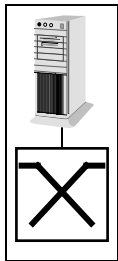


# Netze und Protokolle für das Internet



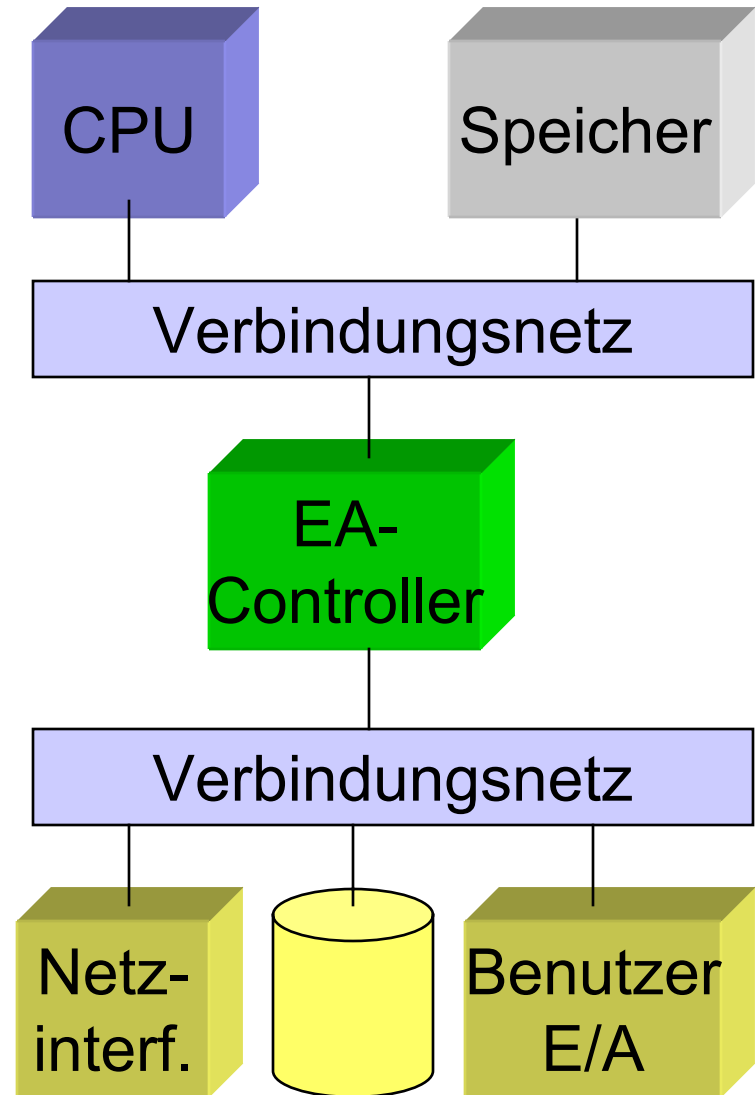
## 11. Implementierungsaspekte in Endsystemen

# Inhalt

- Endsystem-Hardware
- Endsystem-Software
- TCP/IP-Verarbeitung in BSD
  - Memory Buffer
  - Warteschlange
  - Paketverarbeitung
  - Funktionen
  - Transportprotokolle
- Probleme bei der Protokollverarbeitung
- TCP/IP-Verarbeitungs-Overhead
- Entwurfsprinzipien für Protokoll-Software
- Protokollverarbeitung
  - Datenmanipulation
  - Transfersteuerung
  - Asynchrone Steuerfunktionen
- Betriebssystemaspekte
  - Prozessverwaltung
  - Speicherverwaltung
- Protokoll-Software-Optimierungen
  - Protokoll-Bypass
  - Integrated Layer Processing
- Endsystemarchitekturen
  - Alternative Hardwarearchitektur
  - Parallele Protokollverarbeitung
  - Auslagerung von Protokollverarbeitung

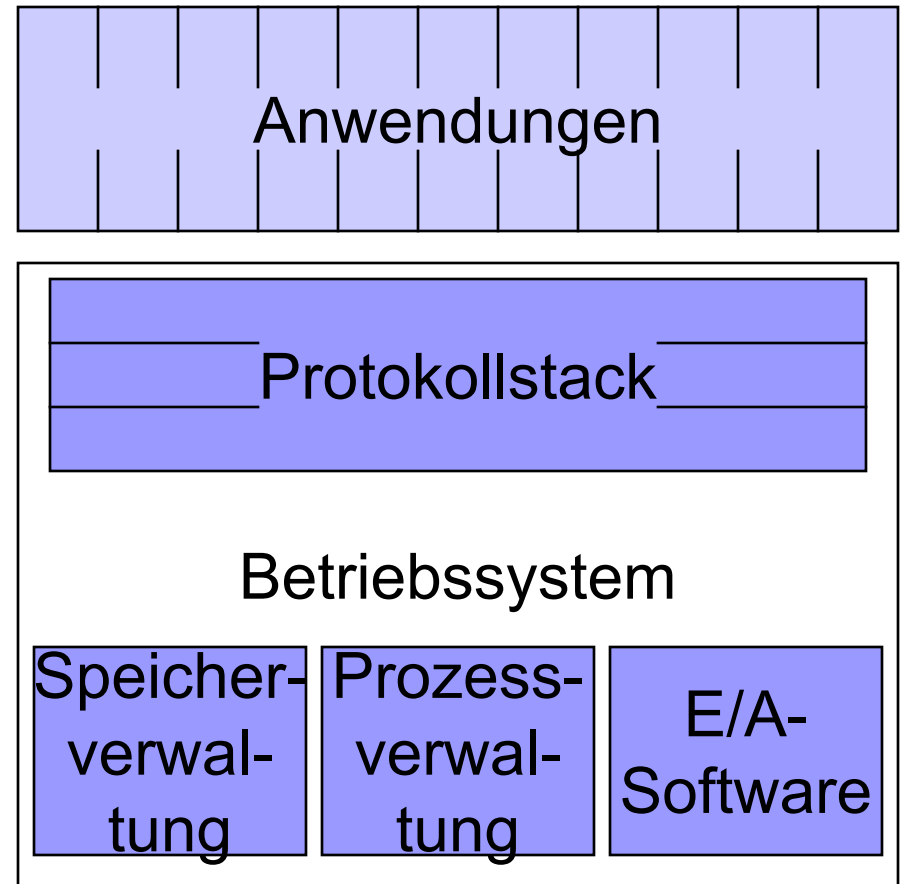
# Endsystem-Hardware

- Verbindungsnetze  
z.B. Busse
  - Prozessor-/Speicherbus
  - E/A-Bus
- Netzinterface
  - Verbindung des Endsystems mit dem Netz
  - ggf. Protokollverarbeitung
- Benutzer-E/A
  - Schnittstelle zu Peripheriegeräten, z.B. Monitor, Tastatur, Maus



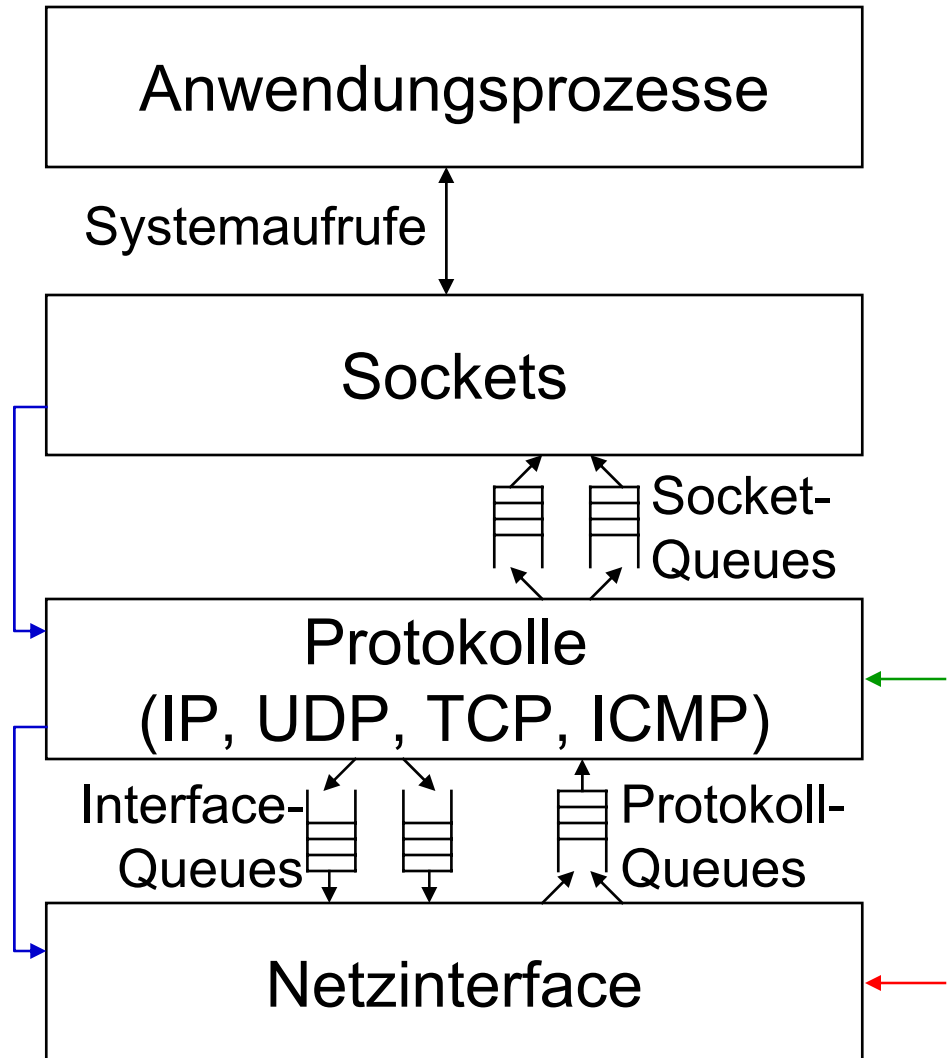
# Endsystem-Software

- **Protokollstack**
  - üblicherweise im Betriebssystemkern
- **Speicherverwaltung**
  - realer und virtueller Speicher
  - Adressabbildung
- **Prozessverwaltung**
  - Scheduling für Threads und Prozesse
- **E/A-Software**
  - E/A-Subsystem
  - Gerätetreiber zur Abbildung von E/A-Aufträgen auf E/A-Instruktionen
- **Sämtliche Komponenten bilden Ansatzpunkte zur Optimierung**



# TCP/IP-Verarbeitung in BSD

- Systemaufrufe durch Anwendungsprozesse
- Speichern der Daten in mbuf-Datenstrukturen
- Speichern der Daten in TCP für Übertragungswiederholungen
- Datentransfers
  - Senden
    - Anwendung → mbuf
    - mbuf → Netzinterface
  - Empfangen
    - Netzinterface → mbuf
    - mbuf → Anwendung
- Funktionsaufrufe zwischen Protokollen
- Software/Hardware-Interrupts

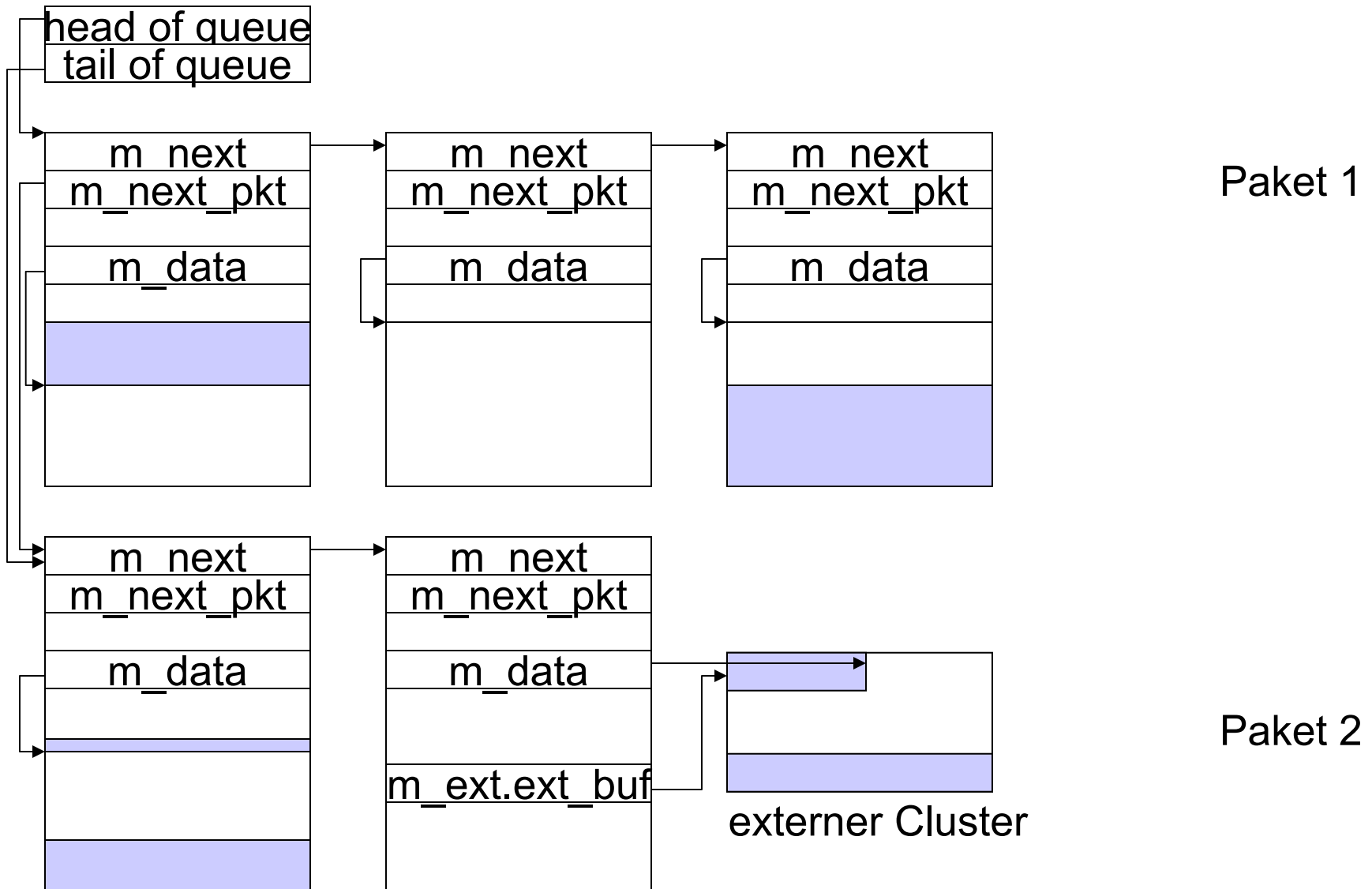


# Memory Buffer

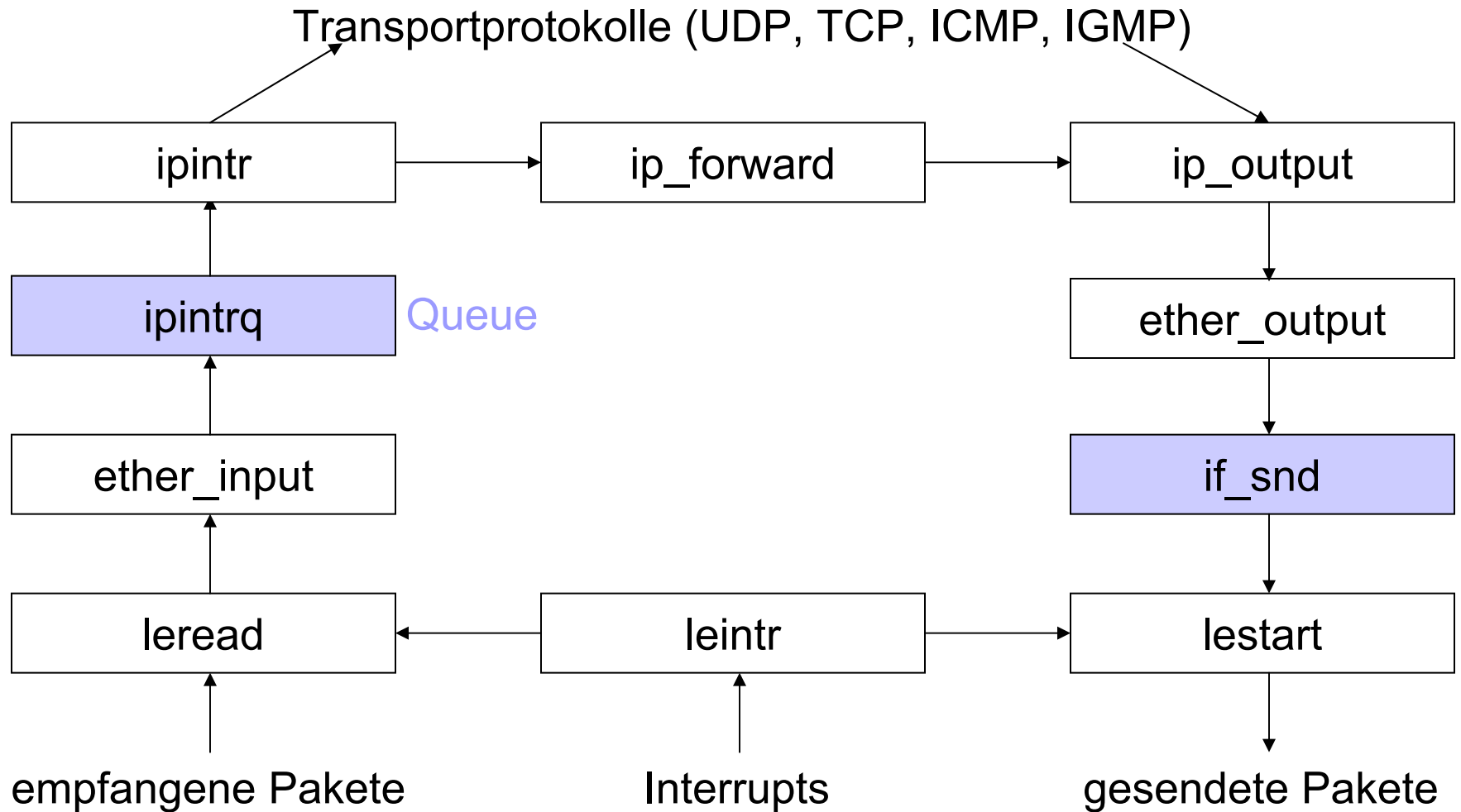
- mbuf =  
Datenstruktur zum Aufnehmen von Daten
- Grösse: 128/256 Bytes (BSD)
- Protokoll-Header und -Anhänge sollten ohne Kopieren angehängt werden können.  
→ mbuf-Daten können an beliebiger Stelle beginnen.
- Zeiger erlaubt Verkettung von mbufs.
- mbuf-Typen
  - Daten
  - Paket-Header
  - externer Puffer
  - Paket-Header und externer Puffer

m_next
m_nextpkt
m_len
m_data
m_type
m_flags
m_pkthdr.len
m_pkthdr.rcvif
m_ext.ext_buf
m_ext.ext_free
m_ext.ext_size

# Warteschlange



# Paketverarbeitung



# Funktionen

- `leintr`
  - Aufruf bei Interrupt (empfangenes oder gesendetes Paket)
- `leread`
  - Transfer eines empfangenen Pakets vom Interface in mbuf
- `lestart`
  - versucht neues Paket zu senden
- `ether_input`
  - Link-Level-Demultiplexing → ARP, IP
- `ether_output`
  - Voranstellen des Ethernet-Header vor Daten
- `if_snd`
  - Ausgangs-Queue
- `ipintrq`
  - Eingangs-Queue
- `ipintr`
  - IP-Eingangsverarbeitung
  - Aufruf durch Software-Interrupt
- `ip_output`
  - IP-Ausgangsverarbeitung
- `ip_forward`
  - Weiterleiten von IP-Paketen

# Transportprotokolle

- Empfangen
  - Ablegen der Daten in Socket-Puffer (udp/tcp\_input)
- Senden
  - Entnahme der Daten aus Socket-Puffer und Übergabe an IP (udp/tcp\_usrreq, udp/tcp\_output; ip\_output)
- Operationen nach Ablauf von TCP-Zeitgebern
  - Erzeugen von Quittungen
  - Übertragungswiederholungen

# Probleme bei der Protokollverarbeitung

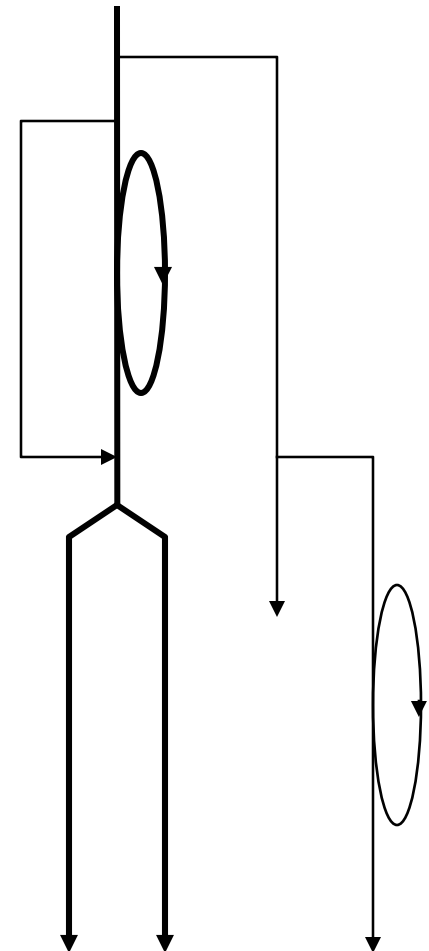
- Datenpfad
  - Kopiervorgänge zwischen Anwendungsprozessen, Betriebssystemkern und Netzinterface
- Kontrollpfad
  - Kontextwechsel durch (Software-)Interrupts

# TCP/IP-Verarbeitungs-Overhead

- Messungen durch V. Jacobson (1989)
- Sun-3/60 Workstation mit 20 MHz Motorola 68020, Speicherzykluszeit 250 ns
- Annahme: 1460-Byte-Pakete
- Byte-Funktionen
  - Benutzer-System-Kopie: 200  $\mu$ s
  - TCP Prüfsumme: 185  $\mu$ s
  - Netz-Speicher-Kopie: 386  $\mu$ s
- Paket-Funktionen
  - Ethernet-Treiber: 100  $\mu$ s
  - TCP, IP, ARP: 100  $\mu$ s
  - Betriebssystem: 240  $\mu$ s

# Entwurfsprinzipien für Protokoll-Software

- Betriebssystem sollte Netzanbindung als vorrangige Aufgabe ansehen.
- Geschichtete Protokollarchitekturen müssen nicht unbedingt in geschichteter Implementierung resultieren.
  - Funktionsaufrufe der einzelnen Schichten statt Prozesse
- Kopiervorgänge auf Daten oder andere Verarbeitungsschritte auf einzelnen Bytes sollten wo immer möglich vermieden werden.  
ideal: Zero-Copy-Interface
- Wie bei anderer Software sollte der kritische Pfad (Daten- und Kontrollpfad) optimiert werden.
  - entscheidend: Zeit der Instruktionen, nicht die Anzahl
  - wichtig: mehrmals zu durchlaufende Schleifen
    - Cache muss gross genug sein, um Schleife aufzunehmen.
    - Schleifen sollten in wenige virtuellen Seiten passen.
    - Datenstrukturen sollten in Cache-Zeilen passen.



# Protokollverarbeitung

Protokollverarbeitungsfunktionen können in drei Klassen eingeteilt werden (Clark, 1990):

- Datenmanipulation
- Transfersteuerung
- Asynchrone Steuerfunktionen

# Datenmanipulation

- Datenmanipulationsfunktionen lesen und schreiben Daten.
- Beispiele
  - Kopieroperationen vom/zum Netz
  - Zwischenspeicherung und Kopieroperationen von der / zur Anwendung
  - Zwischenspeicherung zur Übertragungswiederholung
  - Bitfehlererkennung und -korrektur (Prüfsummenberechnung)
  - Verschlüsselung und Entschlüsselung
  - Darstellungsfunktionen ((Un-)Marshalling)

# Transfersteuerung

- Steuerfunktionen, die meist auf Paketen arbeiten
- Kandidaten für den kritischen Pfad
- Beispiele
  - Fluss- und Staukontrolle
  - Erkennung verlorener oder nicht geordneter Pakete beim Empfänger
  - Quittierung empfangener Pakete
  - Multiplexen und Demultiplexen von Datenflüssen
  - Generieren von Zeitstempeln für zu sendende Pakete und Synchronisationserhaltung bei empfangenen Paketen
  - Einkapseln und Entkapseln von Protokolldateneinheiten
  - Segmentieren und Reassemblieren

# Asynchrone Steuerfunktionen

- Steuerfunktionen, die auf der Granularität von Verbindungen oder Flüssen auftreten, oder die asynchron zum Pakettransfer auftreten
- Beispiele
  - Verbindungsverwaltung
  - Routing
  - Sitzungssteuerung

# Betriebssystemaspekte

- Betriebssystem stellt Ressourcen in Form von CPU-Zyklen und Speicher den Anwendungen und unterstützenden Aktionen zur Verfügung.
- Die Betriebssystemkomponenten
  - Prozessverwaltung
  - Speicherverwaltung

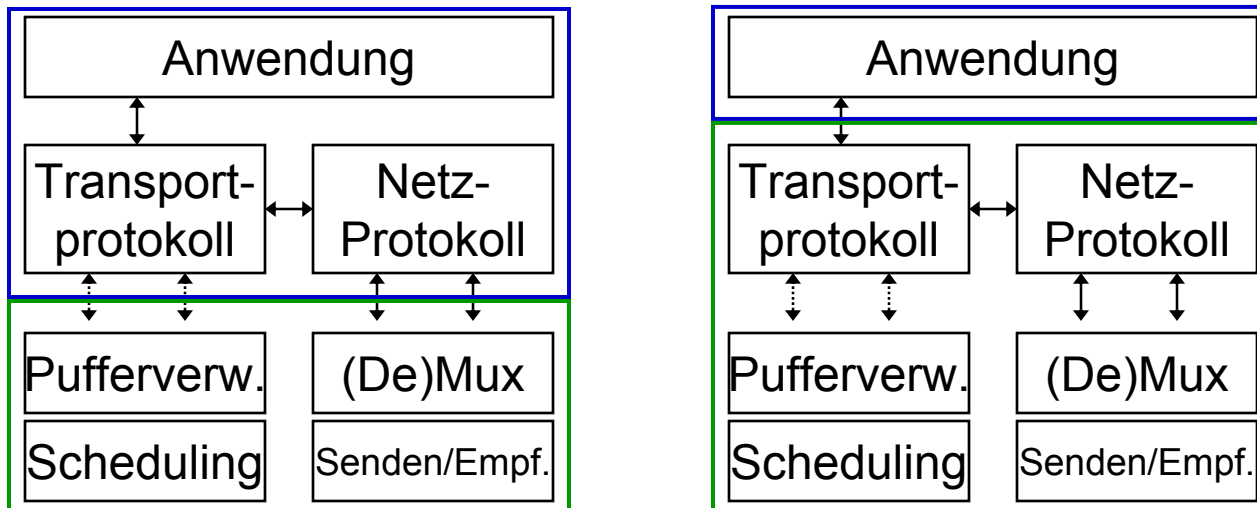
haben signifikanten Einfluss auf die Kommunikationsleistung.

# Prozessverwaltung

- Scheduling und Kontext-Switching
  - Gründe für Kontext-Switches
    - Senden und Empfangen von Paketen
    - Umschaltung zwischen Prozessen bei der Protokollverarbeitung
  - Kontext-Switches sollten auf ein Minimum reduziert werden, d.h. 1 Kontext-Switch pro Anwendungsdateneinheit (application data unit, ADU)
  - Leichtgewichtige Threads zur Reduzierung des Overheads
- Interrupts und Polling
  - Interrupts erlauben asynchrone Benachrichtigung über eingegangene Daten, erzeugen aber einen teuren Kontext-Switch.
  - Polling als Alternative, wenn Wissen über die Empfangsmuster verfügbar ist (Effizienz vs. Verzögerung).
  - Kombination von Interrupts und Polling möglich: Polling für Bursts.

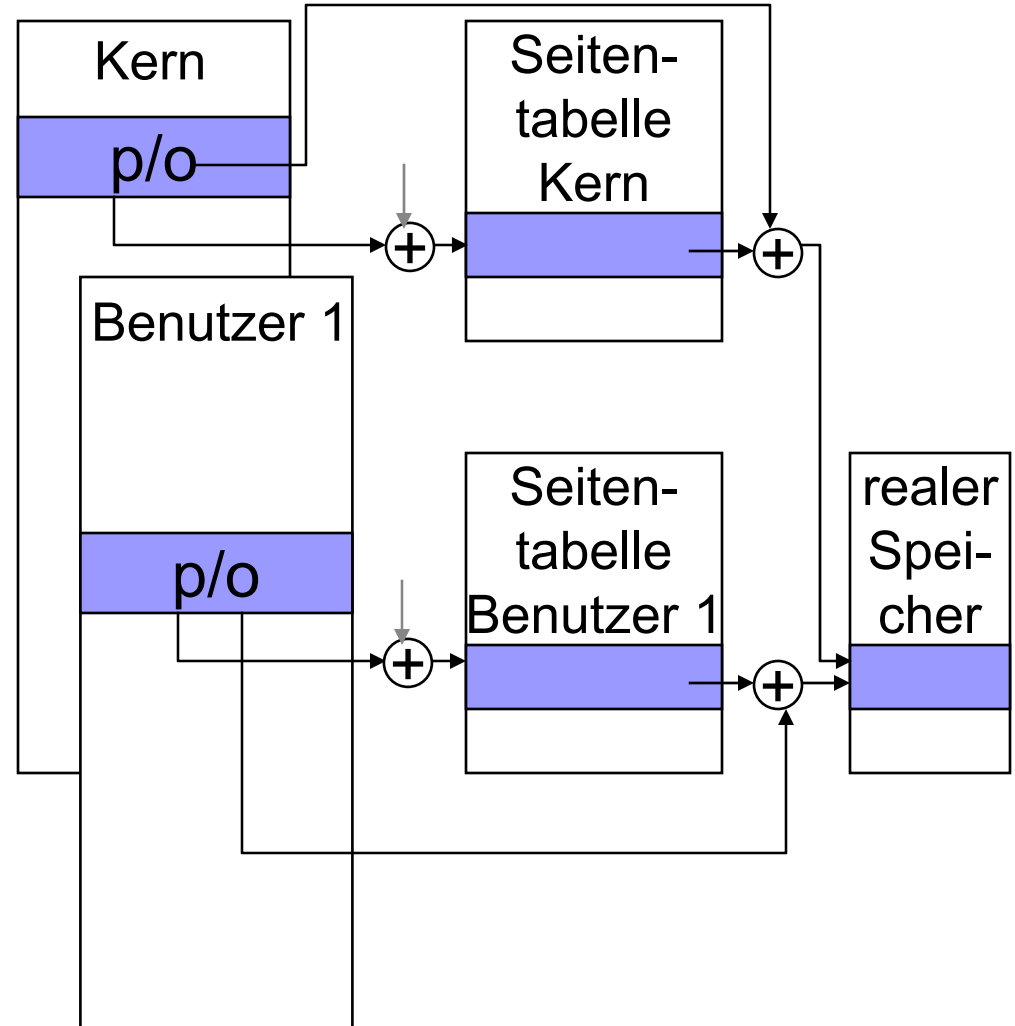
# Prozessverwaltung

- Protokollverarbeitung im **Kern** oder **Anwendungsadressraum**
- Protokolle nutzen zum Senden und Empfangen von Paketen eine Vielzahl von Systemdiensten, z.B. Pufferverwaltung, Scheduling, DMA, Zugriff auf I/O-Geräte
- Implementierung von Protokollen im Anwendungsadressraum würde zu mehreren Systemaufrufen für ein Paket führen.
- Vorteil von Anwendungsadressraum: Protokollentwicklung



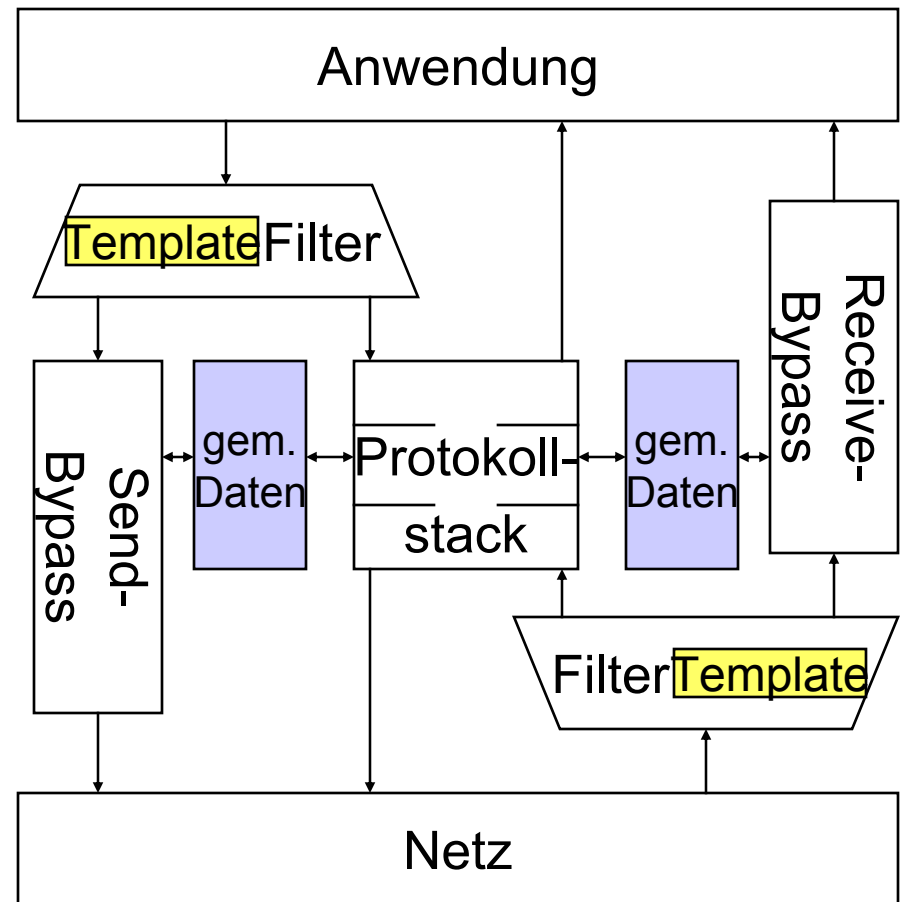
# Aufgaben der Speicherverwaltung

- Allokation von Speicher für zu sendende oder empfangene Pakete
- Teilen von Speicher, ggf. zwischen Anwendungen und Protokollverarbeitung
  - Abbilden von virtuellen Adressen verschiedener Prozesse auf den selben physikalischen Speicher (ggf. teuer)
  - Zeiger auf Seitentabelle
  - Bei empfangenen Daten werden Seiten nach Protokollverarbeitung in Anwendungsadressraum eingeblendet.
  - Problem beim Senden
    - Anwendung will Daten nach Senden modifizieren.
    - Übertragungswiederholung



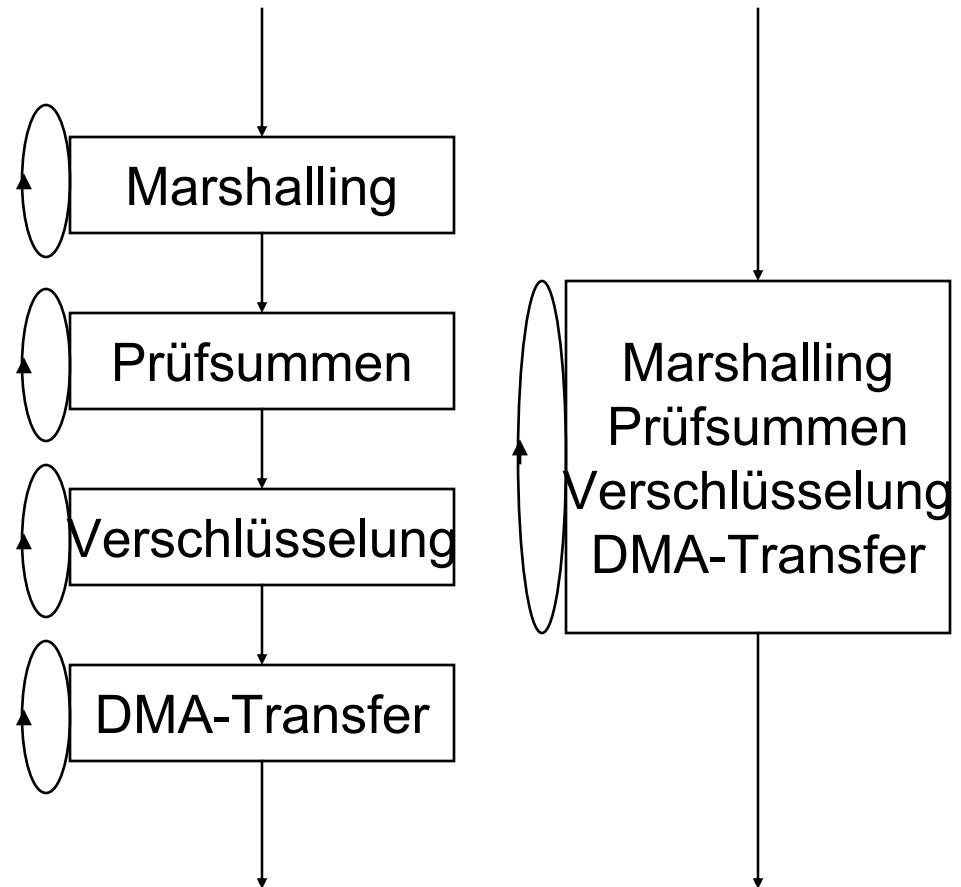
# Protokoll-Bypass

- Optimierter Code für übliche Fälle in Bypass
  - Header-Prediction beim Empfangen
    - erwartete Sequenznummer
    - keine Optionen
    - keine Änderung der Fenstergrösse
  - vorberechnetes Template für zu sendende Daten
    - Ports
    - nächste Sequenznummer
    - Quittungsnummer
- Spezialfälle werden durch Protokoll-Stack bearbeitet.
- Protokoll-Stack und Bypass arbeiten auf gemeinsamen Daten.



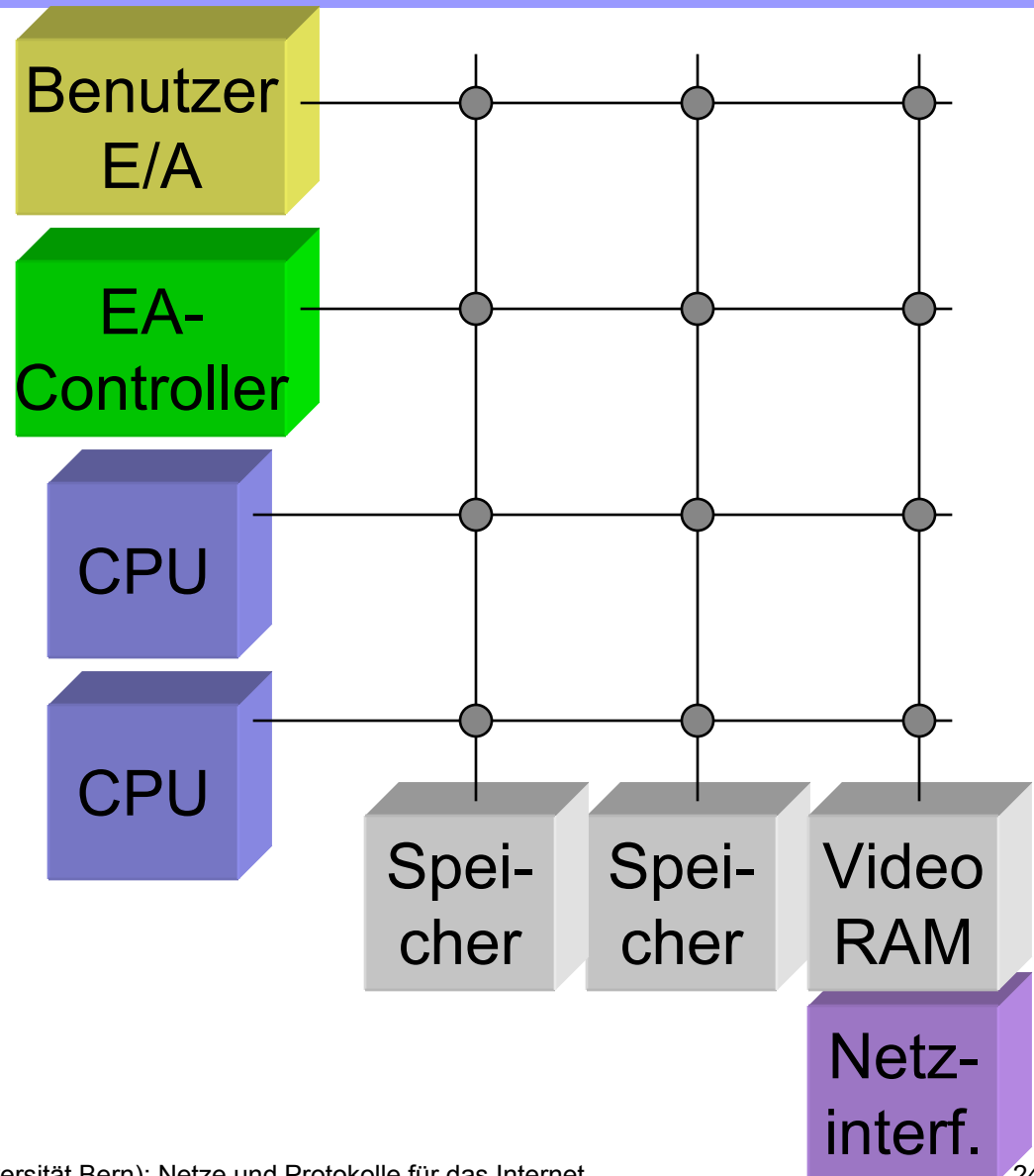
# Integrated Layer Processing

- Geschichtete Implementierung
  - Daten müssen mehrmals zwischen Hauptspeicher und Cache transportiert werden.
- Integration der Datenmanipulationsfunktionen in eine Schleife
  - Einmaliges Lesen der Daten vom Hauptspeicher in den Cache
  - Operationen auf dem Cache
  - Schreiben der Daten vom Cache in den Hauptspeicher
  - ! Schleife sollte in den Cache passen



# Alternative Hardwarearchitektur

- Verbindung von Komponenten über leistungsfähige Verbindungsstrukturen (vgl. Parallelrechner)
- gemeinsamer Speicher für Anwendungen und Netzinterface
  - Multiport-Speicher, z.B. Video-RAM
  - teuer, daher eher kleiner Umfang
  - beschränkte Nutzung durch Anwendungen



# Parallele Protokollverarbeitung

- Parallelisierungsmöglichkeiten
  - Prozessor pro Schicht
  - Prozessor pro Verbindung
  - Prozessor pro Paket
  - Prozessor für Sende- und Empfangsrichtung
- gemeinsame Speicher
  - für Daten und Verbindungszustände

