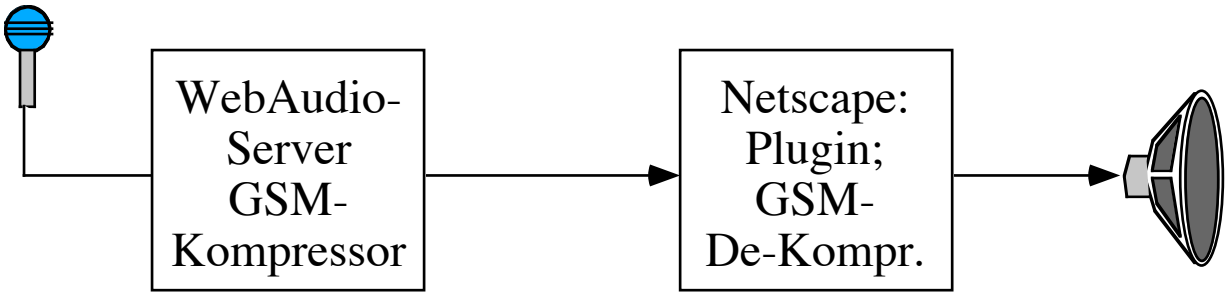
- Internet-Modellbahn als Beweis



- WebVideo
  - Mini-WWWServer
  - Windows 95, Linux, SUN-OS

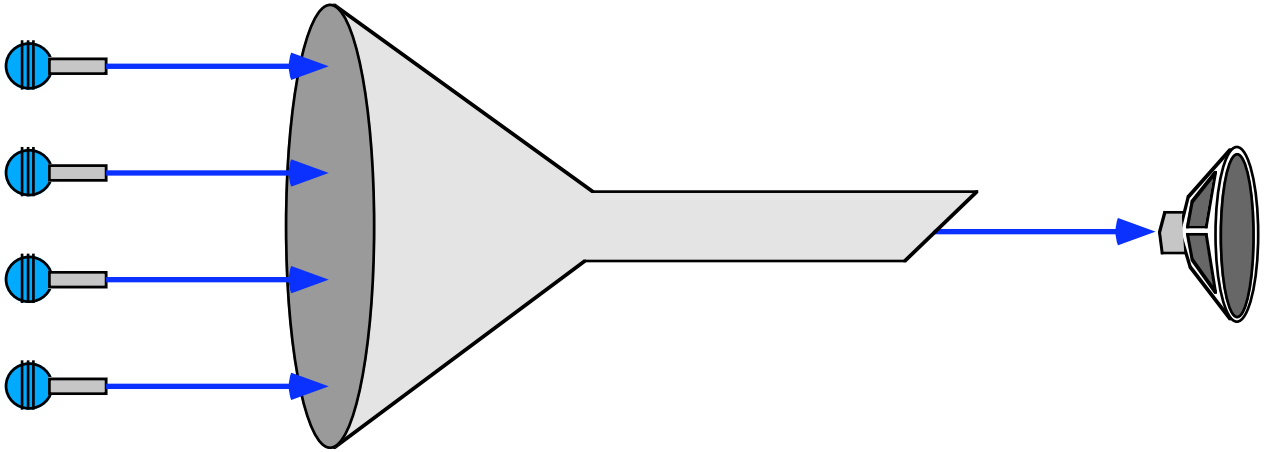
7.3.1.2 WebAudio

- GSM-Codec
- Stärkere Kompression nutzlos
  - IP-Pakete 'kosten'
  - Länge nur wenig relevant
  - Zeit in einem Paket
  - 7 kbit/s = 0,9 kByte/s => 1 sec = 1 IP-Paket
  - Delay 100 msec => 90 Byte/Paket ~ 180 Byte/Paket



- Netscape-Plugin
  - Java zu langsam
- WebAudio-Server
  - Mini-WWWserver

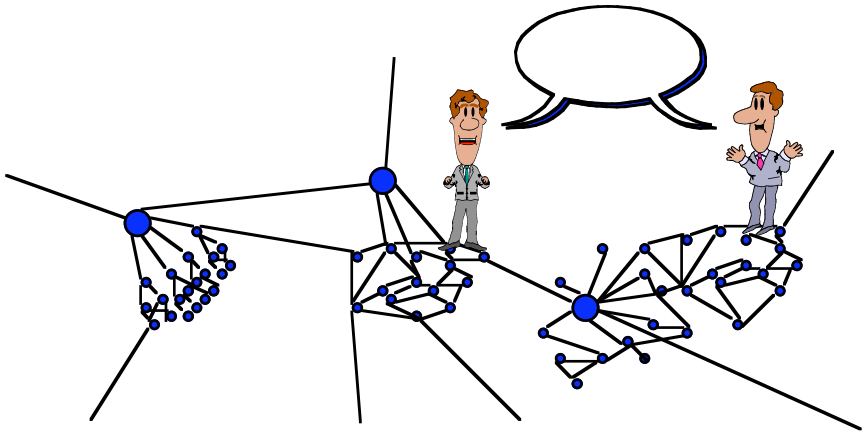
- Mixer im Empfänger für Mehrpunktszenarien



- Redundanz
  - Idee: Kompressionsverfahren mischen
  - Paket $P_n = (A_n , B_{n+1,n+2})$

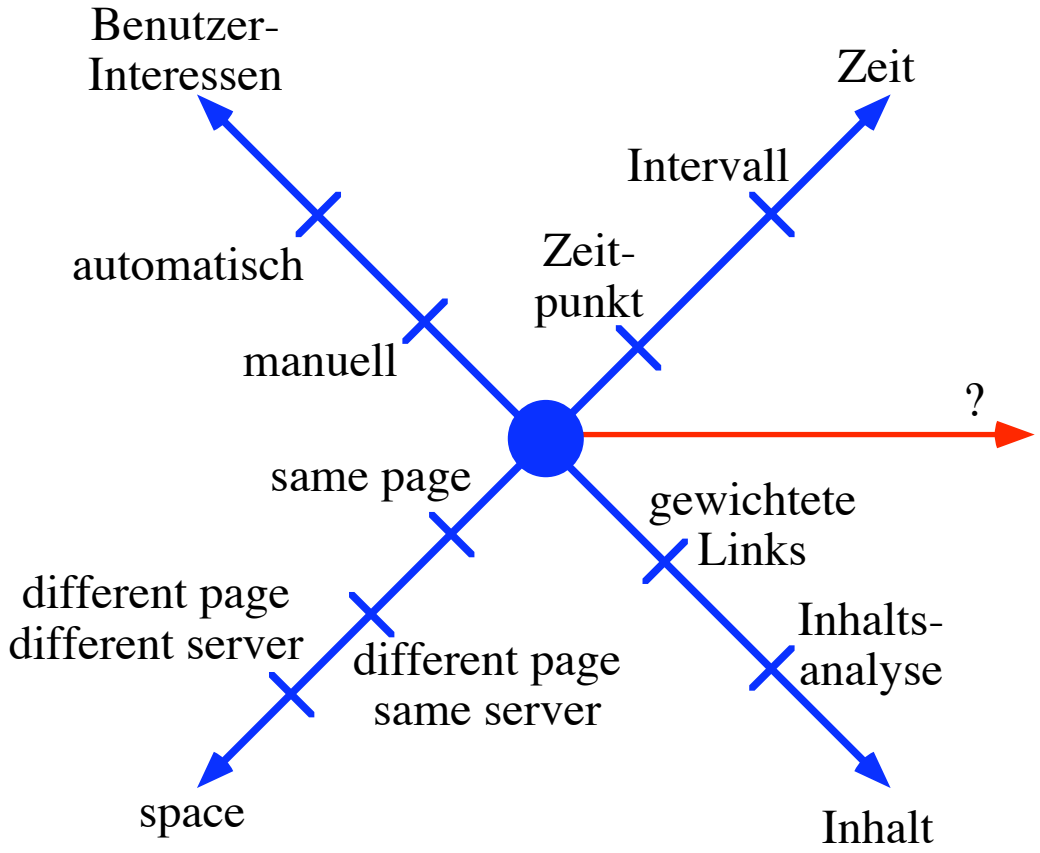
7.3.2 Virtuelle Präsenz

- WWW ist ein chaotischer Informationsraum
  - Benutzer allein
  - Suchmaschinen und Linkpages

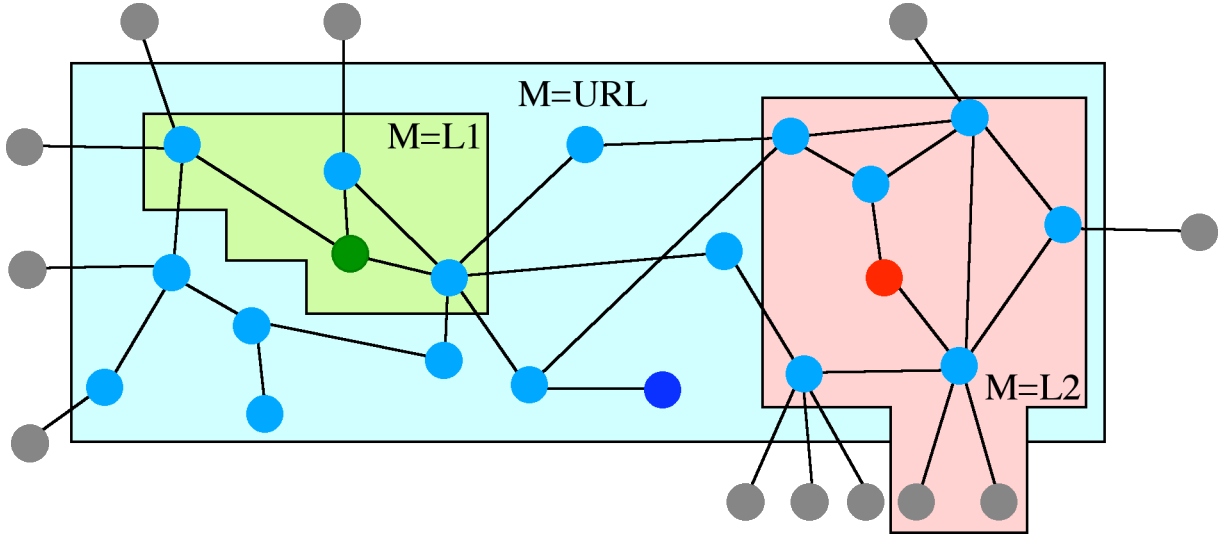


- Kontakte herstellen
  - andere Brower sichtbar machen
  - Treffpunkte eröffnen
- Konferenzen eröffnen
  - Audio, Video, Application-Sharing, Chat, …
  - Co-Browsing
- Neues UI für Konferenzdienste
- Orthogonal zu AltaVista

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

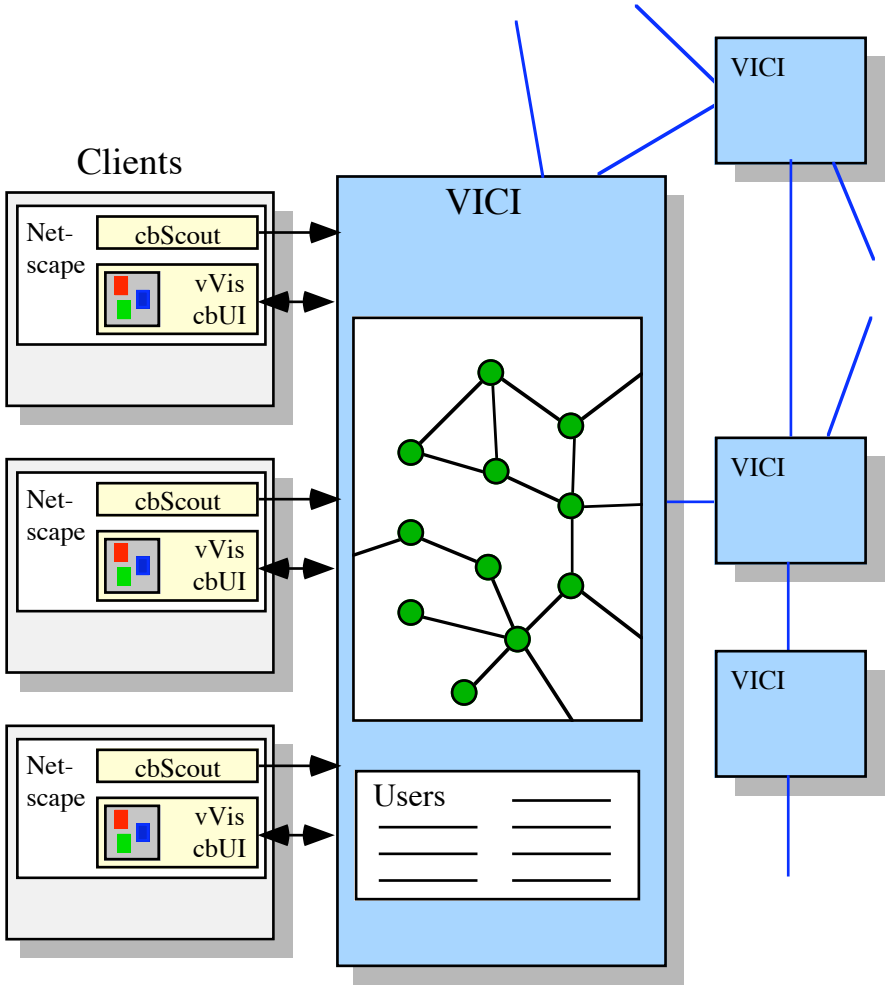


- URL-basierte Metriken
  - Schluß vom Aufenthaltsort des Benutzers auf Interessen des Benutzers
  - space: Linkdistanz < r
  - Inhaltsbezug: Gewichtete Links: $\sum w_i < r^*$;
    Contentmatching: $|C_i - C_k| < r'$
  - Auswertung mit Karte des WWW

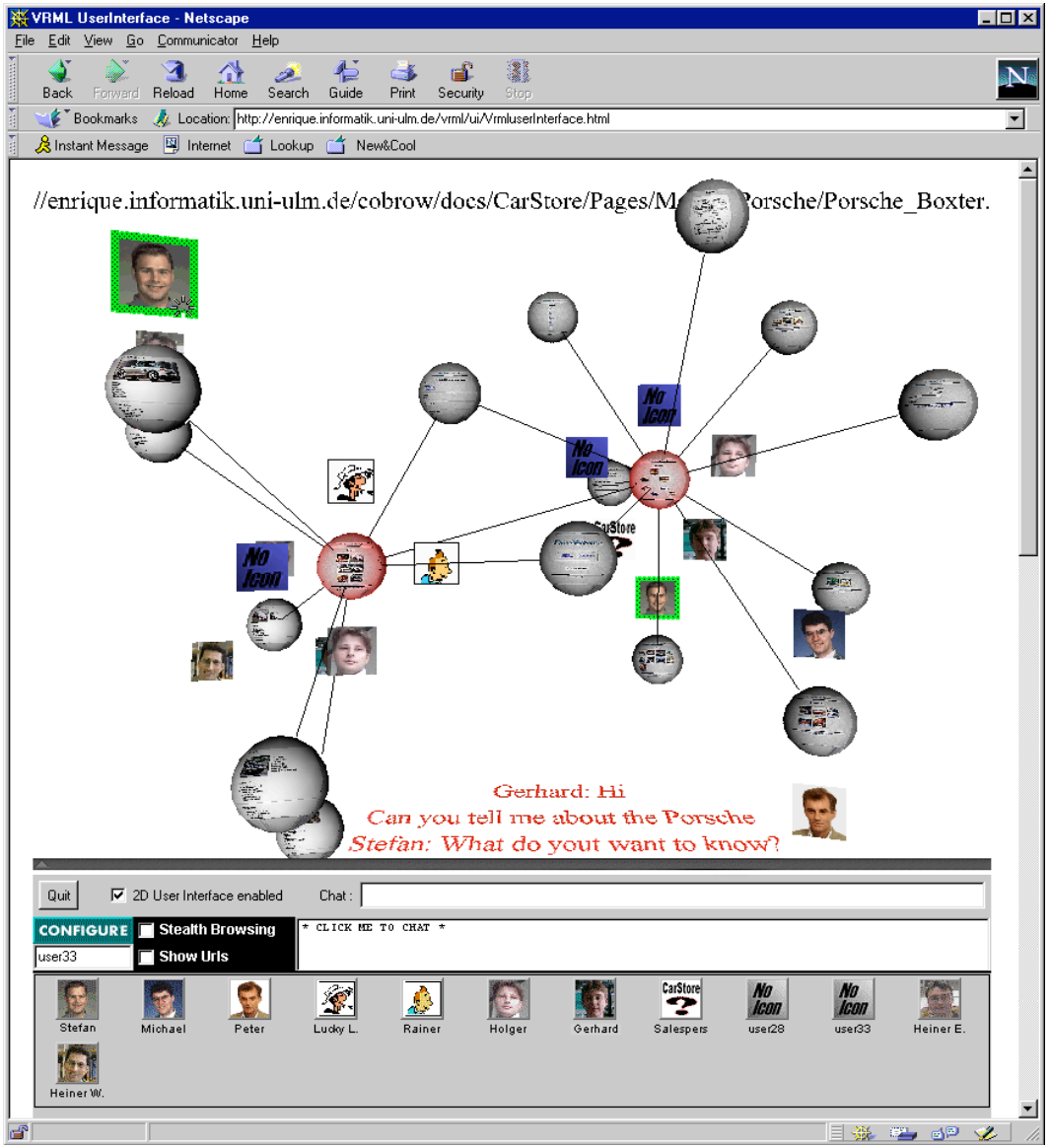


- Benutzerbasierte Metriken
  - Benutzerinteressen im CoBrow-Server registrieren
  - manuell eingeben oder/und automatisch ermitteln (Browserfilter)
  - Profile 'matchen'

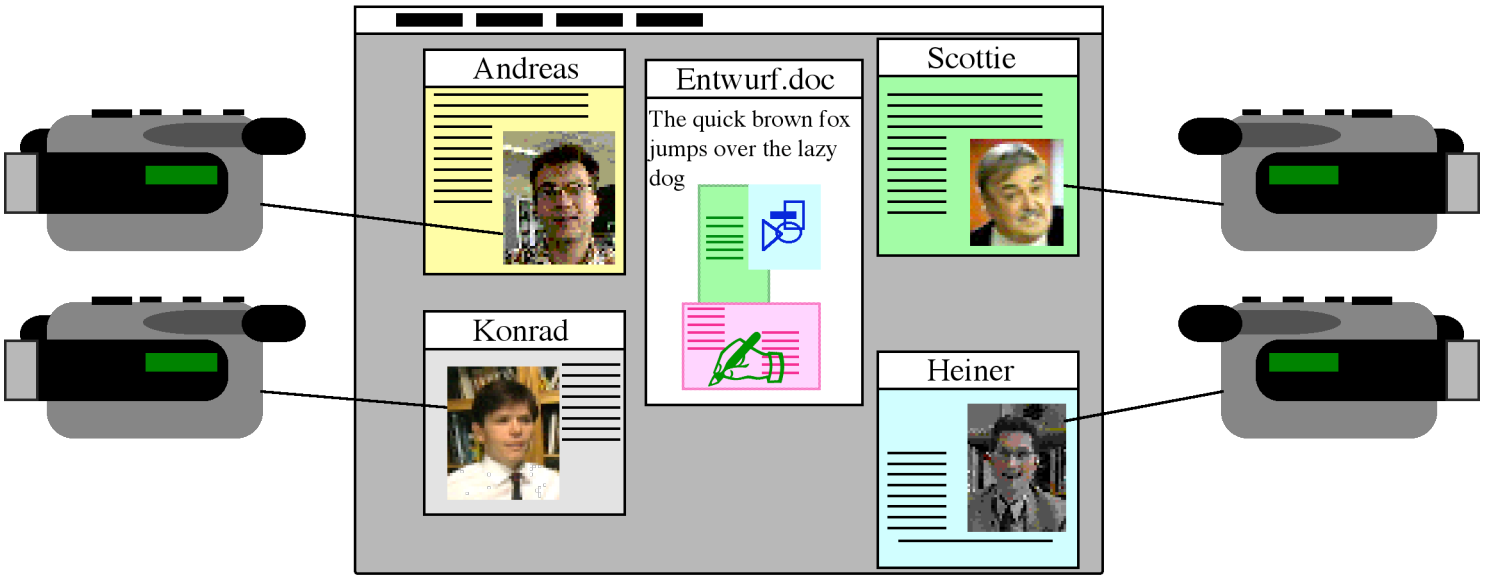
- Vicinity-Server
  - user-Position feststellen
  - Nachbarschaft berechnen
  - metrics: space, time, content
  - verteiltes System
- cbScout
  - Applet
  - hält Verbindung: endloses GIF
  - meldet betrachtete Seiten
  - alternativ: JavaScripts



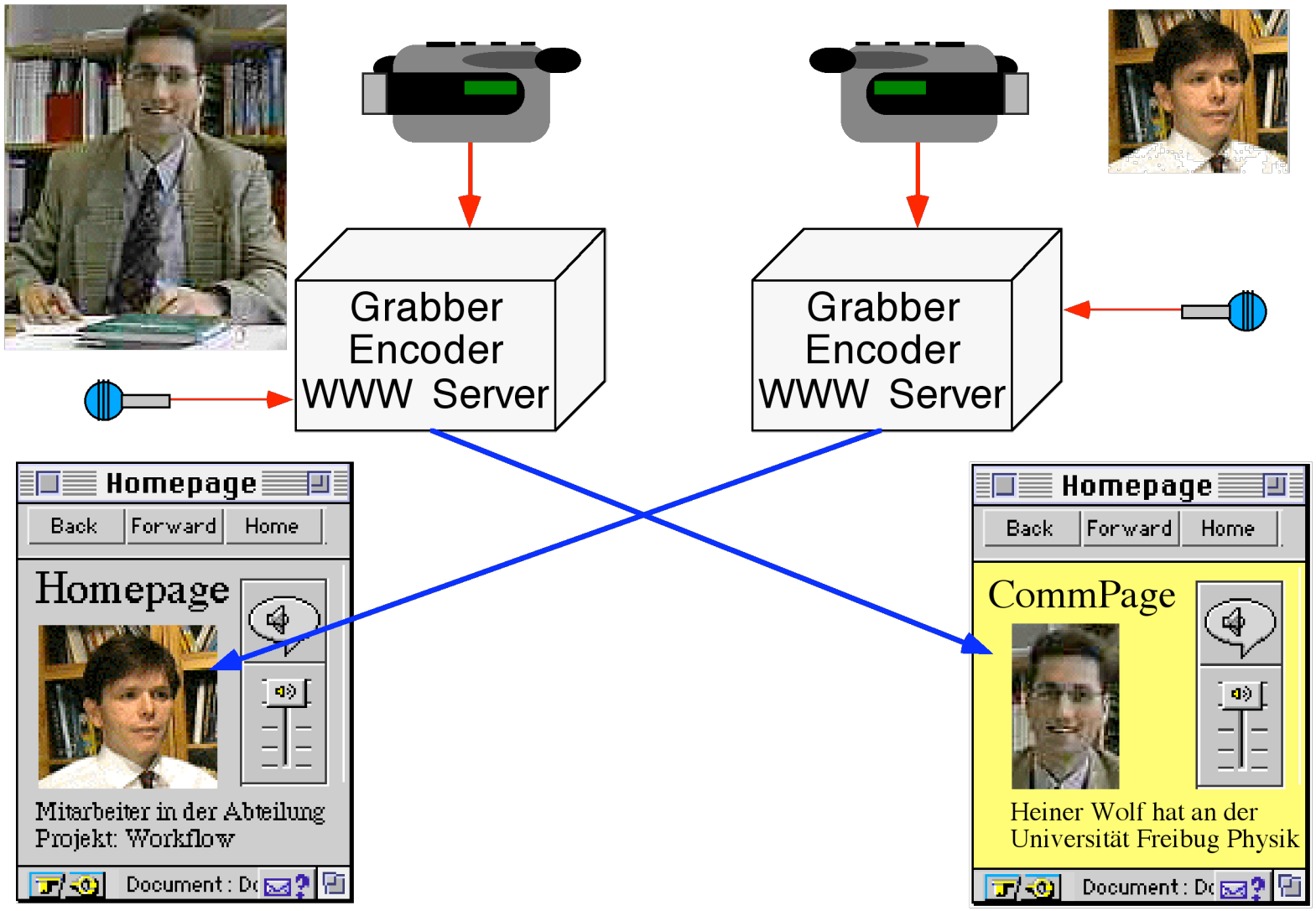
- Meeting Place
  - Visualisierung der Nachbarschaft
  - Konferenz-Management
- Primitive Variante
  - ASCII-Liste von Nachbarn
  - Button drücken: Konferenztool starten
- WWW-Variante
  - Icon-Liste mit URLs
  - Anklicken bringt Comm-Page
- VRML
  - Virtual Reality Modelling Language
  - MPEG-4, Java3D,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



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



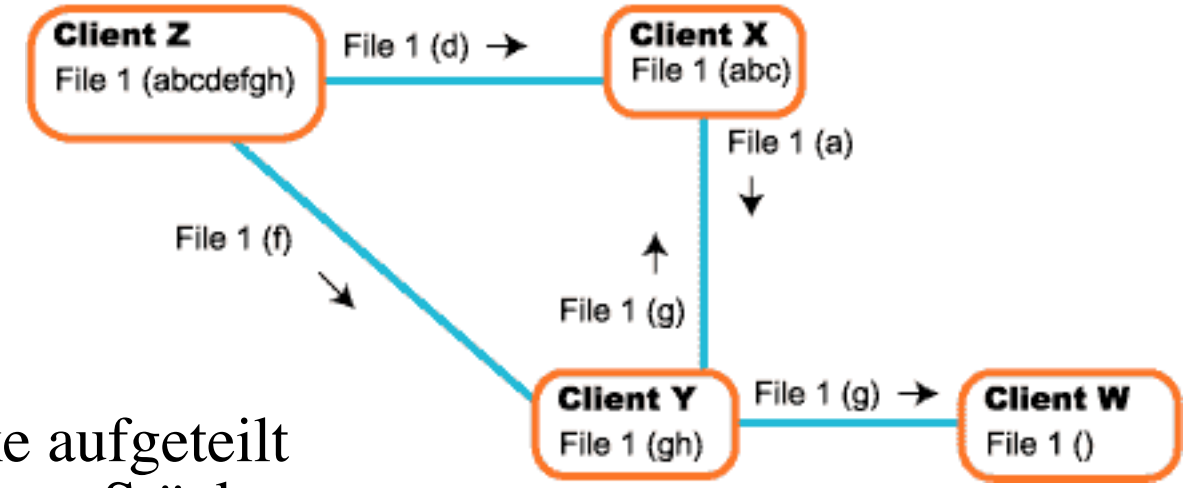
- WebMedia als Basis



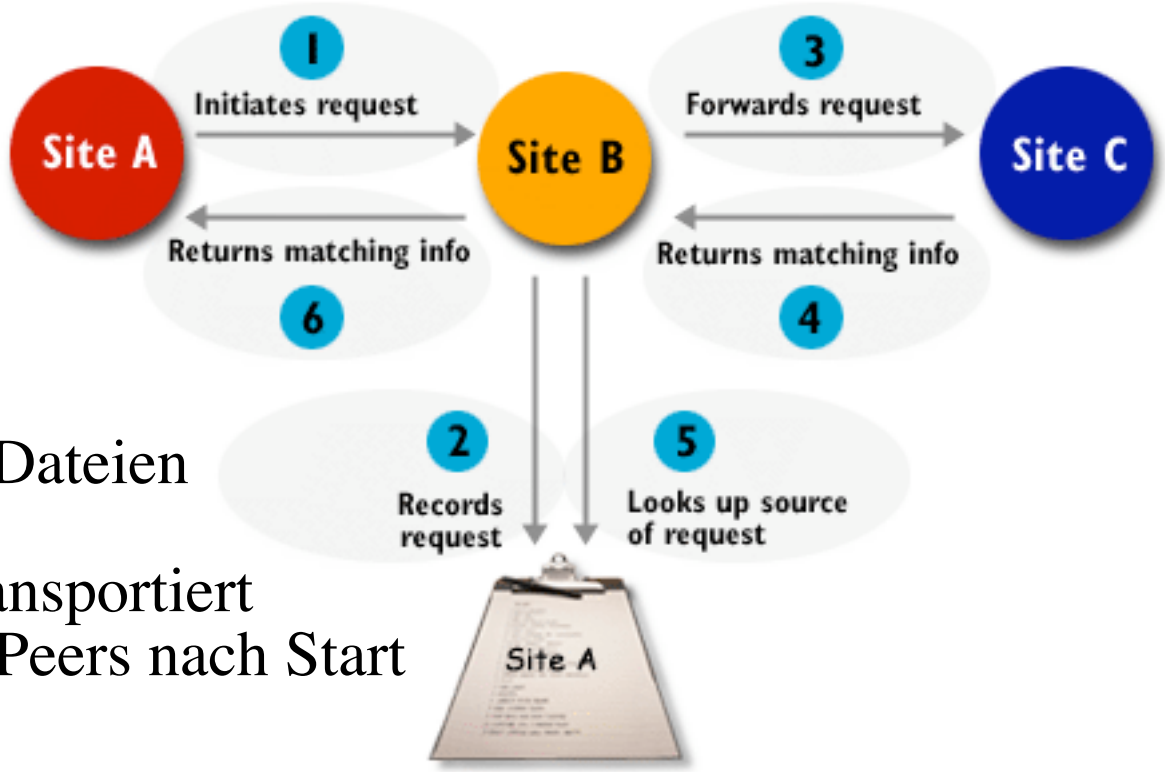
7.4 Dateiverteilung: Peer to Peer Netzwerk

- www.openp2p.com (O'Reilly)
- Idee: Dezentrale Speicherung größter Datenmengen
  - Summe der PC-Festplatten
  - jeder PC kann auch Fileserver sein
  - Problem: Dateien *finden*
  - 'Decentralized Resource Discovery'
  - Erweiterung auf Computing siehe Teil 2
- Zentraler Verzeichnisdienst
  - Peers registrieren Dateien und Metadaten
  - Überlastproblem und Fehlertoleranz lösbar (-> Google)
  - juristisch verwundbar (-> Napster)
  - Businessmodell?
- Dezentrale Verzeichnisse
  - in jedem Peer suchen skaliert schlecht
  - Untermengen bilden und dort suchen
  - Wie findet man Peers?
  - 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)



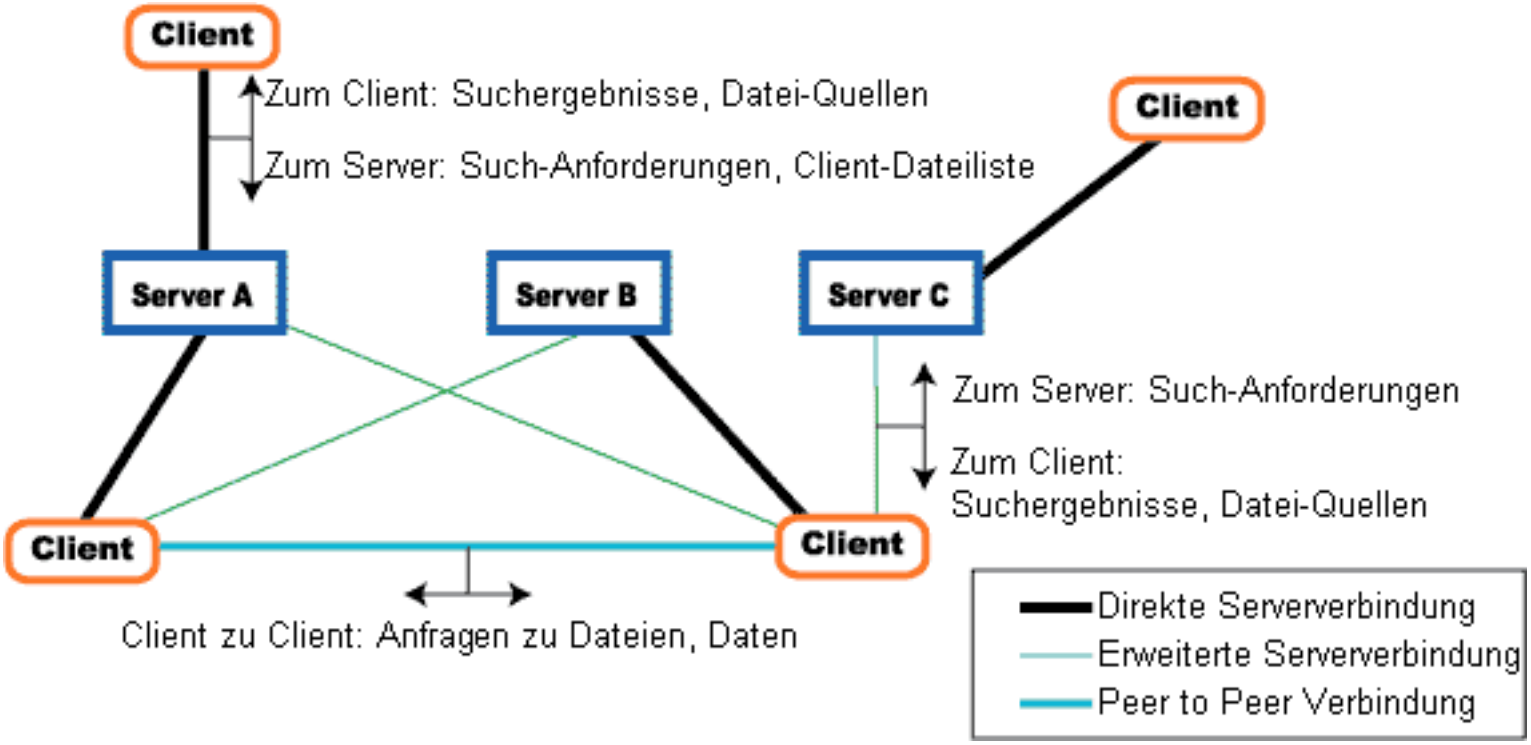
- Chunks
  - Datei wird in viele kleine Stücke aufgeteilt
  - paralleler Download verschiedener Stücke
  - sequentieller Download nichtzusammenhängender Stücke
  - Software setzt Stücke zusammen
  - gleicht heterogene Netzwerksturkturen aus
  - Fehlertoleranz durch breite Download-Basis
  - funktioniert 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

- 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