


Softwareentwicklung für eingebettete Systeme

Bachelorstudiengang Technische Informatik
Hochschule Pforzheim

Vorlesung 4
Sommersemester 2014

Dipl.-Ing.(FH) Marc Jüttner

IPC und RTOS

- IPC in RTOS unterscheidet sich nicht von „normalem“ IPC
 - Queues, Semaphore, Mutexe
- Prozess → Task 
- Beispiel...

Interprozessorkommunikation

- Kommunikation zwischen Prozessoren
 - Signalisierung, Events
 - Datenaustausch
- Synchronisation von Prozessen auf unterschiedlichen Prozessoren
 - Triggern von Signalverarbeitungsschritten
- Oft Shared Memory erforderlich

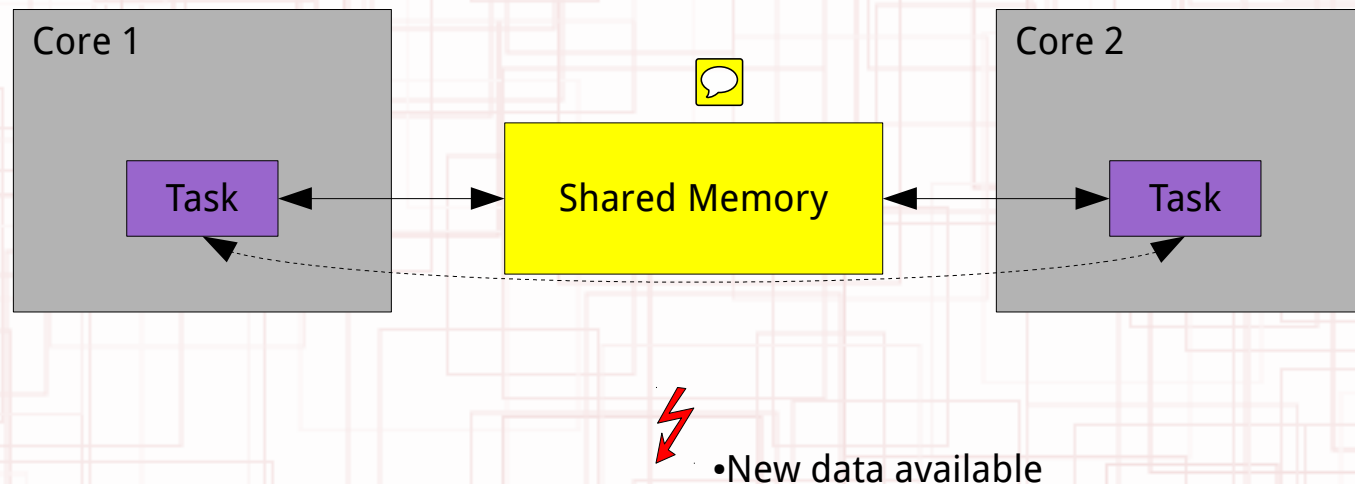
Direkte Kommunikation

- Ähnlich einer Pipe/Queue
- Hardwareunterstützung
 - Interrupt zur Signalisierung
 - Abstraktion auf Taskebene



Indirekte Kommunikation

- Shared Memory für Datenaustausch
- Signalisierungsmechanismus erforderlich



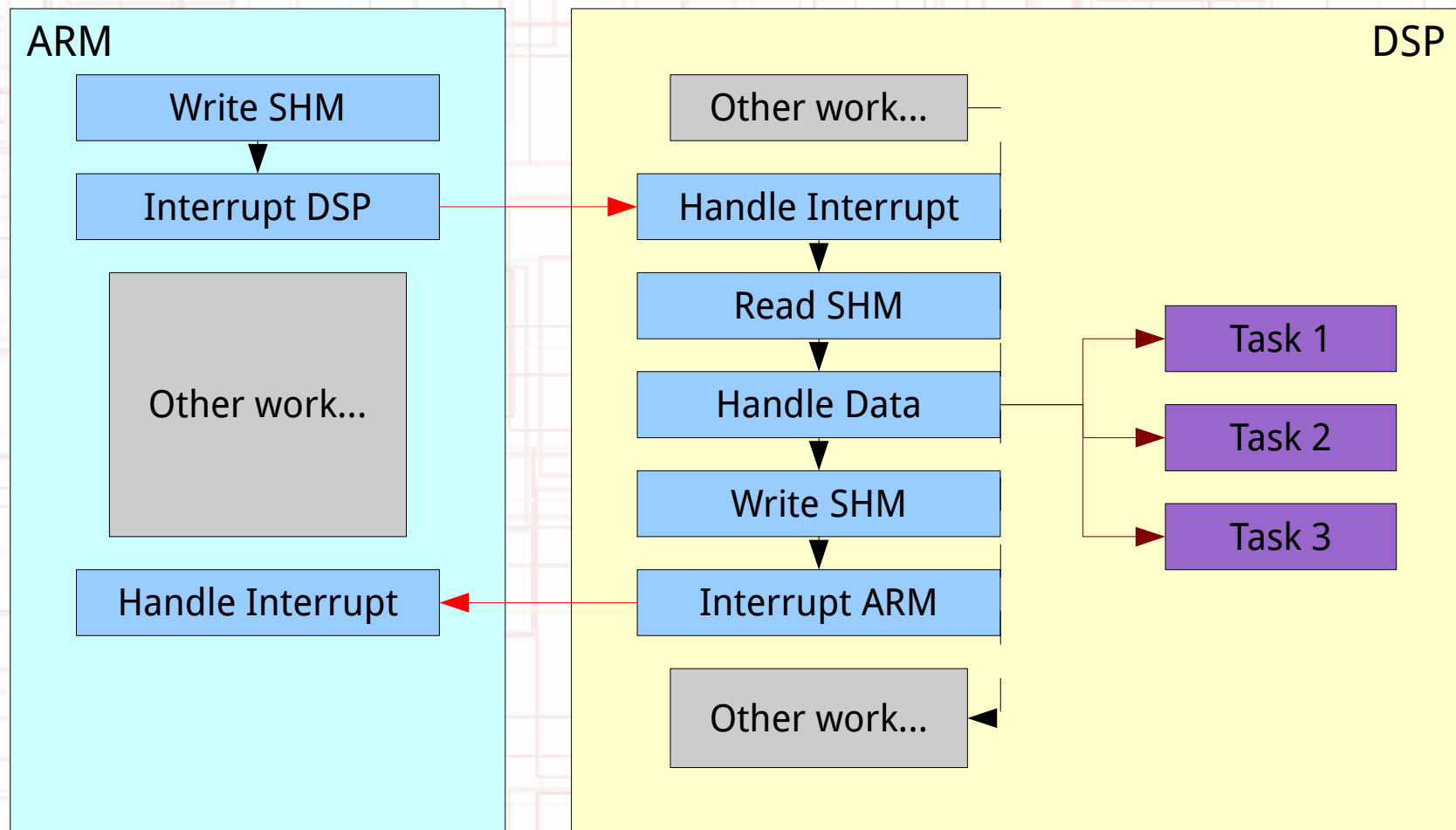
Signalisierung

- Polling von Bits im Speicher
 - Verschwendet CPU-Zeit
- Interprozessorinterrupts
- Hardwaremailboxen
 - FIFO-Mechanismus
 - Signalisierung mittels Interrupts

Polling

- Statusabfrage mit atomaren Operationen
- Polling in längeren Zeitabständen verringert Verlust von CPU-Zeit
 - Höhere Latenz!
- Cacheverwaltung muss korrekt erfolgen

OMAP L138



Pseudocode

ARM process

```
memcpy(src, shm, size);  
interrupt_dsp();  
wait_for_event(evt);  
...  
process(shm);  
...
```

ARM interrupt handler

```
...  
signal_event(evt);  
...
```

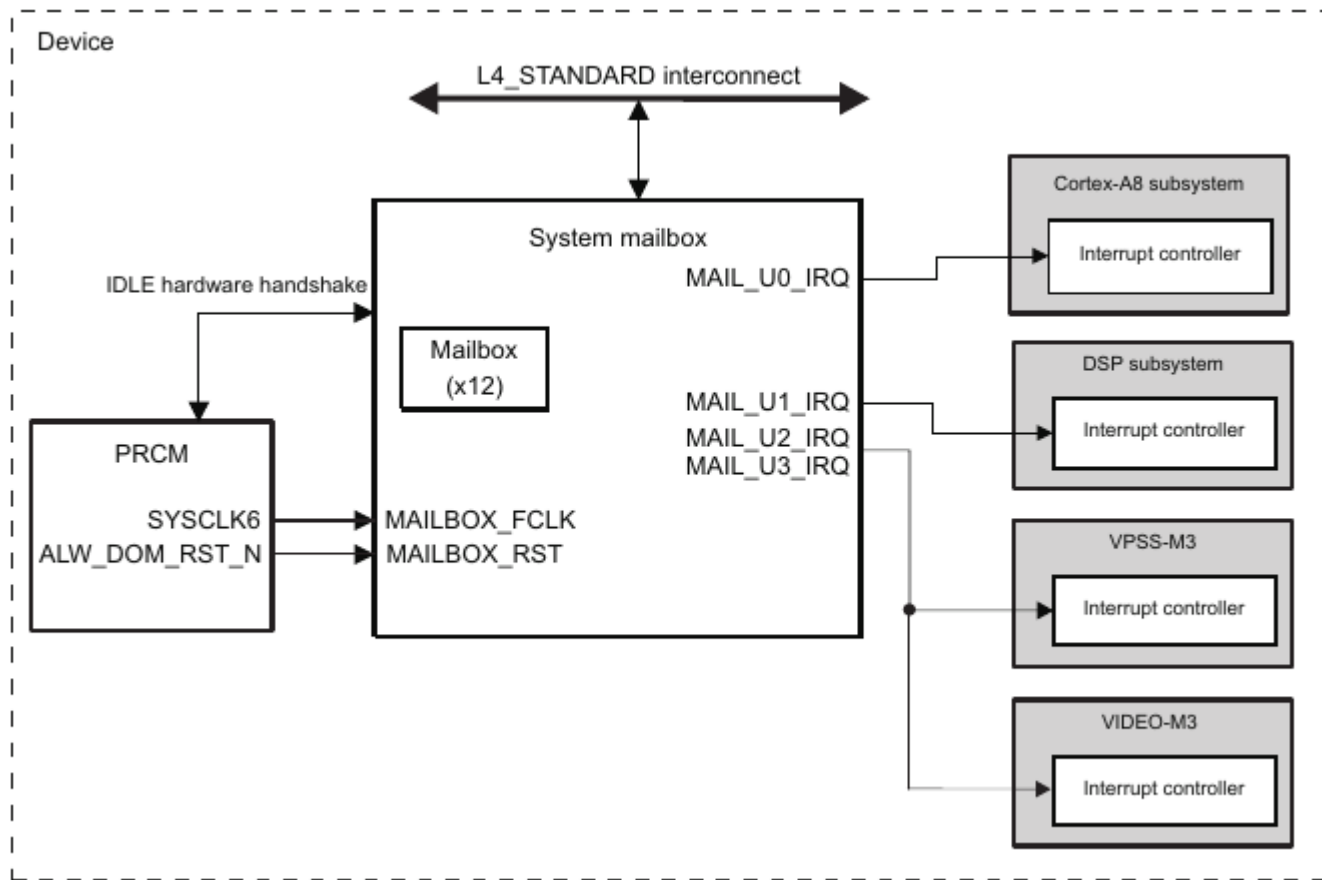
DSP interrupt handler

```
signal_task(tsk);
```

DSP task

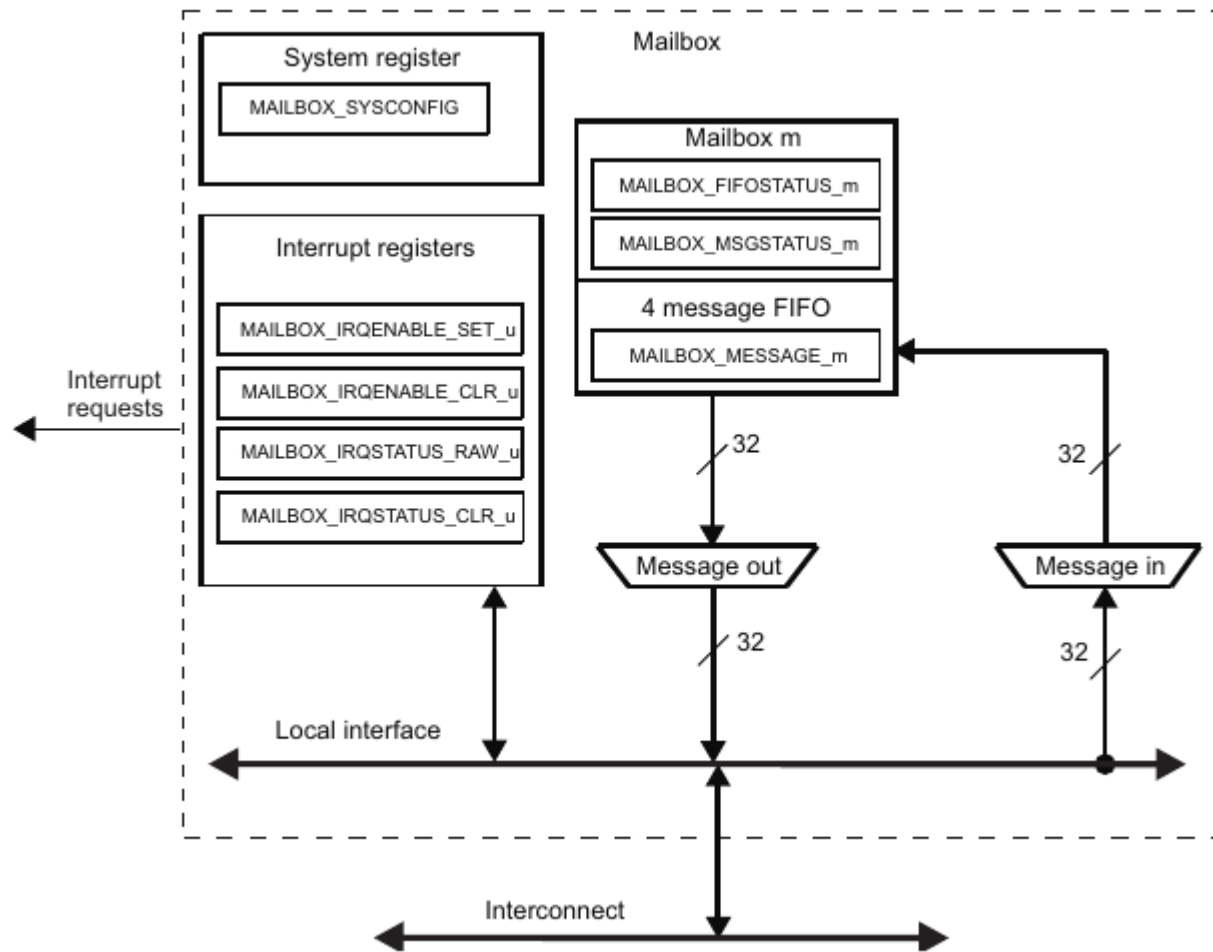
```
msg = (struct msg*)shm;  
switch(msg->code)  
{  
    /* call handler fn */  
}  
interrupt_arm();
```

Mailboxen DM8148



Quelle: ti.com,
SPRUGZ8E,
DM8148 TRM

Mailbox DM8148



Quelle: ti.com,
SPRUGZ8E,
DM8148 TRM

Senden via Mailbox

Table 1-114. Sending a Message (Polling Method)

Step	Register/Bitfield/Programming Model	Value
IF : Is FIFO full ?	MAILBOX_FIFOSTATUS_m[0].FIFOFULL MB	=1h
Wait until at least one message slot is available	MAILBOX_FIFOSTATUS_m[0].FIFOFULL MB	=0h
ELSE		
Write message	MAILBOX_MESSAGE_m[31:0].MESSAG EVALUEMBM	----h
ENDIF		

Table 1-115. Sending a Message (Interrupt Method)

Step	Register/Bitfield/Programming Model	Value
IF : Is FIFO full ?	MAILBOX_FIFOSTATUS_m[0].FIFOFULL MB	=1h
Enable interrupt event	MAILBOX_IRQENABLE_SET_u[1+ m*2]	1h
User(processor) can perform another task until interrupt occurs		
ELSE		
Write message	MAILBOX_MESSAGE_m[31:0].MESSAG EVALUEMBM	----h
ENDIF		

Quelle: ti.com,
SPRUGZ8E,
DM8148 TRM

Empfangen via Mailbox

Table 1-116. Receiving a Message (Polling Method)

Step	Register/Bitfield/Programming Model	Value
IF : Number of messages is not equal to 0	MAILBOX_MSGSTATUS_m[2:0].NBOFM SGMB	!=0h
Read message	MAILBOX_MESSAGE_m[31:0].MESSAG EVALUEMBM	----h
ENDIF		

Table 1-117. Receiving a Message (Interrupt Method)

Step	Register/Bitfield/Programming Model	Value
Enable interrupt event	MAILBOX_IRQENABLE_SET_u[0 + m*2]	1h
User(processor) can perform another task until interrupt occurs		

Quelle: ti.com,
SPRUGZ8E,
DM8148 TRM

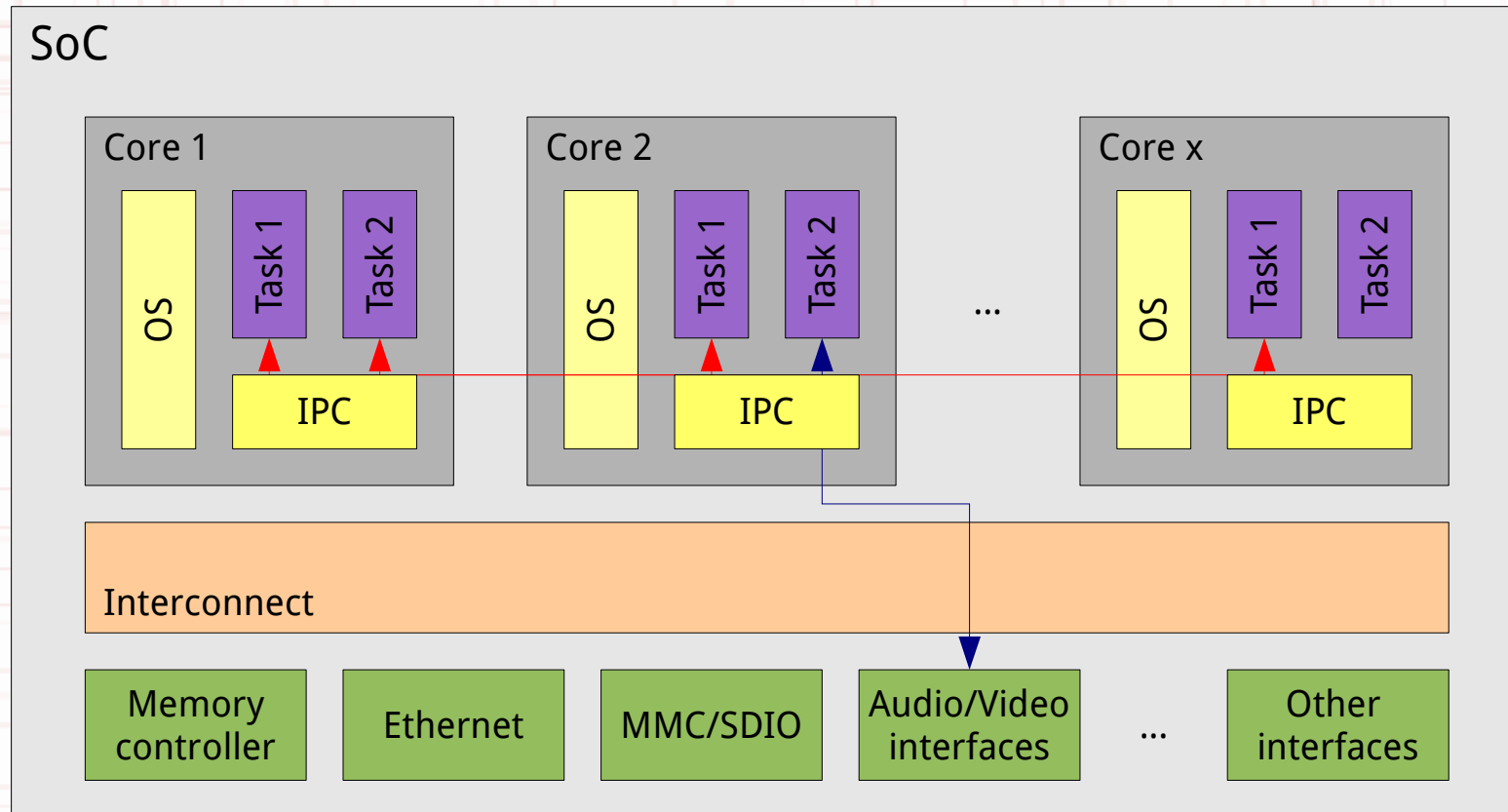
Szenarien

- Kommunikation
 - Messagequeues, Mailboxen
- Gemeinsamer (dynamischer) Speicher
 - Verwaltung erforderlich!
- Synchronisation
 - Spinlocks, Semaphore

Implementierungen

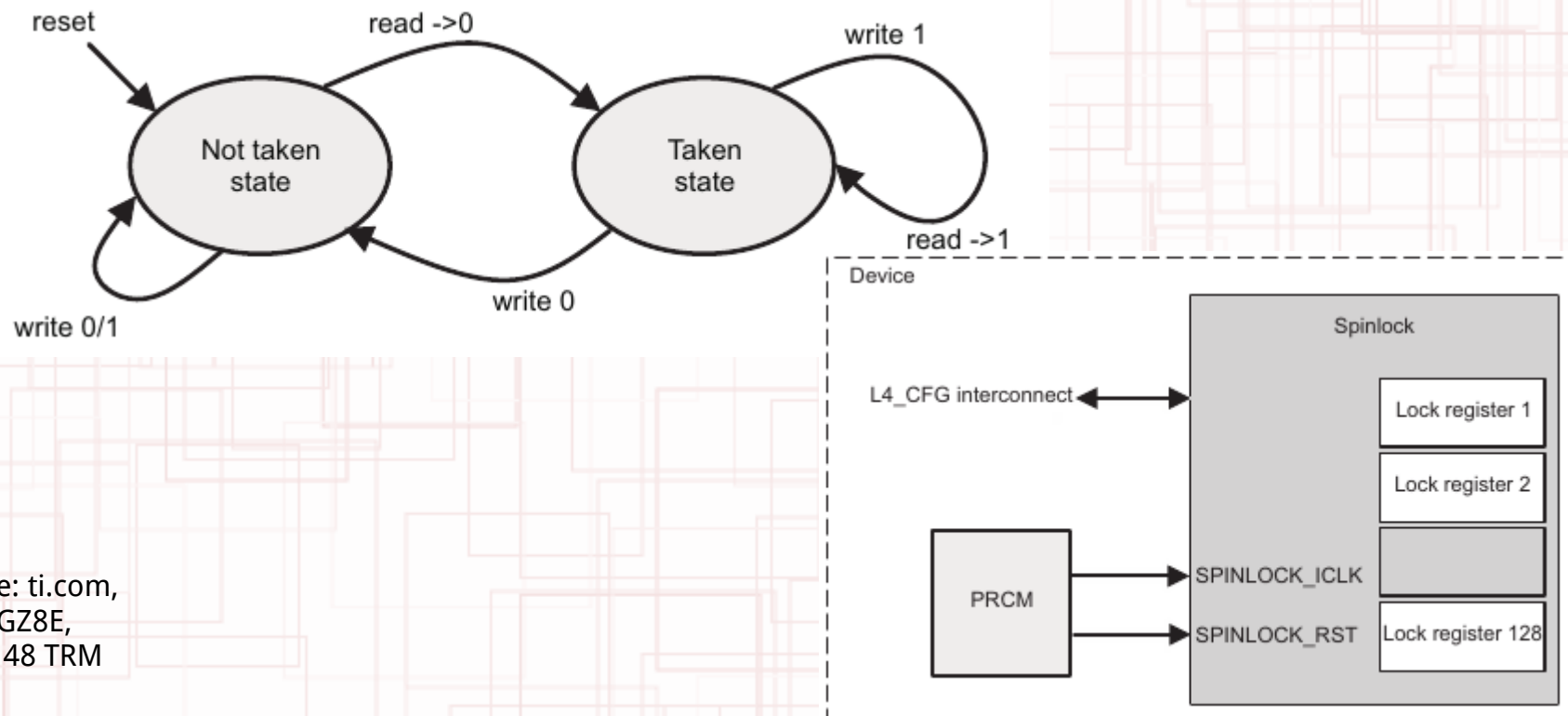
- Kapselung der Hardware
 - Synchronisationsmechanismen sind nicht auf allen SoCs gleich
- Bereitstellung einer generischen API
 - Verbergen der tatsächlichen Systemdetails vor dem Entwickler
- Generische Verwaltung von Erzeugern und Verbrauchern

Multicore-IPC-Infrastruktur



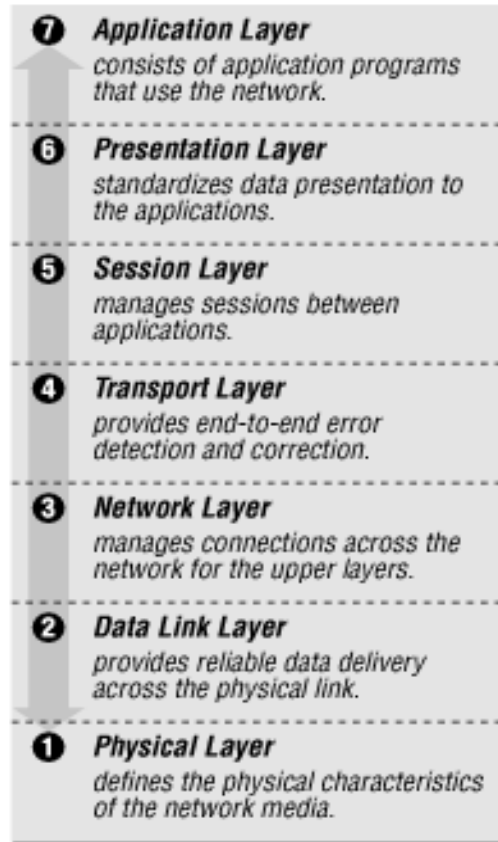
Hardware-Spinlocks

- Atomare Operationen, spezielle Register



Quelle: ti.com,
SPRUGZ8E,
DM8148 TRM

OSI-Referenzmodell

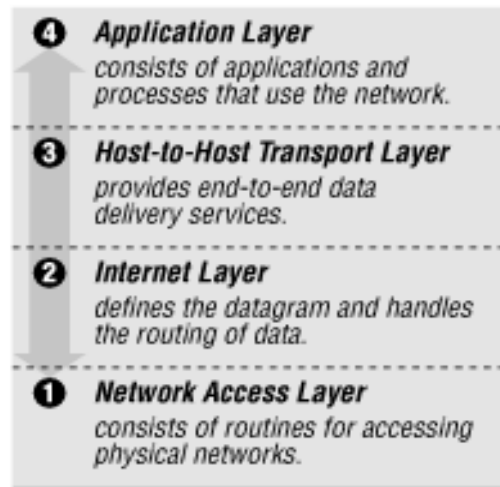


Quelle: Craig Hunt, TCP/IP Network Administration

OSI-Modell vs. TCP/IP

- OSI-Modell erläutert Schichtenkonzept
- Keine Passgenauigkeit für TCP/IP
- Sinnvoll für das Verständnis, für TCP/IP kann ein einfacheres Modell herangezogen werden...

TCP/IP-Schichtenmodell

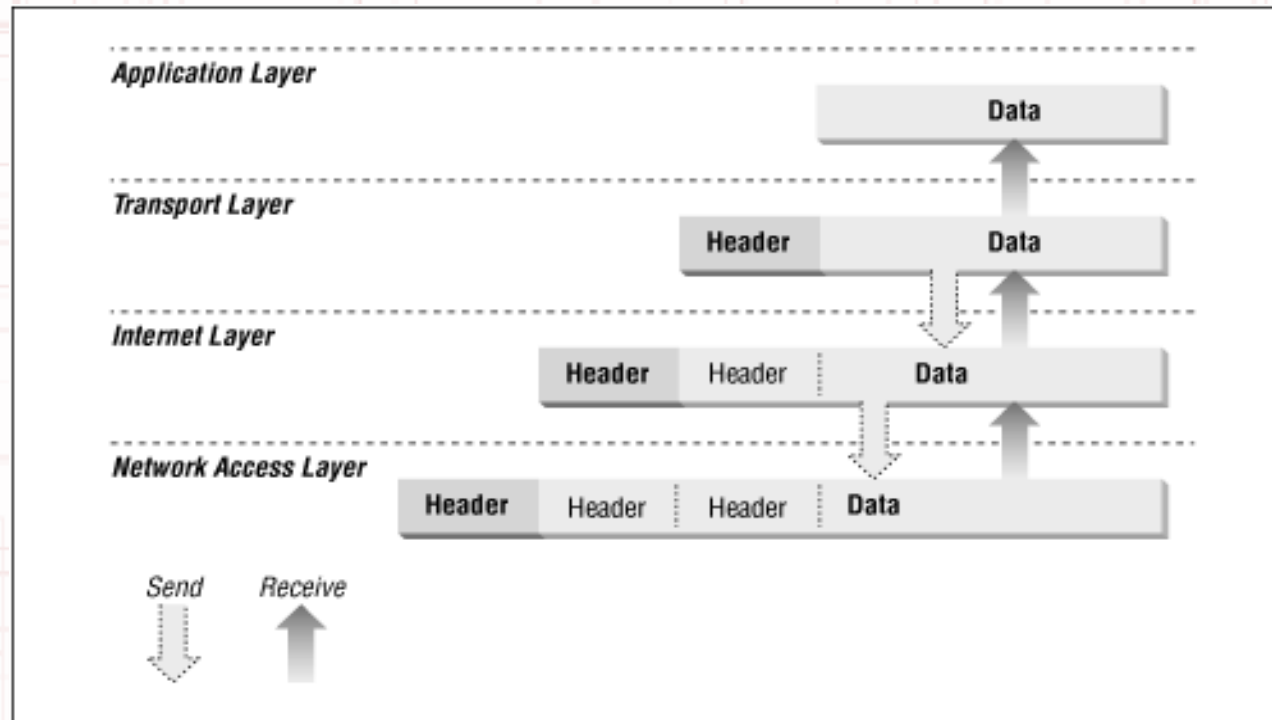


Quelle: Craig Hunt, TCP/IP Network Administration

Datenfluss

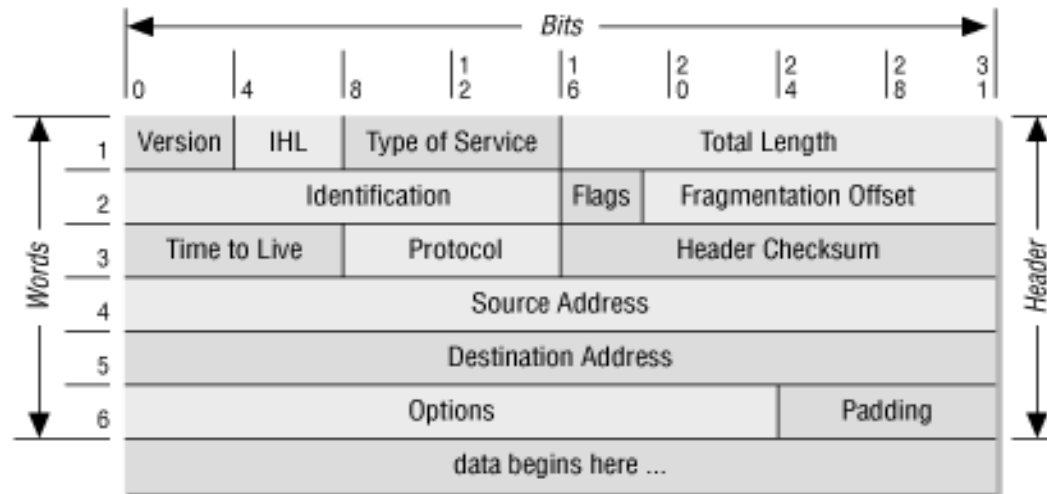
- Logische Kommunikation ist *peer-to-peer*
- Tatsächliche Kommunikation durchläuft Schichten
 - Daten werden von Schicht zu Schicht weitergereicht
 - Jede Ebene fügt Transportinformationen hinzu
 - Beim Empfang: Umgekehrte Reihenfolge

Datenkapselung



Quelle: Craig Hunt, TCP/IP Network Administration

IP-Datagramm

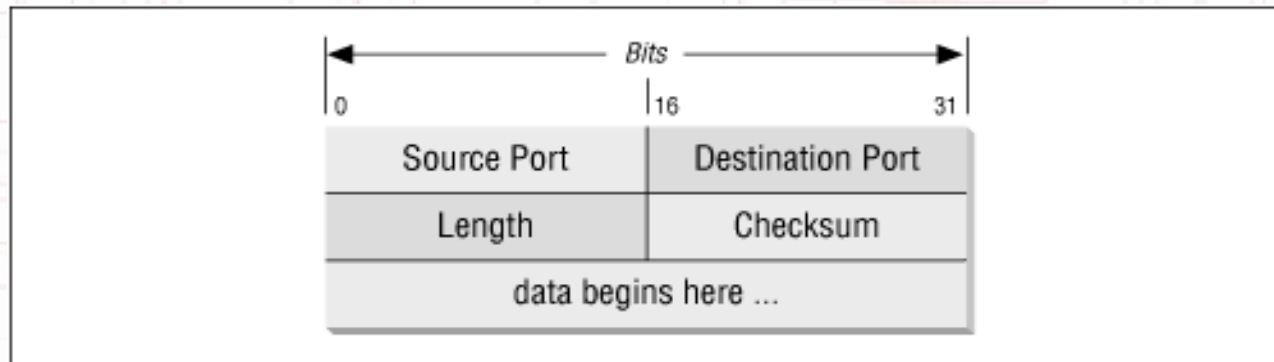


Quelle: Craig Hunt, TCP/IP Network Administration

UDP

- User Datagram Protocol
- Verbindungsloses Protokoll
- Übertragung nicht sicher
 - Unzuverlässiges Protokoll
- Verwendet Ports zur Dienstzuordnung

UDP-Datagramm



Quelle: Craig Hunt, TCP/IP Network Administration

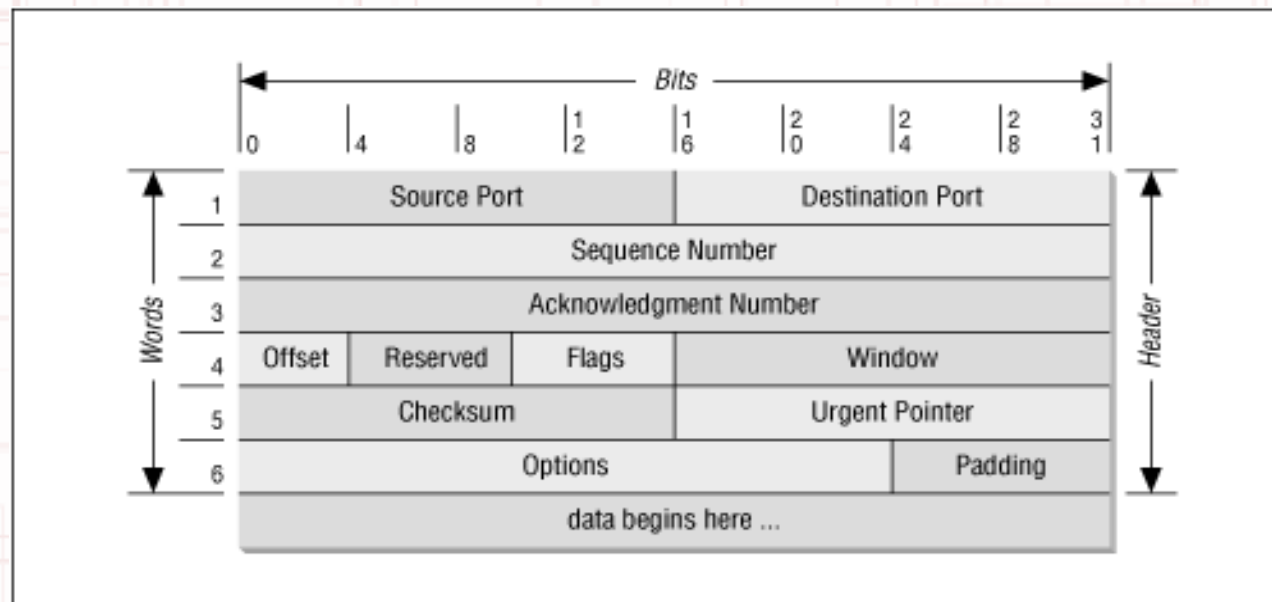
UDP-Merkmale

- Geringer Aufwand bei kleinen Datenmengen
 - Kein Verbindungsaufbau, kein Handshake
- Gute Abbildung von Query/Response
 - Antwort ist Bestätigung
 - Timeout ist Fehler
- Schnelle Übertragung durch geringen Overhead

TCP

- Transmission Control Protocol
- Verbindungsorientiertes Protokoll
- Garantierte Auslieferung der Daten
 - Automatisches Nachsenden
 - Zuverlässiges Protokoll
- Verwendet Ports zur Dienstzuordnung

TCP-Datagramm



Quelle: Craig Hunt, TCP/IP Network Administration

TCP-Merkmale

- Daten werden vollständig und in der richtigen Reihenfolge zugestellt
- Fehlende Pakete werden nachgesendet
- Streamorientierung
 - Pakete und deren Größe sind aus Anwendungssicht nicht sichtbar

TCP und UDP im Vergleich

- TCP

- Zuverlässig
- Automatische Nachsendung
- Transparente Streams
- Aufwändig
- Langsam

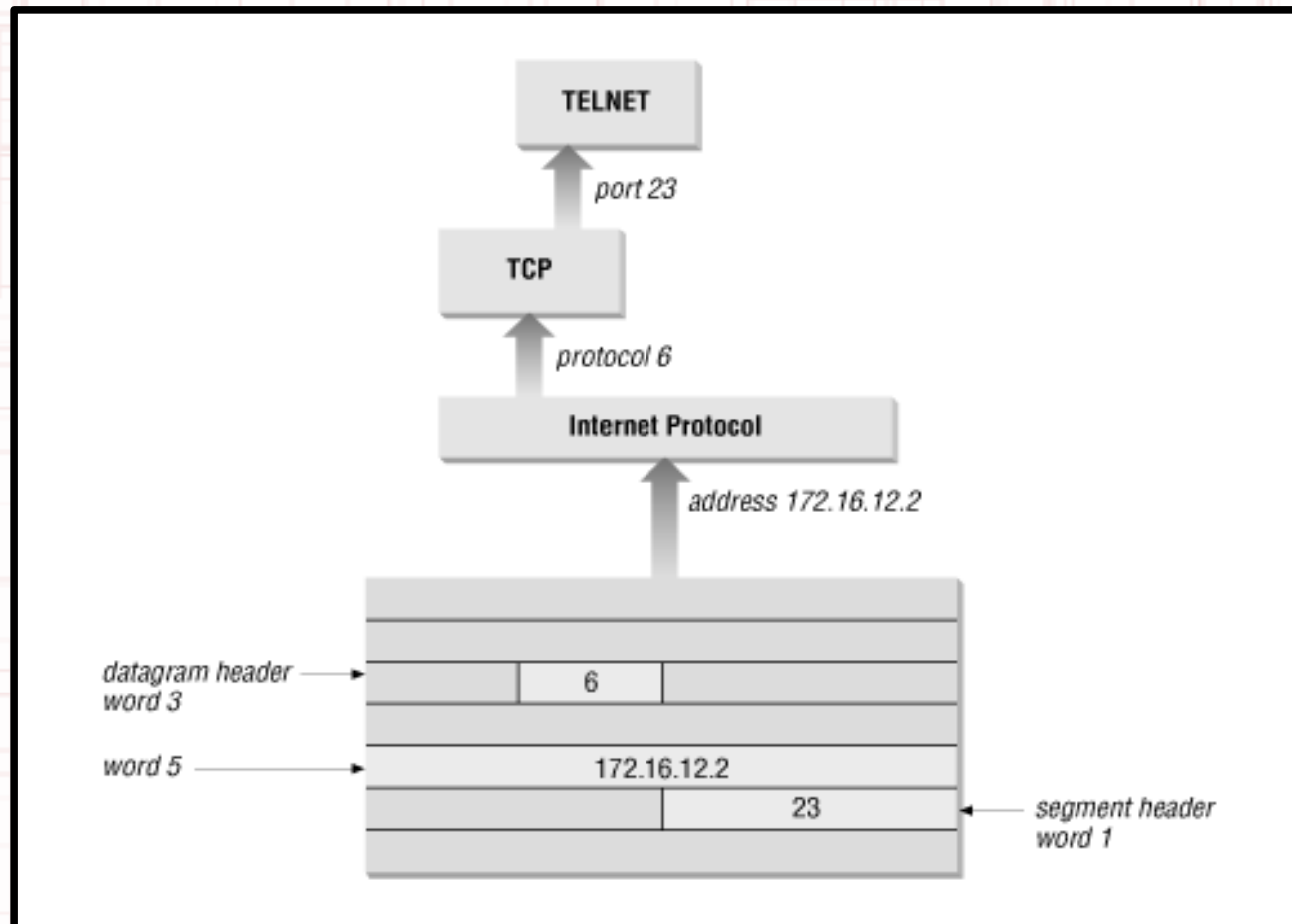
- UDP

- Unzuverlässig
- Manuelle Nachforderung
- Datagramm-übertragung
- Einfach
- Schnell

Dienstzuordnung

- Mehrere Prozesse nutzen ein Netzwerkinterface
- IP: Protokollnummer identifiziert darüberliegendes Protokoll
- UDP/TCP: Portnummern identifizieren, welcher Prozess ein Paket erhält

Dienstzuordnung

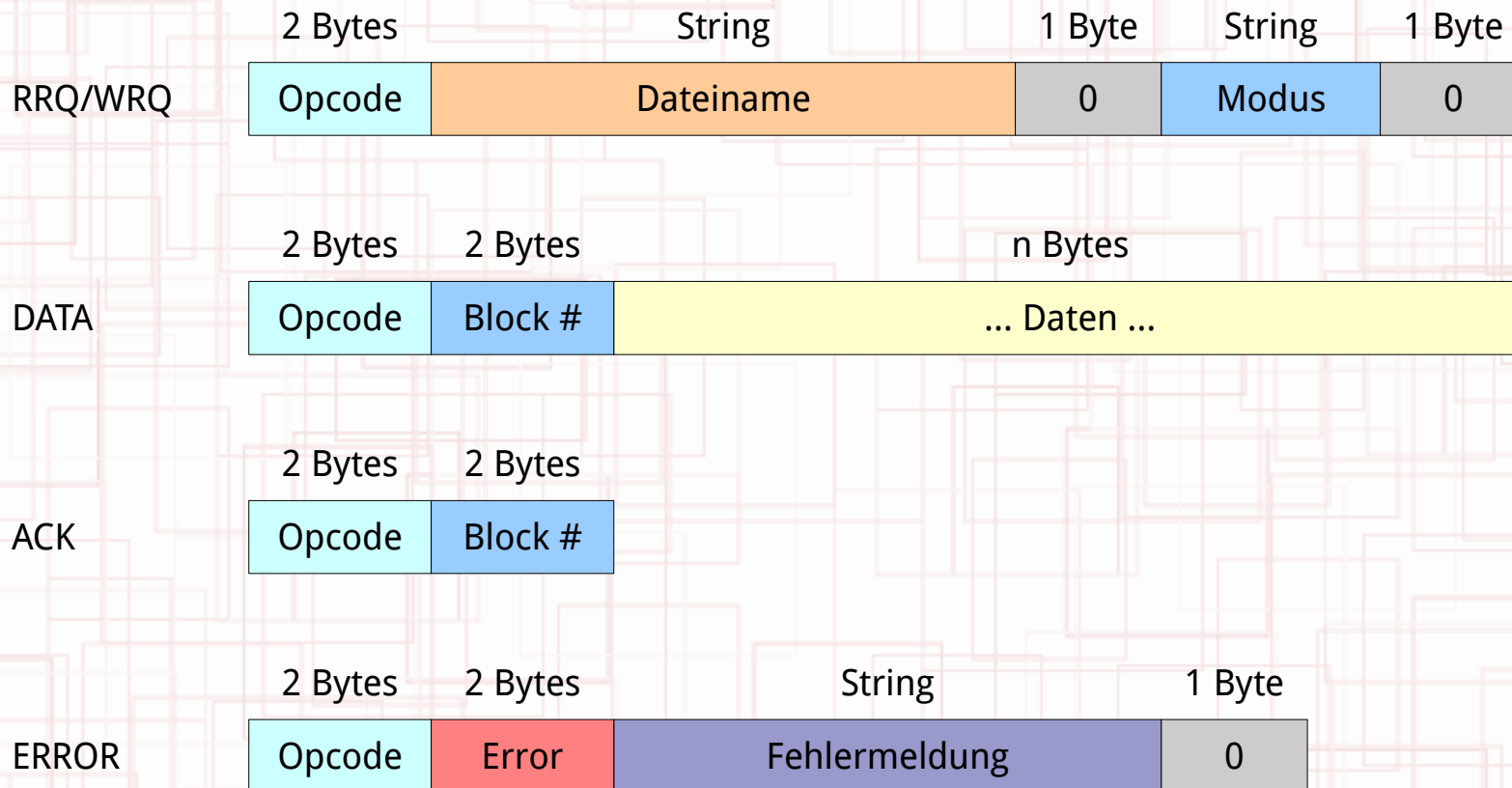


Quelle: Craig Hunt, TCP/IP Network Administration

Toplevelprotokolle

- Netzwerksockets bieten eine abstrakte Kommunikationsschnittstelle
- Verwendung der Schnittstelle muss anwendungsspezifisch definiert werden
 - Protokollddefinition erforderlich

Trivial File Transfer Protocol



Sockets

- Bidirektionale Softwareschnittstelle
 - Interprozess- oder Netzwerkkommunikation
 - Virtueller Kommunikationsendpunkt
- Vollduplexfähig
- Weitgehend plattformunabhängig
 - Verbreitete API: BSD-Sockets-API

Zugriff auf Sockets

- Sockets sind Dateideskriptoren ähnlich
 - `read()`, `write()`
- Netzwerkschnittstellen sind keine Dateien!
- Öffnen und schließen benötigt spezielle Funktionen
 - Verbindung zum Ziel erforderlich
 - Wahl des Protokolls

Socketvarianten

- Internet-Sockets: Abbildung des IP-Protokolls
- Unix domain sockets: Spezielle Dateien
 - IPC-Sockets, nur lokale Kommunikation
- X.25, TokenRing, Netbios, ...
- → Abbildung von Protokollfamilien

Typen von Sockets

- Datagrammsockets (UDP, IPC-Sockets)
 - Versand einzelner Nachrichten, Datagramme
 - Verbindungslose Kommunikation
- Streamsockets (TCP)
 - Kontinuierlicher Bytestrom
 - Verbindungsorientiert

Kommunikation via Sockets

- Verbindungsaufbau
 - Zieladresse, Protokoll
- Datenaustausch
 - `send()`, `recv()`, `write()`, `read()`, ...
- Verbindungsabbau
- Fehlerbehandlung ist wichtig, um Kommunikationsfehler zu erkennen und zu behandeln!

Beispiele...

- Beispiele stammen aus Beej's Guide to Network Programming
 - Datagram client and server
 - Stream server for use with Telnet