Universität Bielefeld
Technische Fakultät

R|V|S
Rechnernetze und
Verteilte Systeme

Vorlesung 2

Rechnerarchitektur

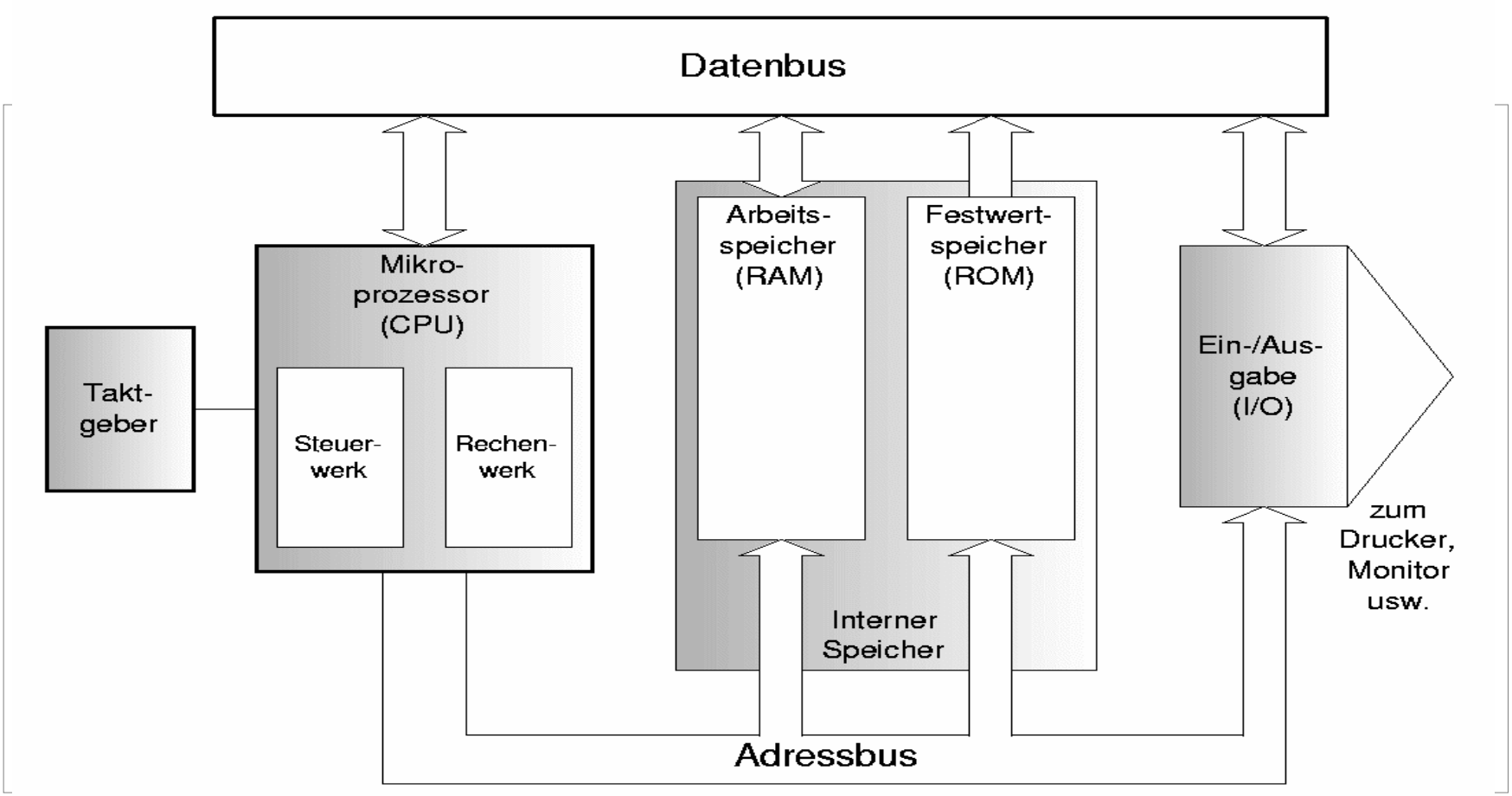
Wintersemester 2001/2002

ladkin@rvs.uni-bielefeld.de

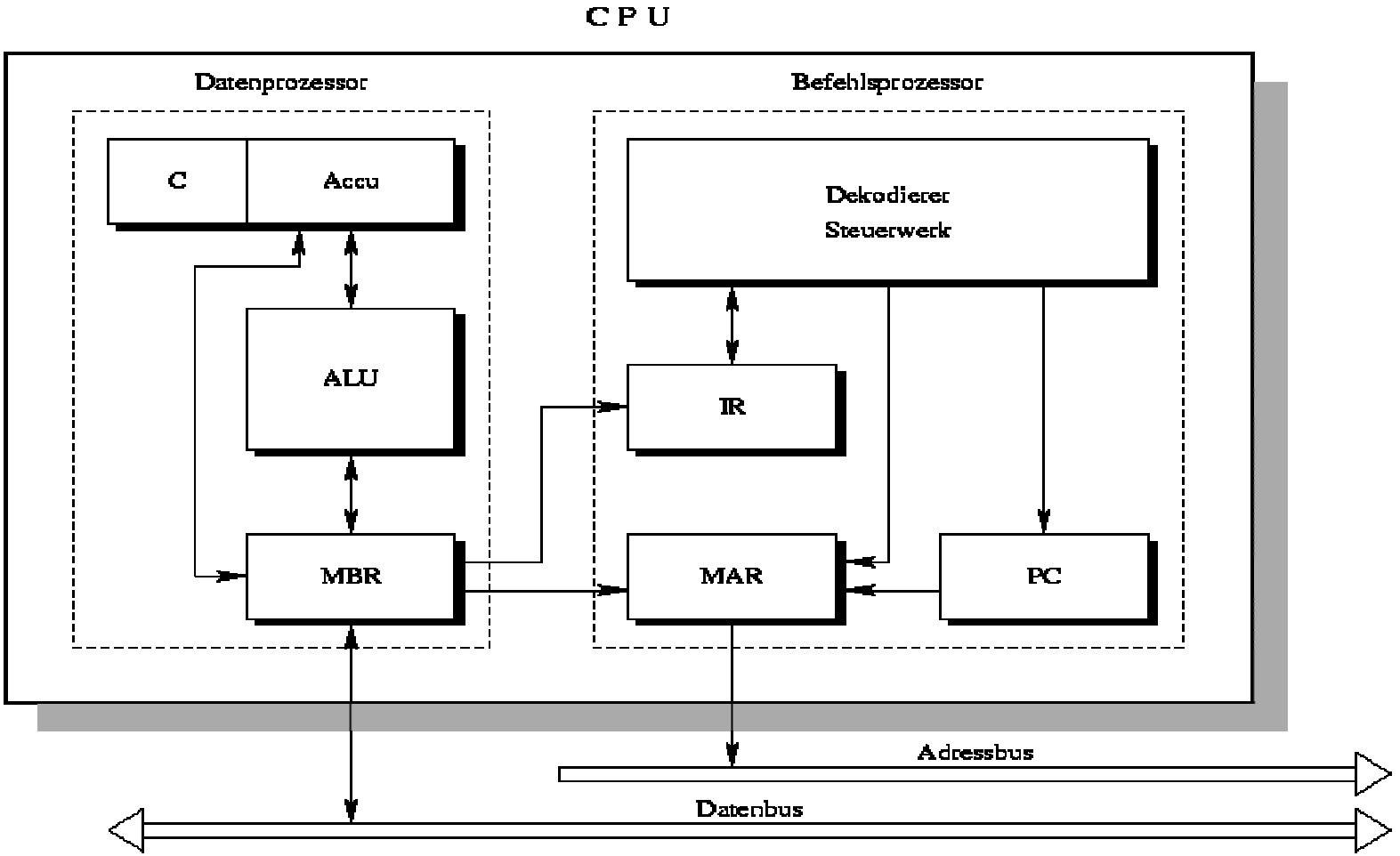
Inhalt

- Allgemeine von Neumann Architektur
- CPU Architektur
- Wie sie funktionieren
- Assembly Sprache
- Befehl Ausführung

Architektur nochmal



CPU


CPU
Datenprozessor
Befehlsprozessor
C
Accu
ALU
MBR
Dekodierer
Steuerwerk
IR
MAR
PC
Adressbus
Datenbus

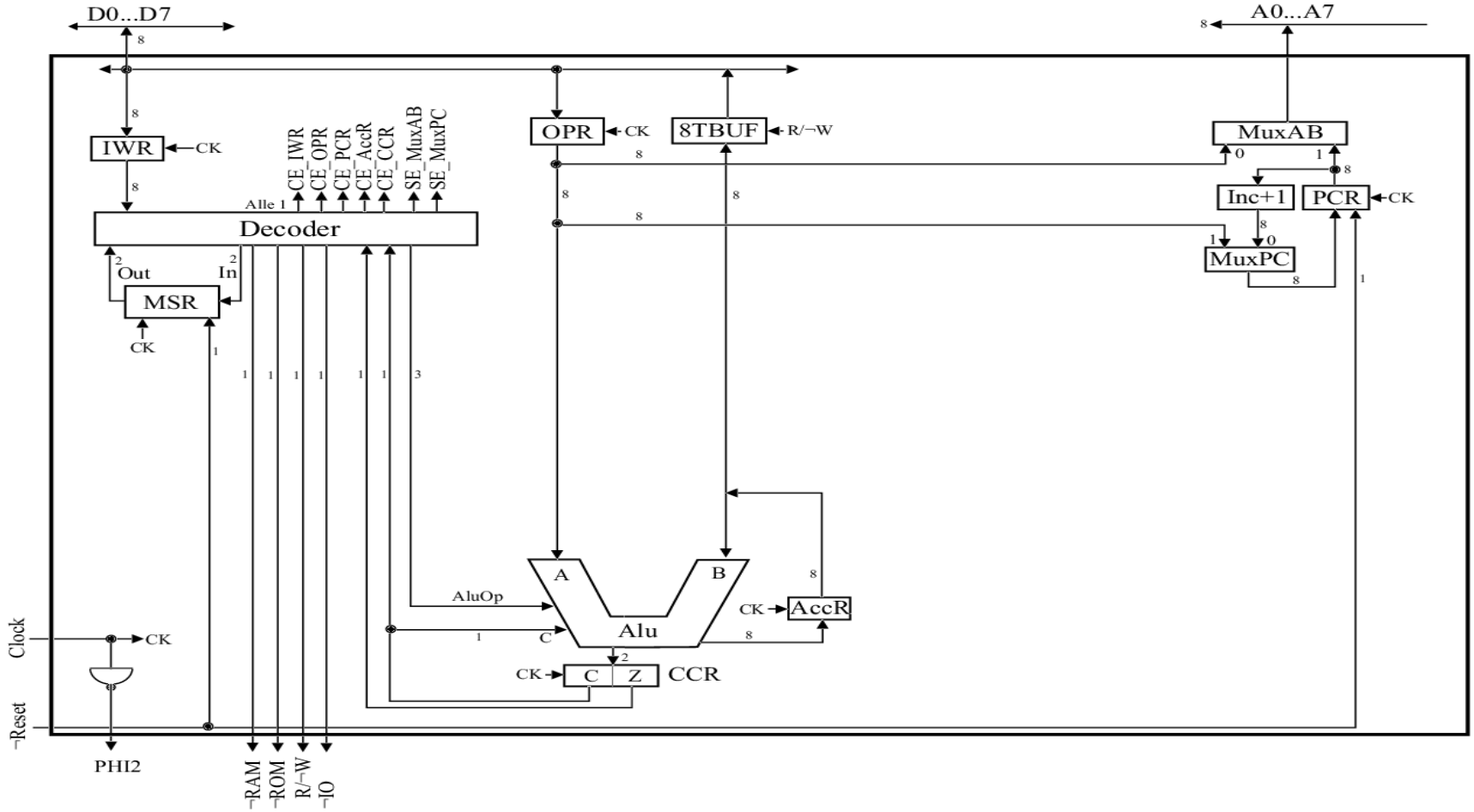
Assembly-Sprache

- Arithmetische Operationen
  - Add: ACC <- ACC + <Sp.Ad.-Inhalt>
  - Subtract: ACC <- ACC - <Sp.Ad.-Inhalt>
  - Shift (multiply by 2): ACC <- ACC * 2 == alle Bits in ACC einmal nach links gestellt mit 0 in Low-Order Bit
  - Multiply: ACC <- ACC * <Sp.Ad.-Inhalt>
  - Divide: ACC <- ACC / <Sp.Ad.-Inhalt>

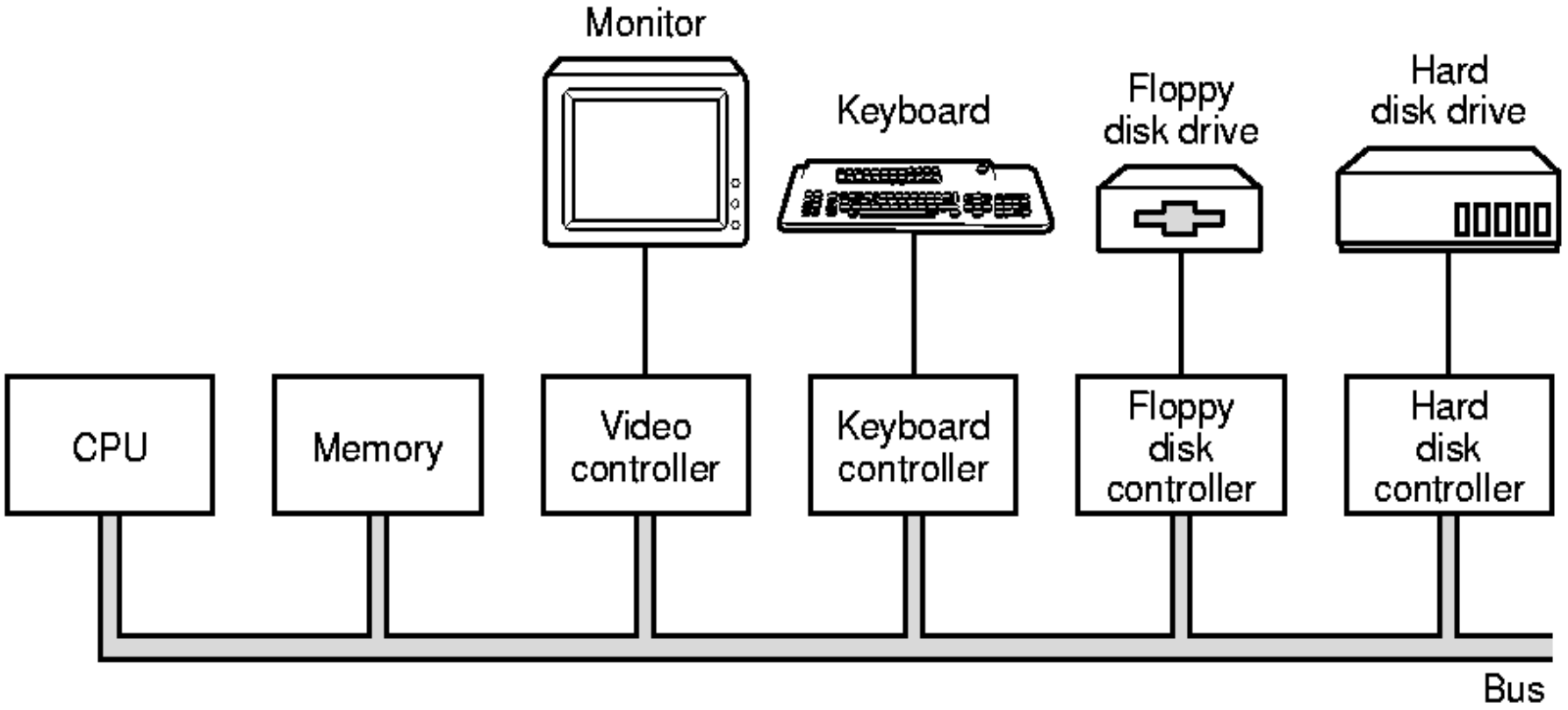
Assembly-Sprache

- Logische Operationen
  - "Jump <Befehl-Ad.>" : Inhalt der <Bef.-Ad.> wird in IR geladen; PC wird <Bef.-Ad.> + 1
  - "Load <Sp.-Ad.>": Inhalt der <Sp.-Ad.> wird in ACC geladen; PC wird PC + 1
  - "Store <Sp.-Ad.>": Inhalt des ACCs wird in <Sp.Ad.> gespeichert; PC wird PC + 1
  - "If ACC then <Bef.-Ad.>": ACC > 0 then PC wird <Bef.-Ad.>; ACC </= 0 then No-Op.

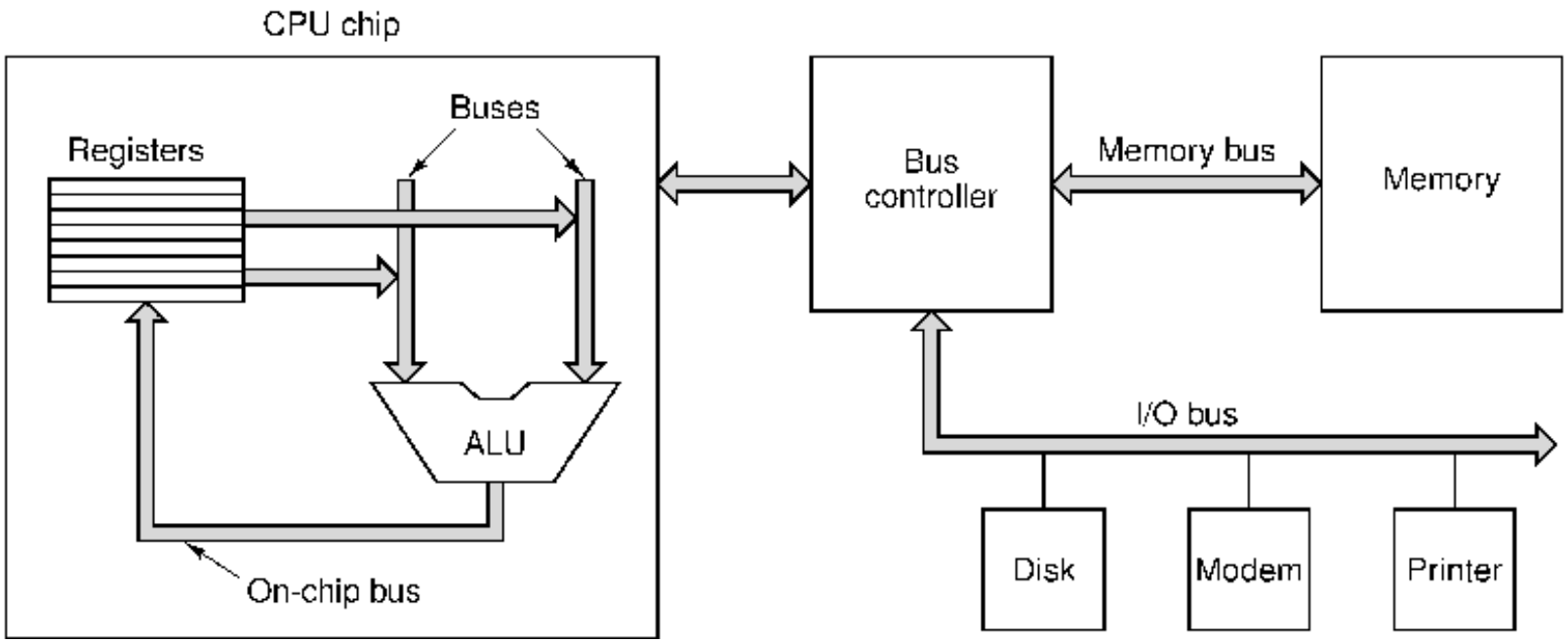
CPU des DEPs


D0...D7
A0...A7
IWR
CK
CE_IWR
CE_OPR
CE_PCR
CE_AccR
CE_CCR
SE_MuxAB
SE_MuxPC
Alle 1
Decoder
OPR
8TBUF
R/¬W
MuxAB
Inc+1
PCR
MuxPC
Out
In
MSR
A
B
Alu
AluOp
C
AccR
CCR
Z
Clock
¬Reset
PHI2
¬RAM
¬ROM
R/¬W
¬IO

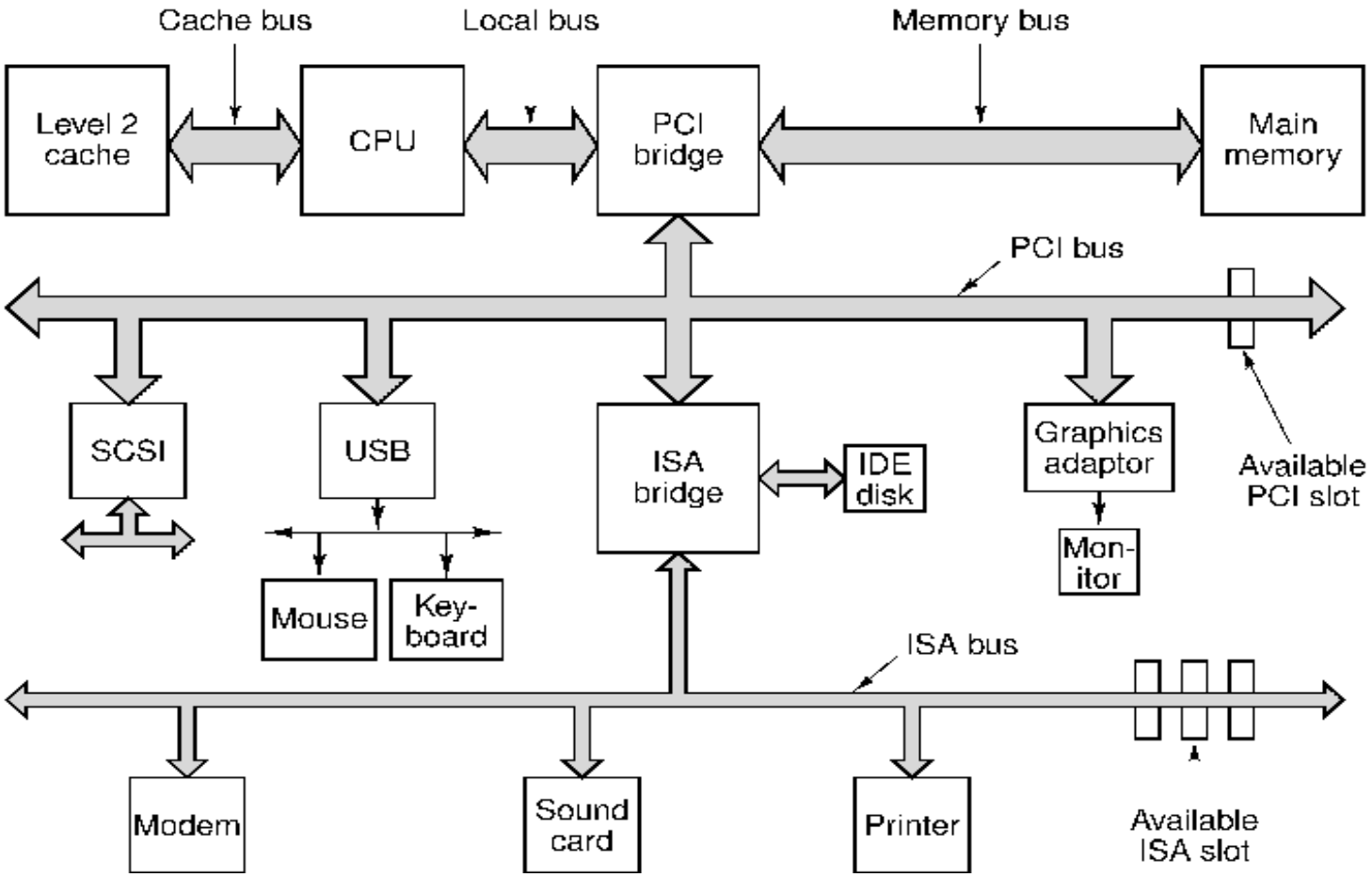
Busarchitektur - das Ideal



Busarchitektur - Der Plan



Busarchitektur - Die Realität



VM-Beispiel

- JUMP <Sp-Adr>
  - Dekodiere JUMP / <Sp-Adr>
  - <Sp-Adr> -> MAR; PC <- <Sp-Adr> + 1
  - Datum -> MBR
  - MBR -> IR
  - IR -> DSW

Universität Bielefeld
Technische Fakultät

RVS
Rechnernetze und Verteilte Systeme

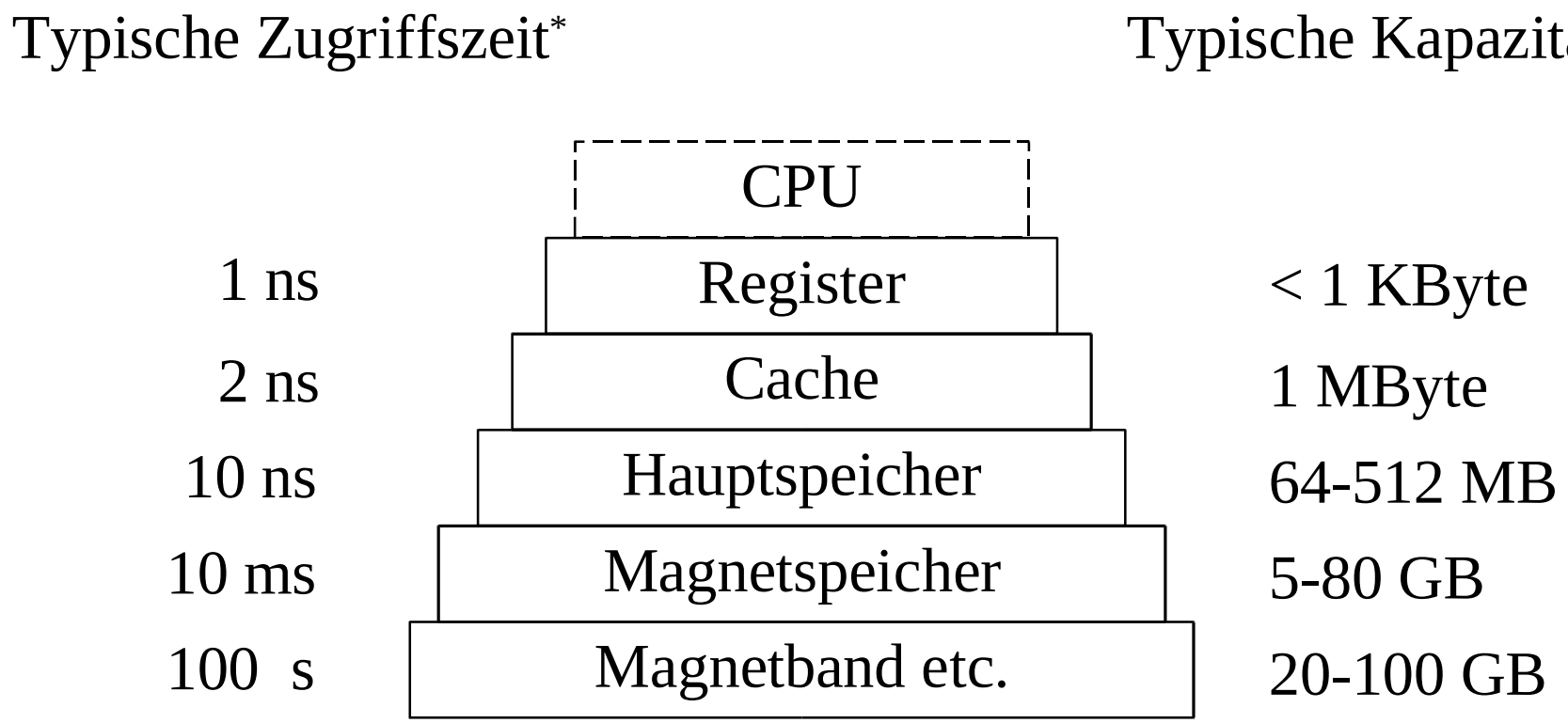
Speicherverwaltung

Technische Informatik II
Wintersemester 2002/03

Sommersemester 2001

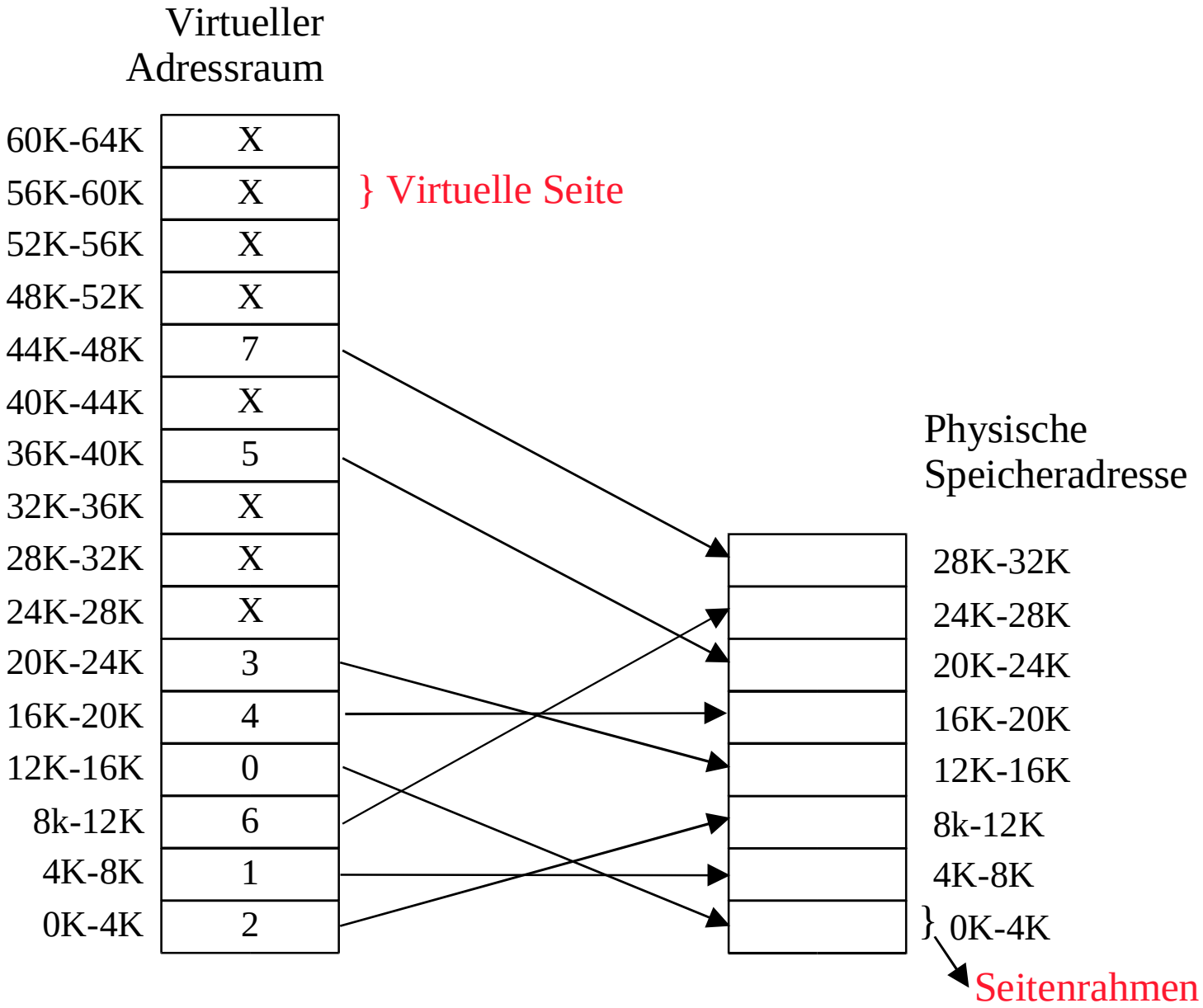
Heiko Holtkamp
Heiko@rvs.uni-bielefeld.de

Speicherhierarchie



* Sehr grobe Schätzungen

Virtueller Speicher: Paging



Das Vermeiden von Busy Waiting

- Die Schleife heisst jetzt

```
i <- 10;
 while i > 0 do
                read (TEG-R);
                Store;
                i <- i - 1
```

  weitermachen

- Die Schleife läuft nur 10 Mal durch

Vorfälle

Die Vorfälle zeigen, welche Probleme es in der Benutzung von Betriebssystemen in Missionskritischen Systemen geben kann

- Der Mars Pathfinder - Ein Scheduling Problem
- Ariane Flight 501 - Ein Requirements Problem
- Der USS Yorktown - Ein System Crash

Prozess Synchronizierung Puzzle I

- ```
  Process 1: (x: integer)
  begin x <- 0; x <- x+1; stop; end
  ```
- ```
  Process 2: (x: integer)
  begin read x; stop; end
  ```
- Was ist der gelesene Wert von x, wenn diese Programme concurrent laufen?

Prozess Synchnozierung Puzzle 2

- `Prozess 1: (x: integer)`
  `begin x <- 0; x <- x+1; stop; end`

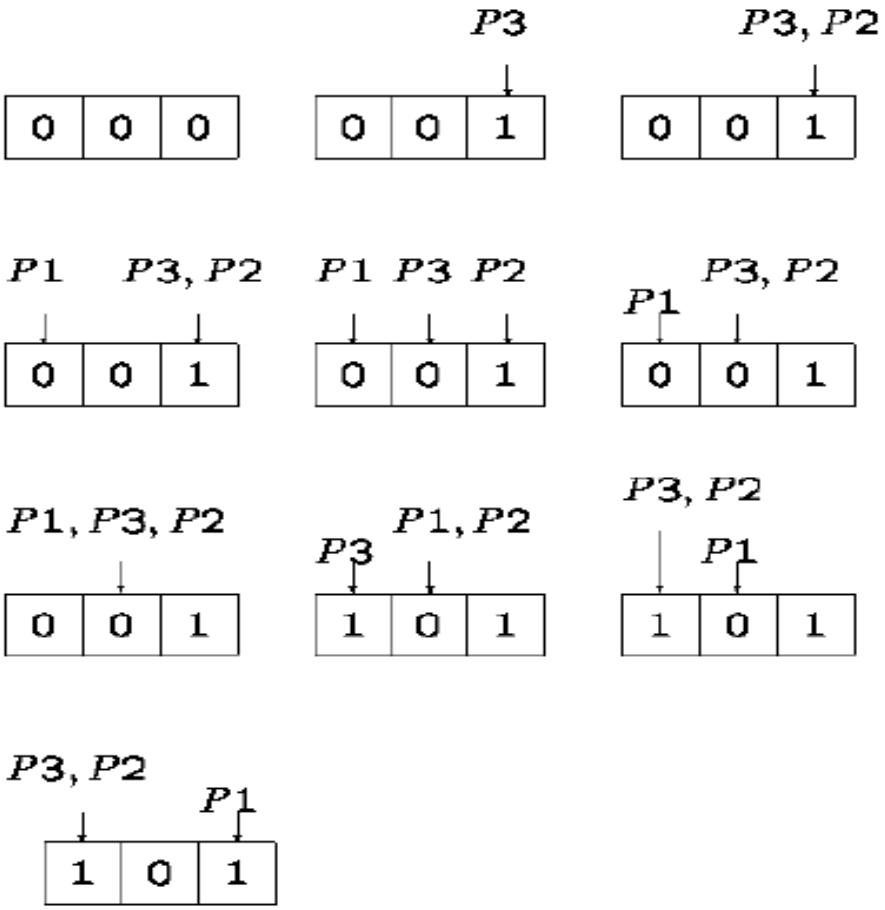
- `Prozess 2: (x,y: integer)`
  `begin y <- 0; y <- x+1; stop; end`
- Voraussetzung: Memory Platz `x` ist dergleiche als Memory Platz `y`
- Werte von `x`, `y` an Ende?

Prozess Synchronizierung Puzzle 3

- Wert von der Variabel z ist 1, falls es existieren 20 Blöcke freiverfügbarem Speicher;
- ...ist 2, falls es ...< 20 Blöcke....
- Wert von z ist 1
- Prozess 1 braucht 15 Blöcke, Prozess 2 auch
- Beide lesen z gleichzeitig
- Was passiert?

Prozess Synchronizierung Puzzle 4

- Programm 1 und Programm 2 lesen Variabel `turn`
- `turn` könnte von Programm 3 geschrieben werden
- `turn` hat 3 Bits
- `turn` = 001 bedeutet, Prog 1 kann den Drucker benutzen
- `Turn` = 101 bedeutet, Prog 2 .....


P3
P3, P2
0 0 0
0 0 1
0 0 1
P1
P3, P2
P1 P3 P2
P3, P2
P1
0 0 1
0 0 1
0 0 1
P1, P3, P2
P3
P1, P2
P3, P2
P1
0 0 1
1 0 1
1 0 1
P3, P2
P1
1 0 1

1

Sequentielle Operationen in Logik

- Sei es, wir möchten eine Sequenz von Operationen spezifizieren:-
- Also $\pi == (\alpha; \beta; \gamma)$
- Logik hat keine Sequentializierungs-Operator
- Wir definieren

$$A: \wedge pc = `a' \quad \wedge \alpha \quad \wedge pc' = `b'$$

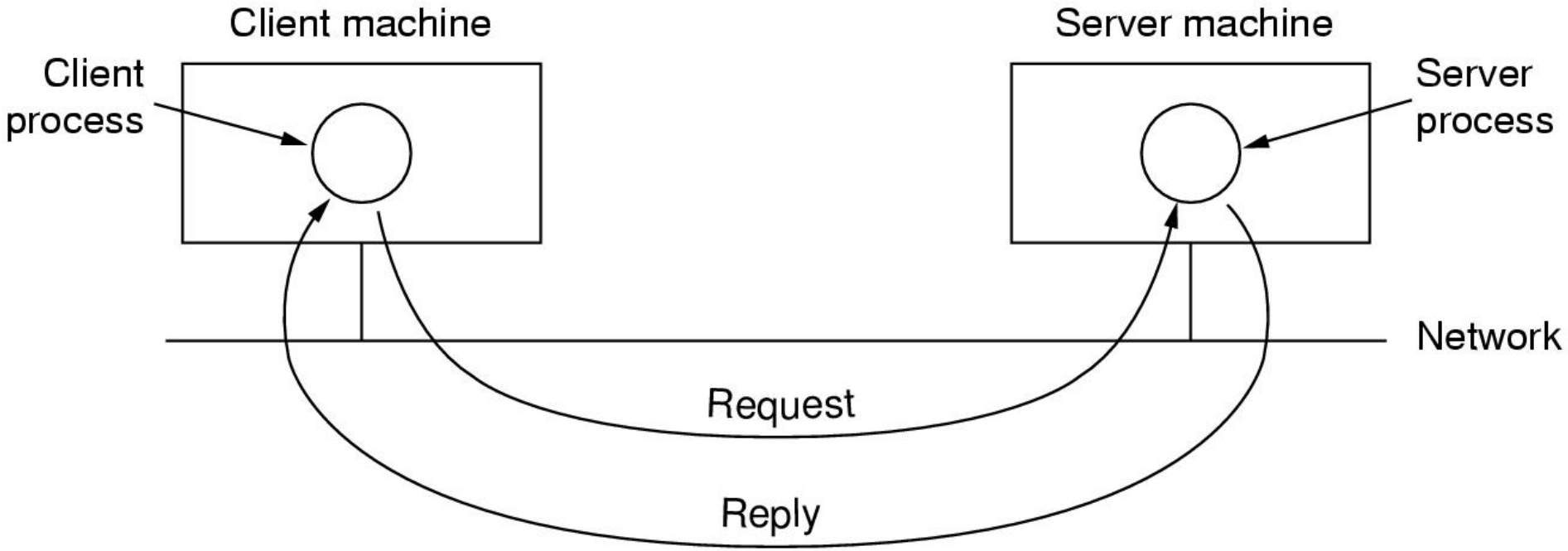
$$B: \wedge pc = `b' \quad \wedge \beta \quad \wedge pc' = `c'$$

$$\Gamma: \wedge pc = `c' \quad \wedge \gamma \quad \wedge pc' = `d'$$

und

$$\Pi == \Box (A \vee B \vee \Gamma \vee NoOp)$$

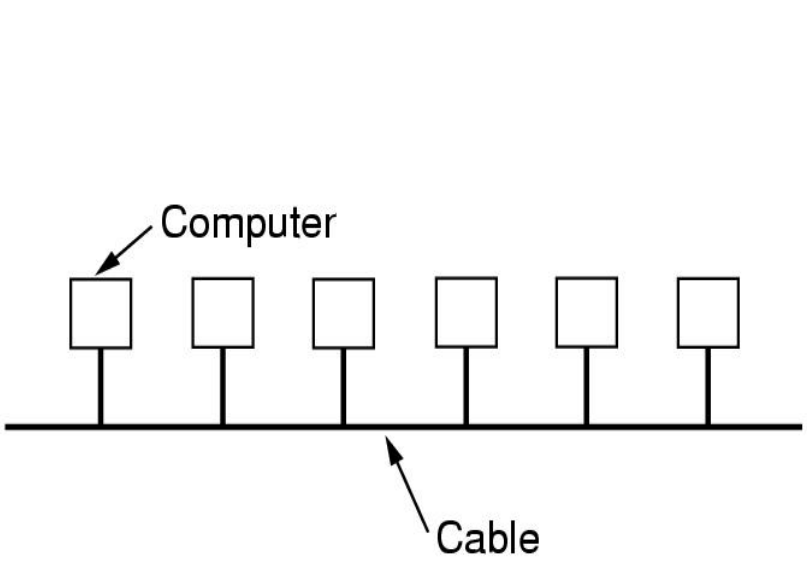
Client-Server Modell



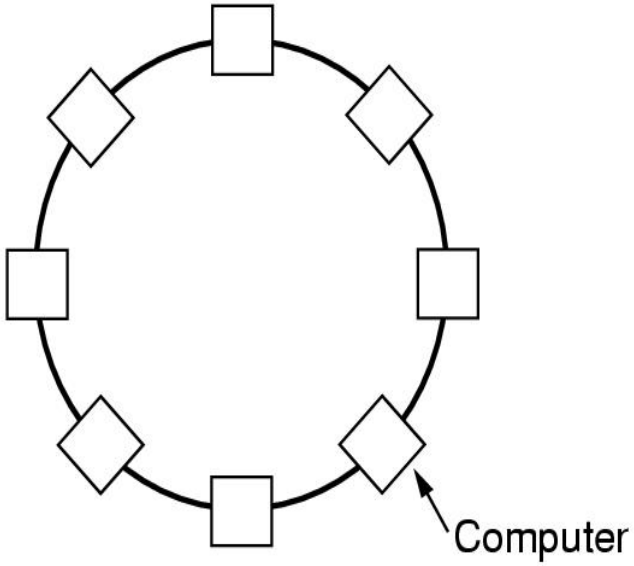
Netz-Typen

Interprocessor distance	Processors located in same	Example
0.1 m	Circuit board	Data flow machine
1 m	System	Multicomputer
10 m	Room	Local area network
100 m	Building	
1 km	Campus	
10 km	City	Metropolitan area network
100 km	Country	Wide area network
1,000 km	Continent	
10,000 km	Planet	The Internet

LAN-Topologie

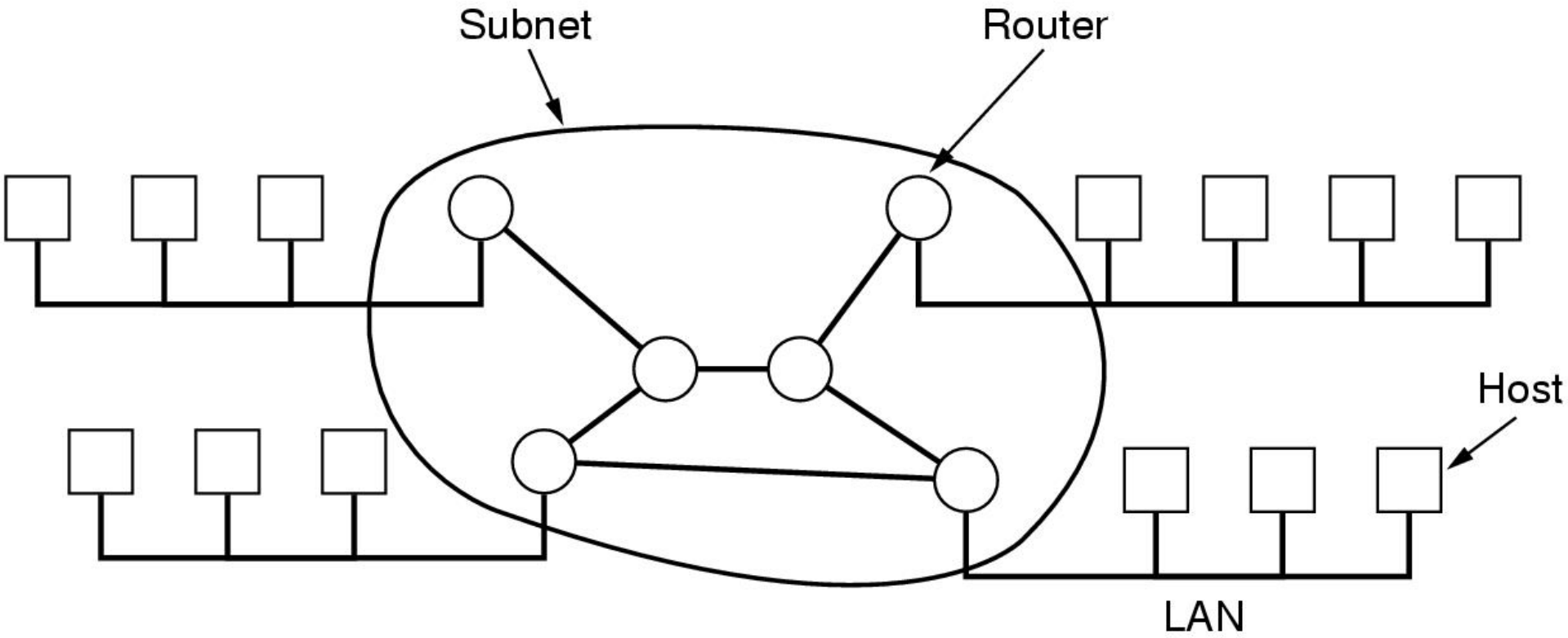


(a)

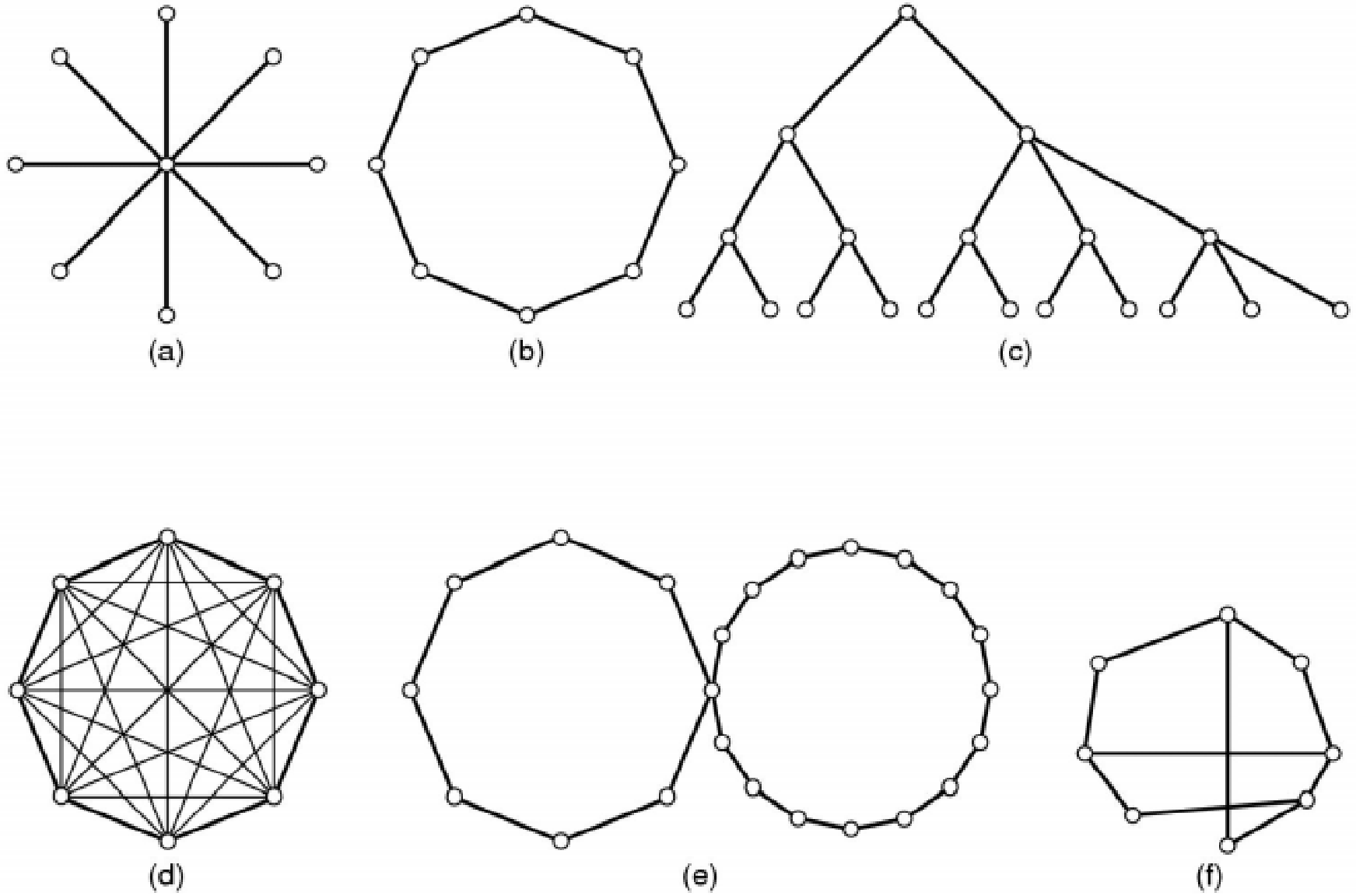


(b)

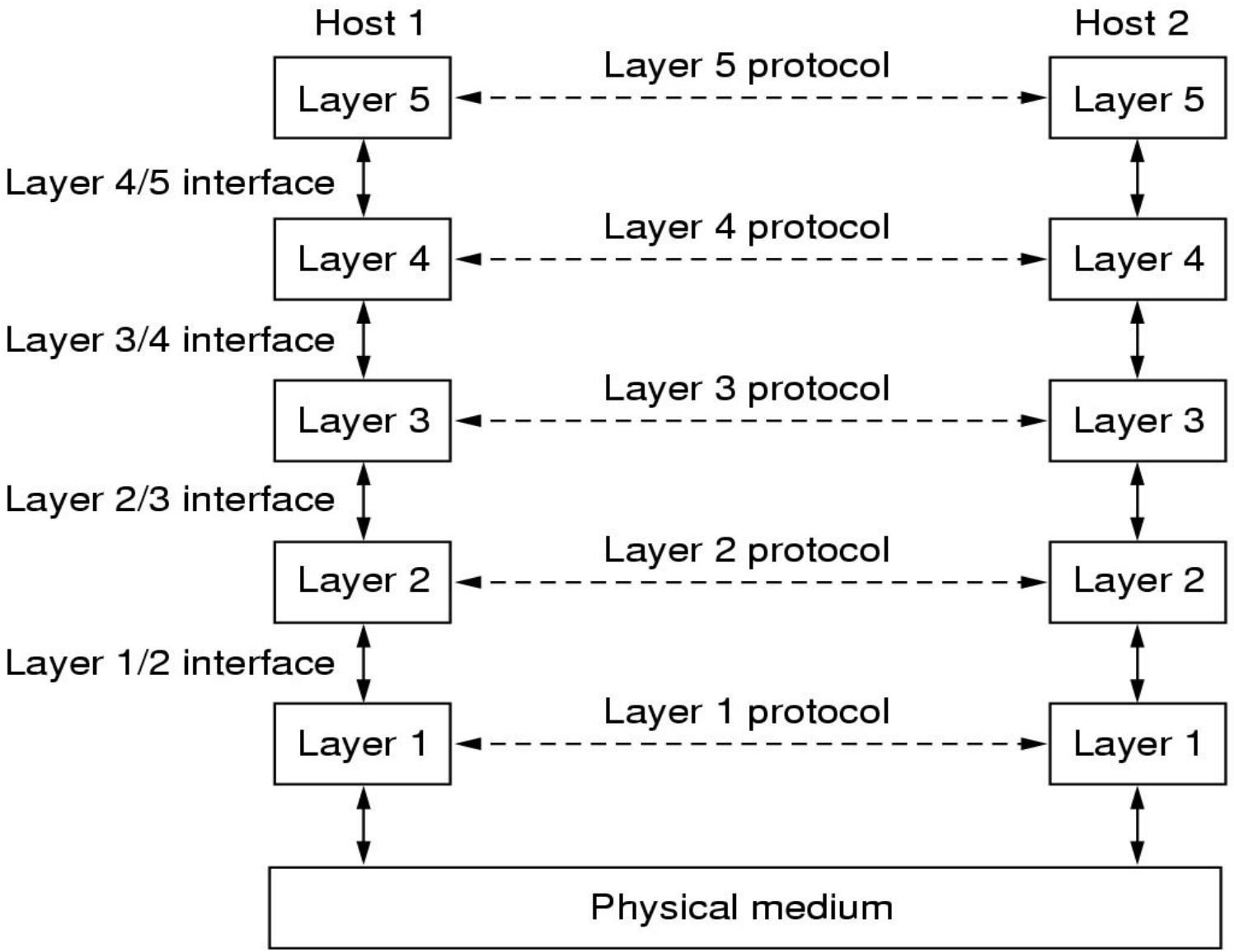
LAN-Topologie



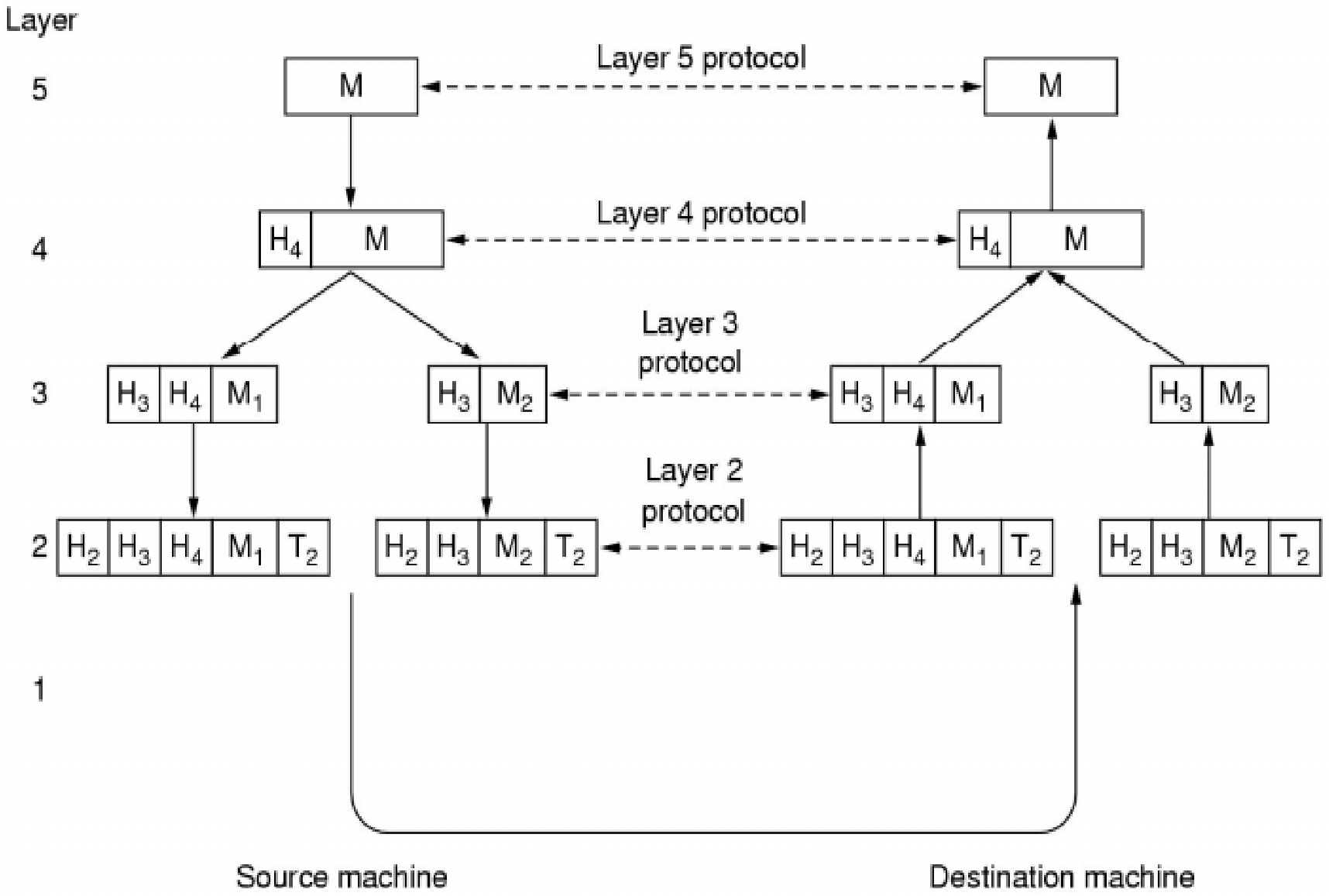
LAN-Topologie



Protokolle- Das Schichtenmodell



Protokolle- Das Schichtenmodell



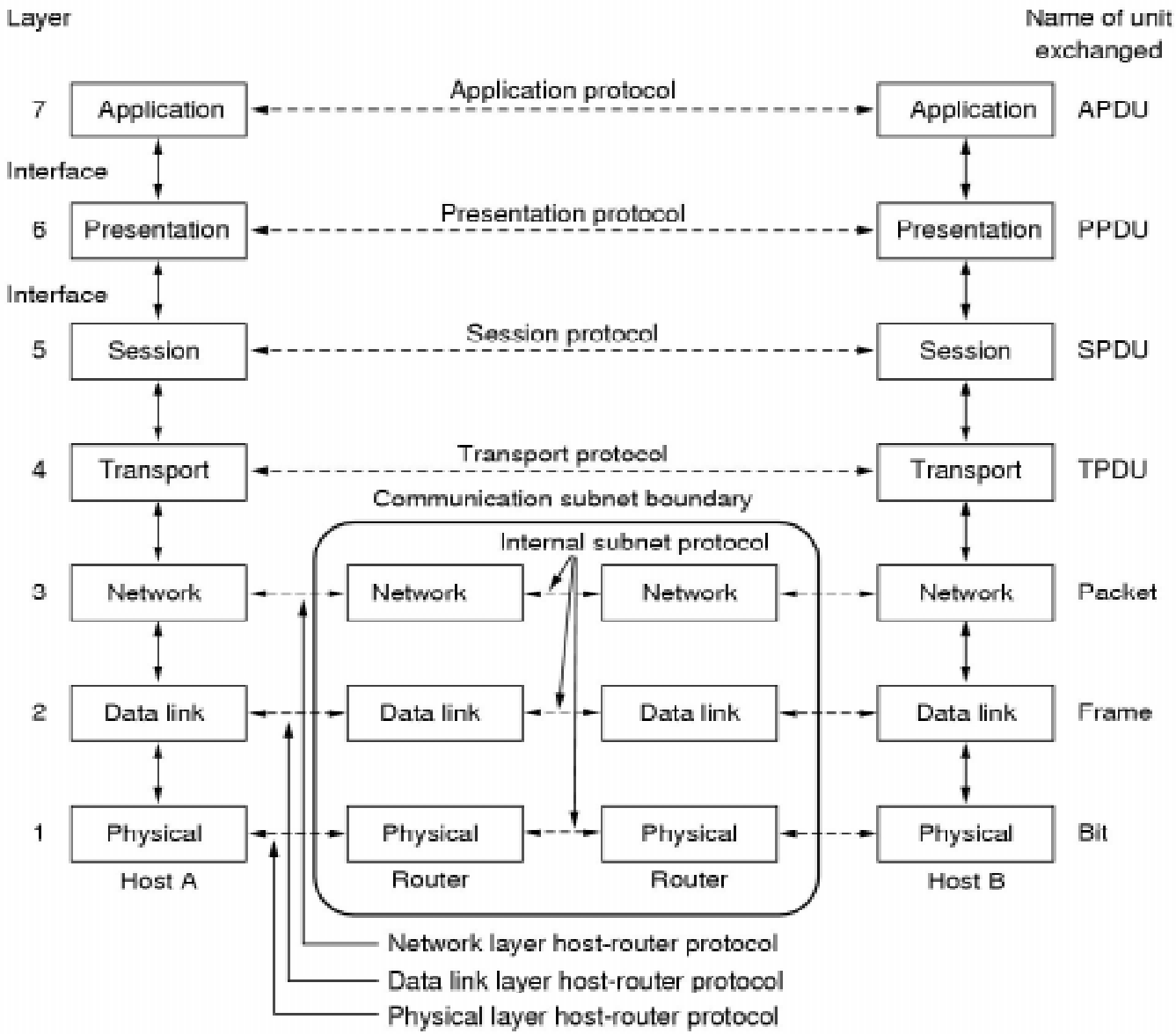
Services- Nachrichtentypen

- Messages
  - Unbestimmte Länge
- Streams
  - Unendliche Länge (nicht in Wirklichkeit!)
- Datagrams/Pakete
  - Bestimmte Länge

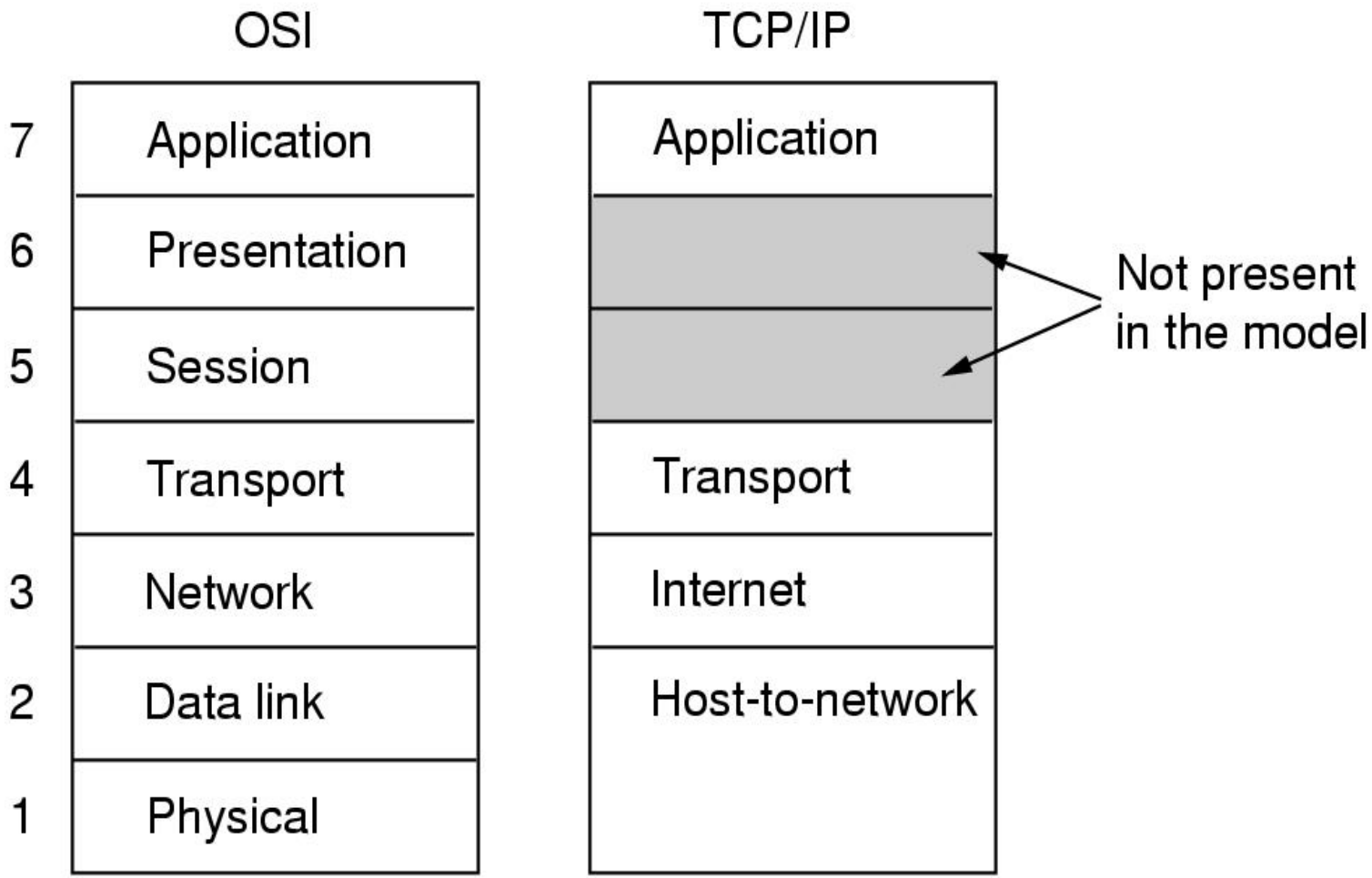
Services-Primitiven

	Service	Example
Connection-oriented	Reliable message stream	Sequence of pages
	Reliable byte stream	Remote login
	Unreliable connection	Digitized voice
Connection-less	Unreliable datagram	Electronic junk mail
	Acknowledged datagram	Registered mail
	Request-reply	Database query

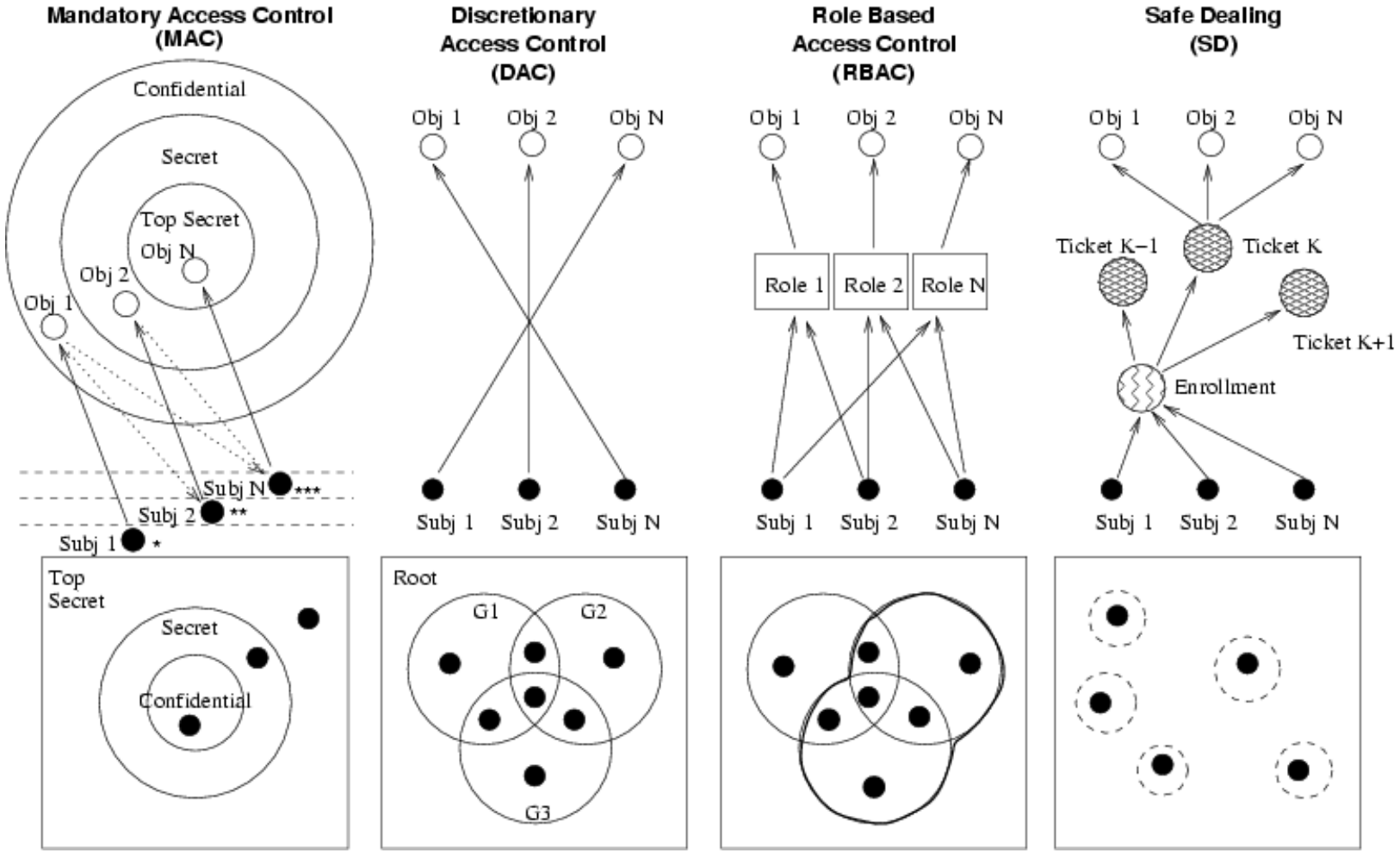
Das OSI Reference Model



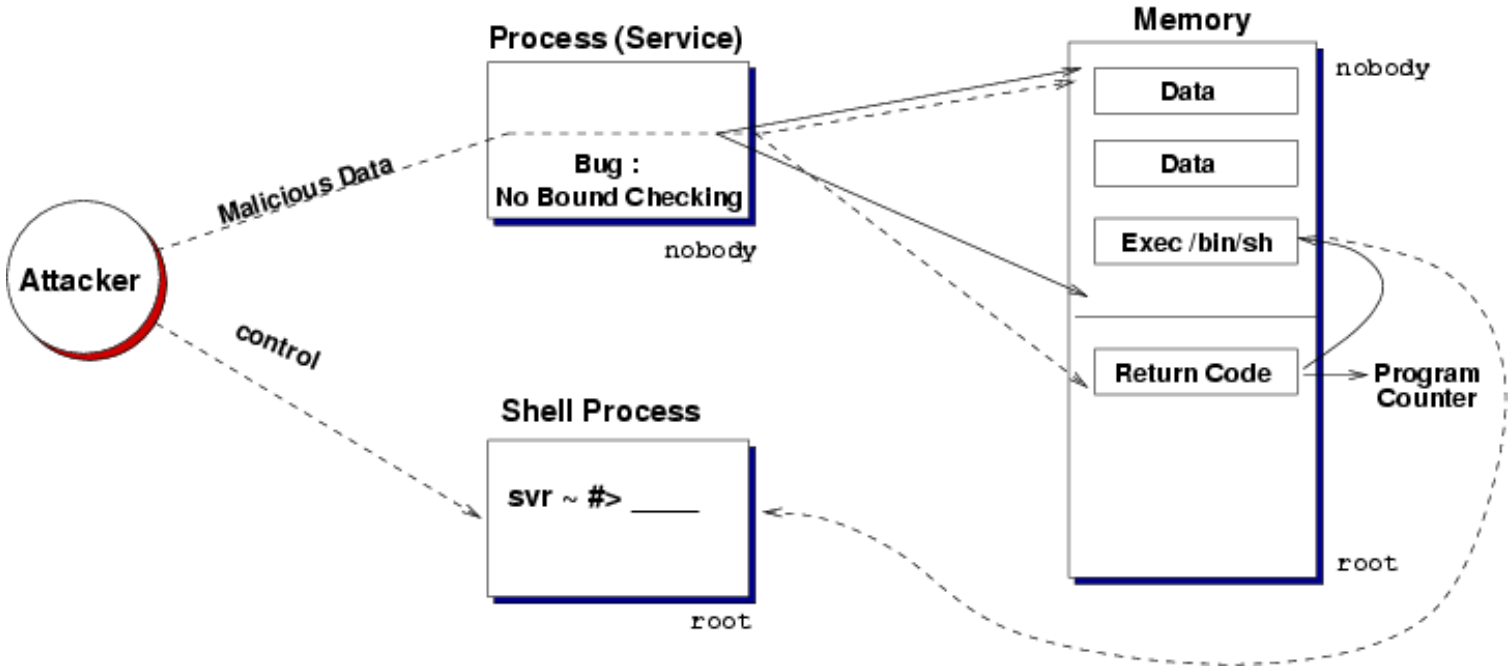
TCP/IP im Vergleich zu OSI



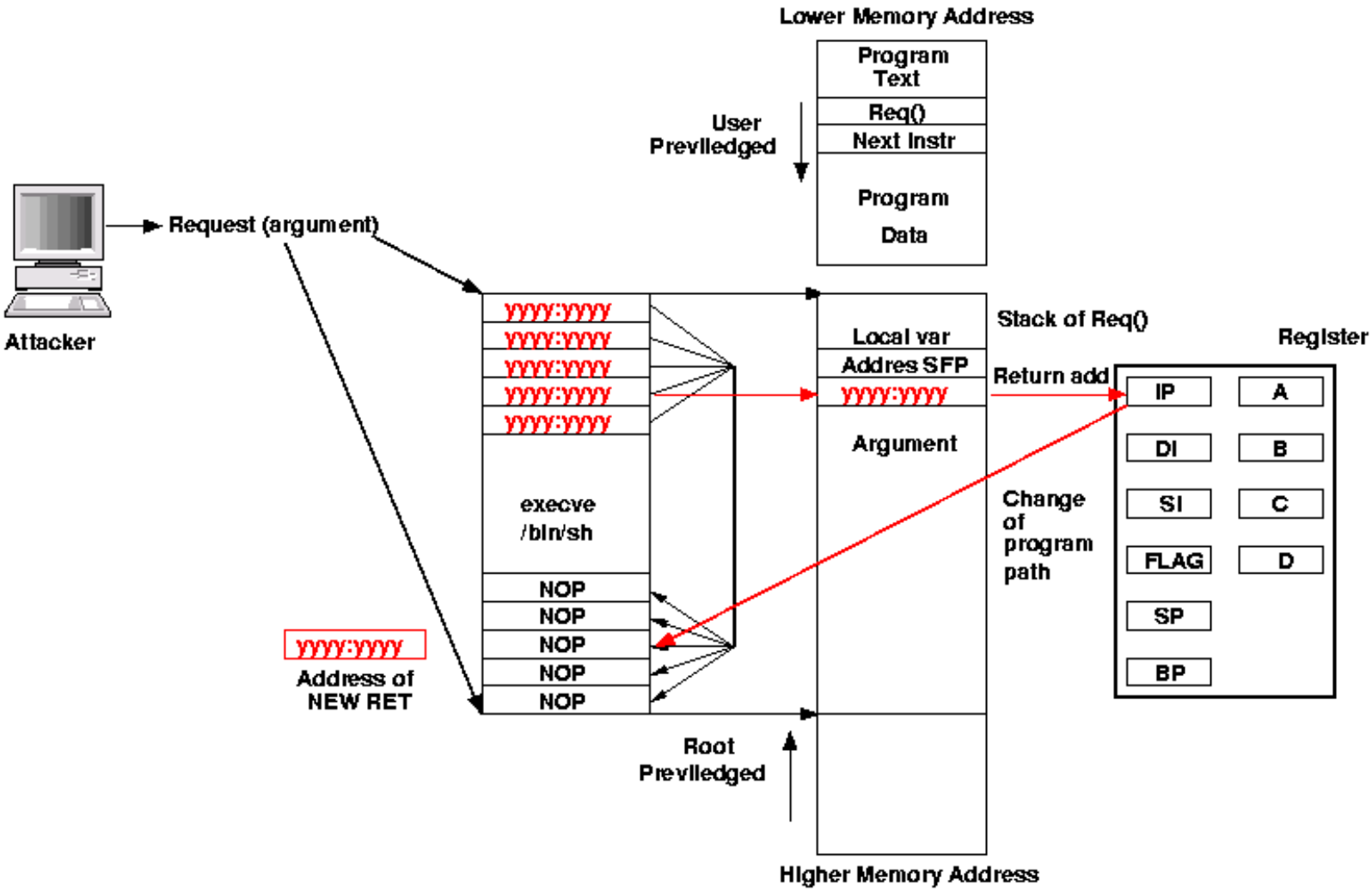
Access Control Modelle



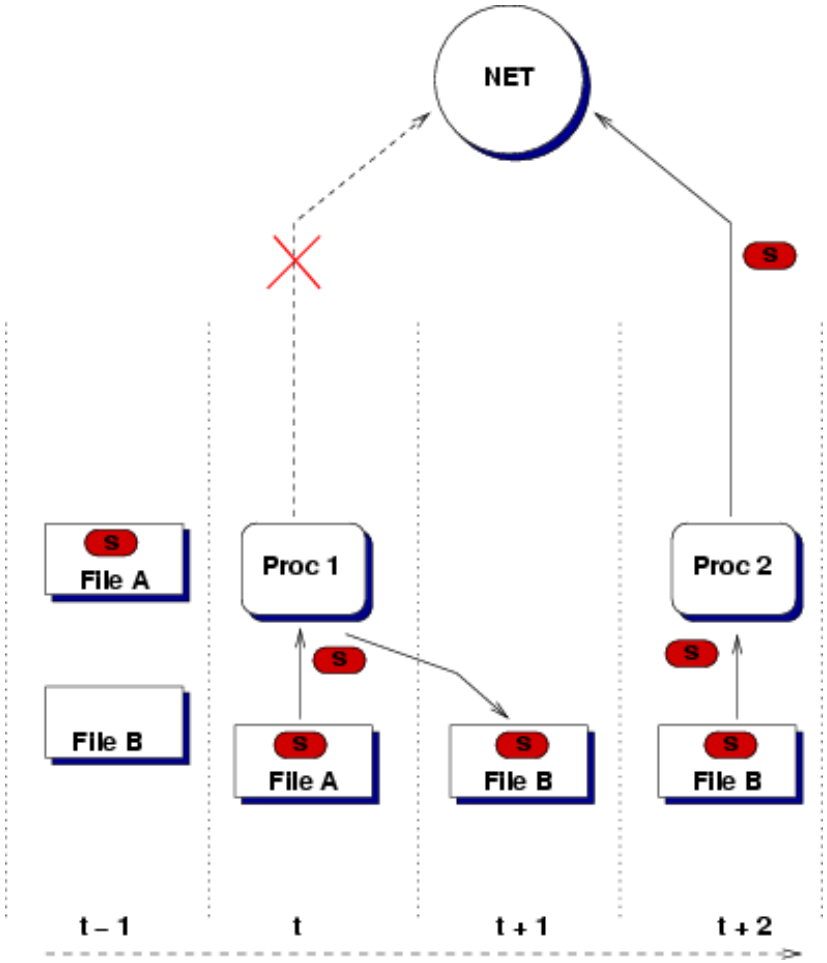
Buffer Overflow Schwäche statisch gezeigt



Buffer Overflow Angriff



Covert Channel



Remote und Local Exploits