Scheduling

Zeitplanen periodischer Prozesse

Zeitplanung periodischer Prozesse

- Annahmen für präemptives Scheduling
  - Alle Prozesse treten periodisch mit einer Frequenz f_i auf.
  - Die Frist eines Prozesses entspricht dem nächsten Startpunkt.
  - Sind die maximalen Ausführungszeiten e_i bekannt, so kann leicht errechnet werden, ob ein ausführbarer Plan existiert.
  - Die für einen Prozesswechsel benötigten Zeiten sind vernachlässigbar.
  - Alle Prozesse sind unabhängig.
- Eine sehr gute Zusammenfassung zu dem Thema Zeitplanung periodischer Prozesse liefert Giorgio C. Buttazzo in seinem Paper „Rate Monotonic vs. EDF: Judgement Day“ (http://www.cas.mcmaster.ca/~downd/rtsj05-rmedf.pdf).

Scheduling

Problem: Prioritätsinversion

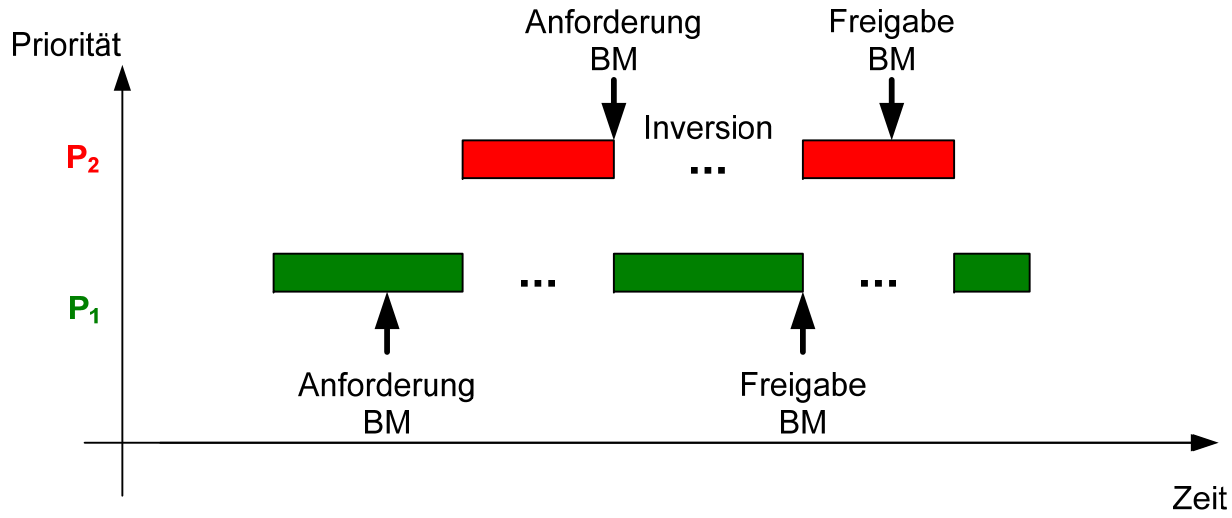
Motivation des Problems

- Selbst auf einem Einprozessoren-System mit präemptiven Scheduling gibt es Probleme bei voneinander abhängigen Prozessen.
- Abhängigkeiten können diverse Gründe haben:
  - Prozesse benötigen Ergebnisse eines anderen Prozesses
  - Betriebsmittel werden geteilt
  - Es existieren kritische Bereiche, die durch Semaphoren oder Monitoren geschützt sind.
- Gerade aus den letzten zwei Punkten entstehen einige Probleme:
  - Die Prozesse werden unter Umständen unabhängig voneinander implementiert $\Rightarrow$ das Verhalten des anderen Prozesses ist nicht bekannt.
  - Bisher haben wir noch keinen Mechanismus zum Umgang mit blockierten Betriebsmitteln kennengelernt, falls hochpriore Prozesse diese Betriebsmittel anfordern.

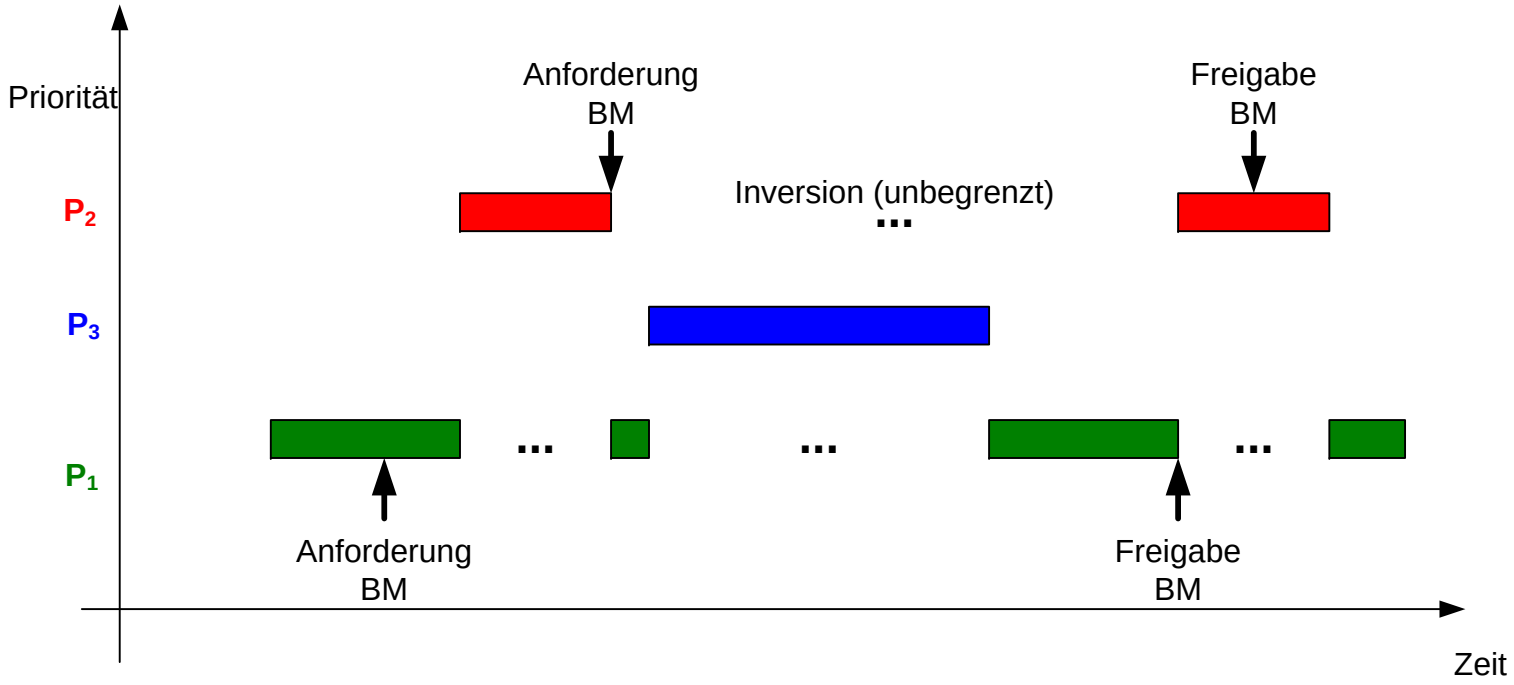
Prioritätsinversion

- **Definition**: Das Problem der **Prioritätsinversion** bezeichnet Situationen, in denen ein Prozess mit niedriger Priorität einen höherpriorisierten Prozess blockiert.
- Dabei unterscheidet man zwei Arten der Prioritätsinversion:
  - **begrenzte** (bounded) Prioritätsinversion: die Inversion ist durch die Dauer des kritischen Bereichs beschränkt.
  - **unbegrenzte** (unbounded) Prioritätsinversion: durch weitere Prozesse kann der hochpriorisierte Prozess auf unbestimmte Dauer blockiert werden.
- Während das Problem der begrenzten Prioritätsinversion aufgrund der begrenzten Zeitdauer akzeptiert werden kann (muss), ist die unbegrenzte Prioritätsinversion in Echtzeitsystemen unbedingt zu vermeiden.

Begrenzte Inversion



Unbegrenzte Inversion



Reales Beispiel: Mars Pathfinder

- **System**: Der Mars Pathfinder hatte zur Speicherung der Daten einen Informationsbus (vergleichbar mit Shared Memory). Der Informationsbus war durch einen binären Semaphore geschützt. Ein Bus Management Prozess verwaltete den Bus mit hoher Priorität. Ein weiterer Prozess war für die Sammlung von geologischen Daten eingeplant. Dieser Task lief mit einer niedrigen Priorität. Zusätzlich gab es noch einen Kommunikationsprozess mittlerer Priorität.
- **Symptome**: Das System führte in unregelmäßigen Abständen einen Neustart durch. Daten gingen dadurch verloren.
- **Ursache**: Der binäre Semaphore war nicht mit dem Merkmal zur Unterstützung von Prioritätsvererbung (siehe später) erzeugt worden. Dadurch kam es zur Prioritätsinversion. Ein Watchdog (Timer) erkannte eine unzulässige Verzögerung des Bus Management Prozesses und führte aufgrund eines gravierenden Fehlers einen Neustart durch.

Ansätze zur Lösung der Prioritätsinversion

- Es existieren verschiedene Ansätze um das Problem der unbegrenzten Prioritätsinversion zu begrenzen:
  - Prioritätsvererbung (priority inheritance)
  - Prioritätsobergrenzen (priority ceiling)
  - Unmittelbare Prioritätsobergrenzen (immediate priority ceiling)
- Anforderungen an Lösungen:
  - leicht zu implementieren
  - Anwendungsunabhängige Implementierung
  - Eventuell Ausschluss von Verklemmungen

Prioritätsvererbung (priority inheritance)

- Sobald ein Prozess höherer Priorität ein Betriebsmittel anfordert, das ein Prozess mit niedrigerer Priorität besitzt, erbt der Prozess mit niedrigerer Priorität die höhere Priorität. Nachdem das Betriebsmittel freigegeben wurde, fällt die Priorität wieder auf die ursprüngliche Priorität zurück.

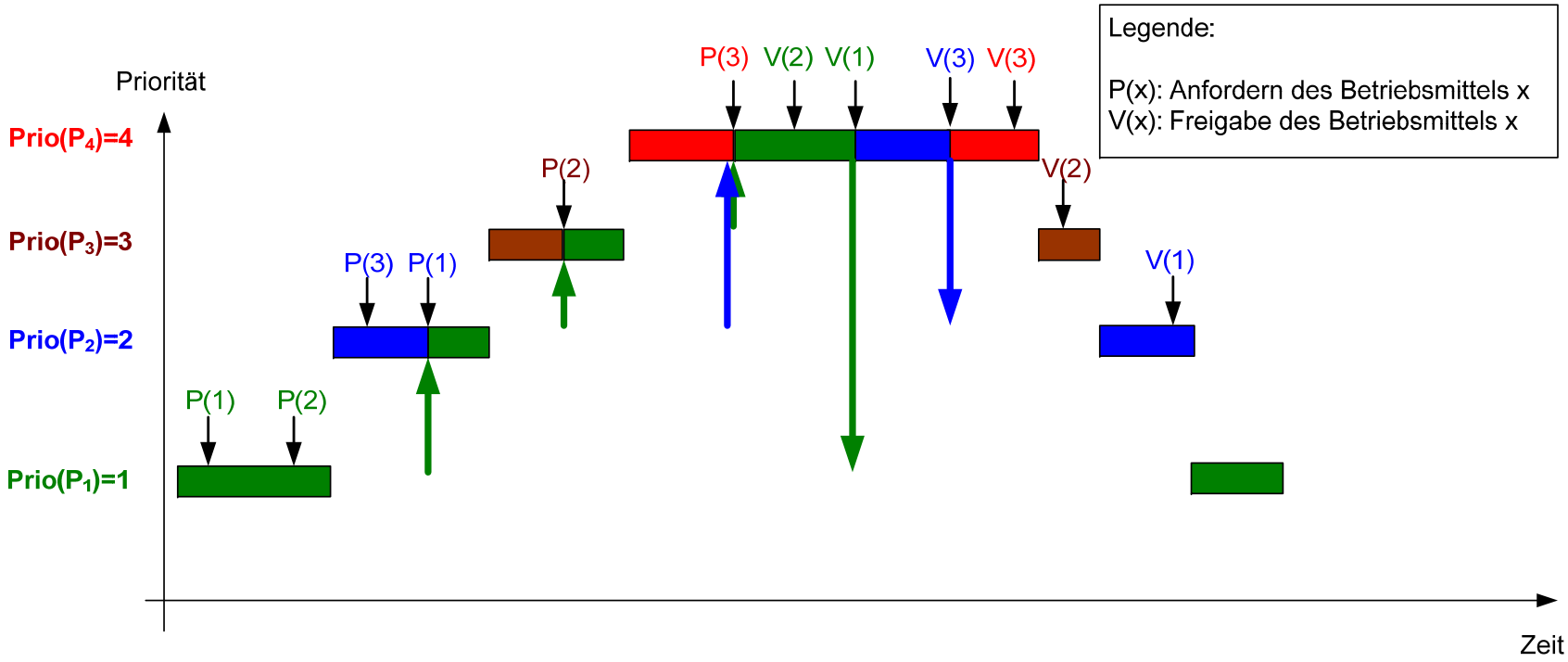
  $\rightarrow$ Unbegrenzte Prioritätsinversion wird verhindert.

  $\rightarrow$ Die Dauer der Blockade wird durch die Dauer des kritischen Abschnittes beschränkt.

  $\rightarrow$ Blockierungen werden hintereinander gereiht (Blockierungsketten).

  $\rightarrow$ Verklemmungen durch Programmierfehler werden nicht verhindert.

Beispiel: Prioritätsvererbung



Prioritätsobergrenzen (priority ceiling)

- Jedem Betriebsmittel (z.B. Semaphor) s wird eine Prioritätsgrenze ceil(s) zugewiesen, diese entspricht der maximalen Priorität der Prozesse, die auf s zugreifen.
  - Ein Prozess p darf ein BM nur blockieren, wenn er von keinem anderen Prozess, der andere BM besitzt, verzögert werden kann.
  - Die aktuelle Prioritätsgrenze für Prozess p ist
    aktceil = max{ ceil(s) | s $\in$ locked} mit locked = Menge aller von anderen Prozessen blockierten BM
  - Prozess p darf Betriebsmittel s benutzen, wenn für seine aktuelle Priorität aktprio gilt: aktprio(p) > aktceil(p)
  - Andernfalls gibt es genau einen Prozess, der s besitzt. Die Priorität dieses Prozesses wird auf aktprio gesetzt.

  $\rightarrow$ Blockierung nur für die Dauer eines kritischen Abschnitts

  $\rightarrow$ Verhindert Verklemmungen

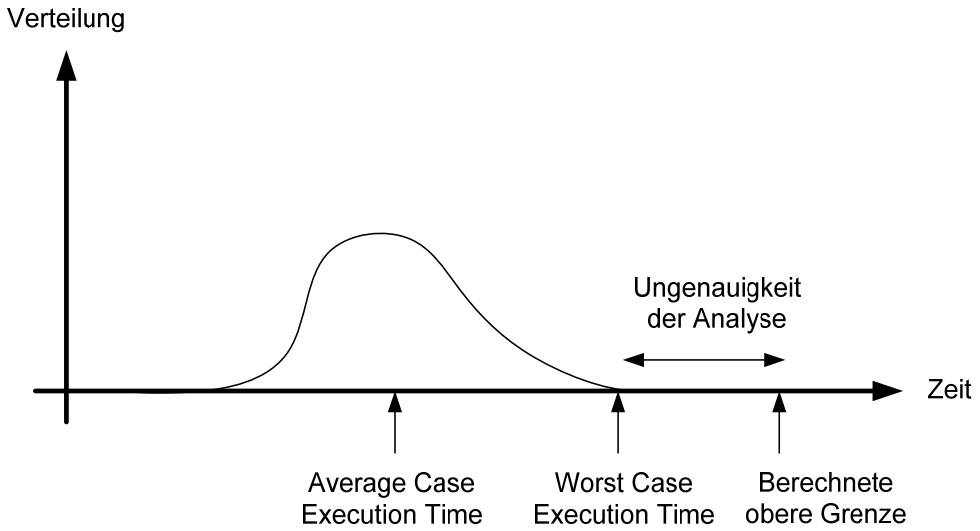
  $\rightarrow$ schwieriger zu realisieren, zusätzlicher Prozesszustand
- Vereinfachtes Protokoll: **Immediate priority ceiling**: Prozesse, die ein Betriebsmittel s belegen, bekommen sofort die Priorität ceil(s) zugewiesen.

Scheduling

Exkurs: WCET (Worst Case Execution Time) - Analyse

WCET Analyse

- Ziel der Worst Case Execution Time Analyse ist die Abschätzung der maximalen Ausführungszeit einer Funktion



- Die Laufzeit ist abhängig von den Eingabedaten, dem Prozessorzustand, der Hardware,…

Probleme bei der WCET Analyse

- Bei der Abschätzung der maximalen Ausführungszeiten stößt man auf einige Probleme:
  - Es müssen unter anderem die Auswirkungen der Hardwarearchitektur, des Compilers und des Betriebssystems untersucht werden. Dadurch erschwert sich eine Vorhersage.
  - Zudem dienen viele Eigenschaften der Beschleunigung des allgemeinen Verhaltens, jedoch nicht des Verhaltens im schlechtesten Fall, z.B.:
    - Caches, Pipelines, Virtual Memory
    - Interruptbehandlung, Präemption
    - Compileroptimierungen
    - Rekursion

	Zugriffszeit	Größe
Register	0.25 ns	500 bytes
Cache	1 ns	64 KB
Hauptspeicher	100 ns	512 MB
Festplatte	5 ms	100 GB

Zugriffszeiten für verschiedene Speicherarten

  - Noch schwieriger wird die Abschätzung falls der betrachtete Prozess von der Umgebung abhängig ist.

Unterscheidungen bei der WCET-Analyse

- Die Analyse muss auf unterschiedlichen Ebenen erfolgen:
  - Was macht das Programm?
  - Was passiert im Prozessor?

- Bei der Analyse werden zwei Methoden unterschieden:
  - **statische** WCET Analyse: Untersuchung des Programmcodes
  - **dynamische** Analyse: Bestimmung der Ausführungszeit anhand von verschiedenen repräsentativen Durchläufen

Statische Analyse

- Aufgaben:
  - Bestimmung von Ausführungspfaden in der Hochsprache
  - Transformation der Pfade in Maschinencode
  - Bestimmung der Laufzeit einzelner Maschinencodesequenzen
- Probleme:
  - Ausführungspfade lassen sich oft schlecht vollautomatisch ableiten (zu pessimistisch, zu komplex)
  - Ausführungspfade häufig abhängig von Eingabedaten
- Lösungsansatz: Annotierung der Pfade mit Beschränkungen (wie z.B. maximale Schleifendurchläufe)

Dynamische Analyse

- Statische Analysen können zumeist die folgenden Wechselwirkungen nicht berücksichtigen:
  - Wechselwirkungen mit anderen Programmen (siehe z.B. wechselseitiger Ausschluss)
  - Wechselwirkungen mit dem Betriebssystem (siehe z.B. Caches)
  - Wechselwirkungen mit der Umgebung (siehe z.B. Interrupts)
  - Wechselwirkungen mit anderen Rechnern (siehe z.B. Synchronisation)
- Durch dynamische Analysen können diese Wechselwirkungen abgeschätzt werden.
- Problem: Wie können die Testläufe sinnvoll ausgewählt werden.

Dimensionierung der Rechenleistungen

- Aufstellen der Worst-Case Analyse:
  - Rechenaufwand für bekannte periodische Anforderungen
  - Rechenaufwand für erwartete sporadische Anforderungen
  - Zuschlag von 100% oder mehr zum Abfangen von Lastspitzen
- Unterschied zu konventionellen Systemen:
  - keine maximale Auslastung des Prozessors
  - keine Durchsatzoptimierung
  - Abläufe sollen determiniert abschätzbar sein

Scheduling

Zusammenfassung

Zusammenfassung

- Kenntniss der Schedulingkriterien (Einhalten von Fristen, Fairness,…) , sowie der verschiedenen Prozessparameter (Startzeit, Laufzeit, Deadline, Priorität).
- Klassische Verfahren (EDF, LST, RM) und Anforderungen an die Optimalität dieser Verfahren
- Problem der Prioritätsinversion, sowie Lösungsverfahren
- Problematik der WCET-Analyse

Kapitel 8 (vorgezogen)

Echtzeitfähige Kommunikation

Inhalt

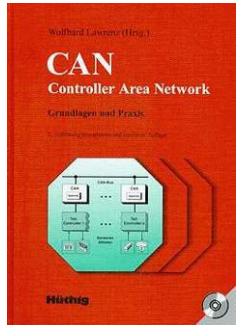
- Grundlagen
- Medienzugriffsverfahren und Vertreter
  - CSMA-CD: Ethernet
  - CSMA-CA: CAN-Bus
  - Tokenbasierte Protokolle: Token Ring, FDDI
  - Zeitgesteuerte Protokolle: TTP
  - Real-Time Ethernet

Literatur

- Spezifikationen:
  - TTTech Computertechnik AG, Time Triggered Protocol TTP/C High-Level Specification Document, 2003 (http://www.vmars.tuwien.ac.at/projects/ttp/)
  - http://www.can-cia.org/
  - http://standards.ieee.org/getieee802/portfolio.html

Andrew S. Tanenbaum, Computernetzwerke, 2005

Wolfhard Lawrenz: CAN Controller Area Network. Grundlagen und Praxis, 2000

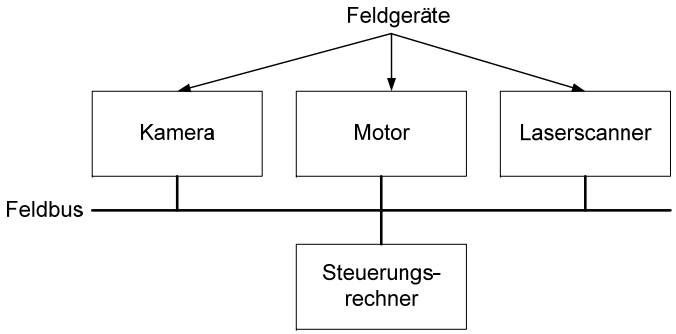


Anforderungen

- Echtzeitsysteme unterscheiden sich in ihren Anforderungen an die Kommunikation von Standardsystemen.
- Anforderungen speziell von Echtzeitsystemen:
  - vorhersagbare maximale Übertragungszeiten
  - kleiner Nachrichtenjitter
  - garantierte Bandbreiten
  - effiziente Protokolle: kurze Latenzzeiten
  - teilweise Fehlertoleranz
- Kriterien bei der Auswahl:
  - maximale Übertragungsrate
  - maximale Netzwerkgröße (Knotenanzahl, Länge)
  - Materialeigenschaften (z.B. für Installation)
  - Störungsempfindlichkeit (auch unter extremen Bedingungen)
  - Kosten, Marktproduktpalette

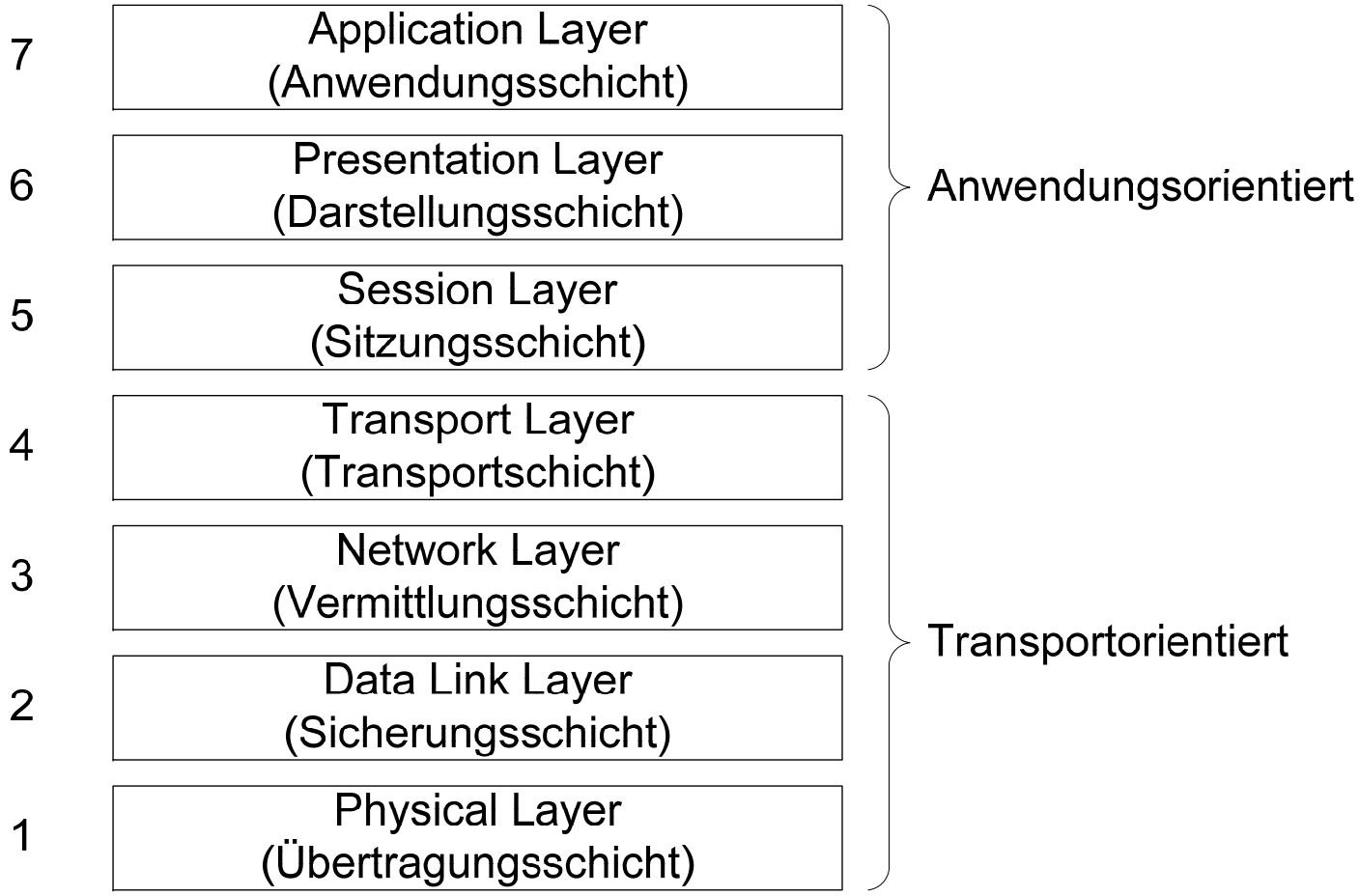
Definition Feldbus

- Die Kommunikation in Echtzeitsystemen erfolgt häufig über **Feldbusse**:



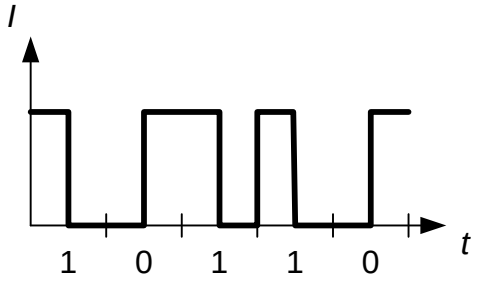
- Feldgeräte sind dabei Sensoren/Aktoren, sowie Geräte zur Vorverarbeitung der Daten.
- Der Feldbus verbindet die Feldgeräte mit dem Steuerungsgerät.
- Beobachtung: echtzeitkritische Nachrichten sind in der Regel kürzer als unkritische Nachrichten.
- Es existiert eine Vielzahl von Feldbus-Entwicklungen: MAP (USA - General Motors), FIP (Frankreich), PROFIBUS (Deutschland), CAN (Deutschland – Bosch), …

Schichtenmodell: ISO/OSI-Modell

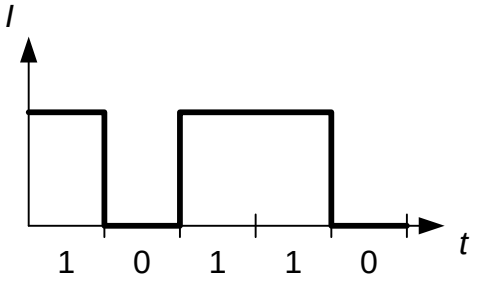


Beschreibung der einzelnen Schichten: Übertragungsschicht

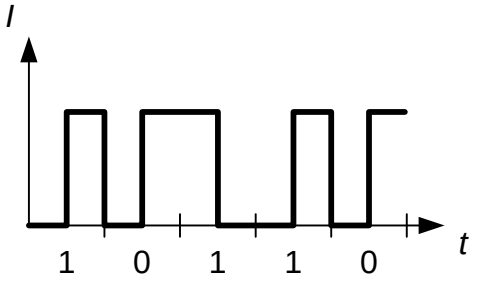
- Aufgaben:
  - Bitübertragung auf physikalischen Medium
  - Festlegung der Medien
    - elektrische, optische Signale, Funk
    - Normung von Steckern
  - Festlegung der Übertragungsverfahren/Codierung
    - Interpretation der Pegel
    - Festlegung der Datenrate



Manchester-Code



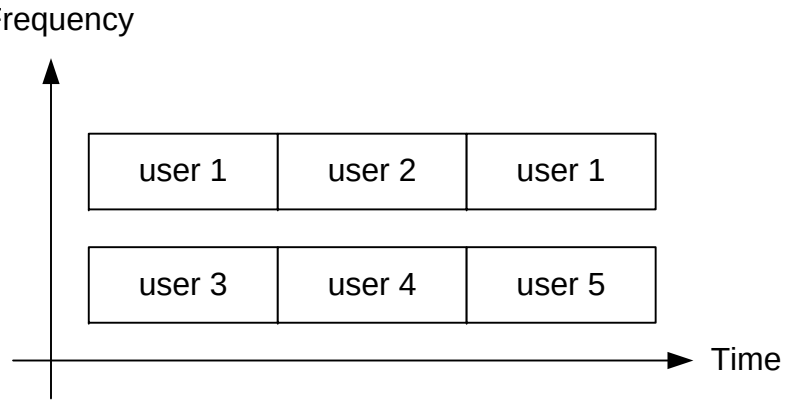
Non-return-to-zero Code



Differentieller Manchester-Code

Beschreibung der einzelnen Schichten: Sicherungsschicht

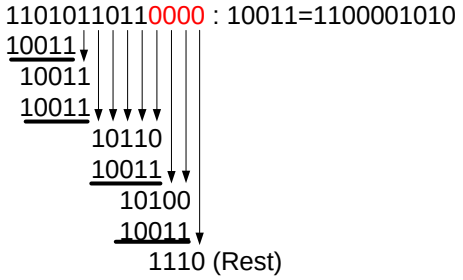
- Aufgaben:
  - Fehlererkennung
    - Prüfsummen
    - Paritätsbits
  - Aufteilung der Nachricht in Datenpakete
  - Regelung des Medienzugriffs
  - Flusskontrolle



TDMA+FDMA

								LRC
	1	0	1	1	0	1	0	1	1
	0	1	1	0	0	1	0	0	1
	0	0	0	1	1	0	1	1	0
	1	1	1	0	0	1	0	0	0
VRC	0	0	1	0	1	1	1	0	

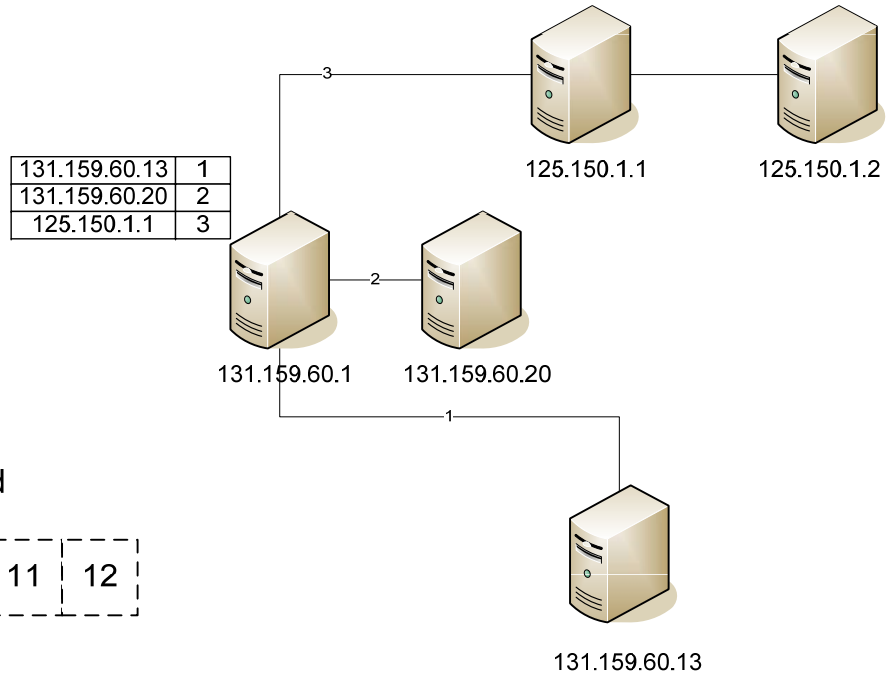
Paritätsbits

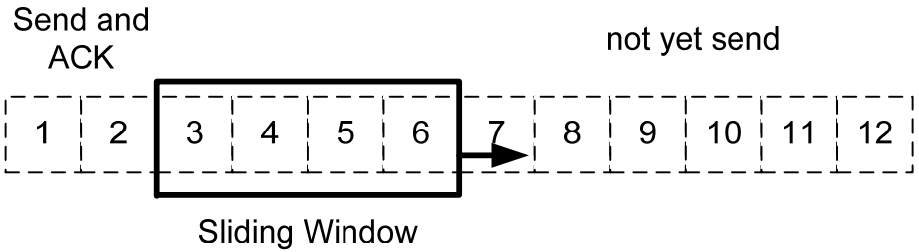


CRC-Verfahren

Beschreibung der einzelnen Schichten: Vermittlungsschicht

- Aufgaben:
  - Aufbau von Verbindungen
  - Weiterleitung von Datenpaketen
    - Routingtabellen
    - Flusskontrolle
    - Netzwerkadressen





Sliding Window Protokoll

Weitere Schichten

- Transportschicht:
  - Transport zwischen Sender und Empfänger (End-zu-End-Kontrolle)
  - Segmentierung von Datenpaketen
  - Staukontrolle (congestion control)
- Sitzungsschicht:
  - Auf- und Abbau von Verbindungen auf Anwendungsebene
  - Einrichten von Check points zum Schutz gegen Verbindungsverlust
  - Dienste zur Organisation und Synchronisation des Datenaustauschs
  - Spezifikation von Mechanismen zum Erreichen von Sicherheit (z.B. Passwörter)
- Darstellungsschicht:
  - Konvertierung der systemabhängigen Daten in unabhängige Form
  - Datenkompression
  - Verschlüsselung
- Anwendungsschicht:
  - Bereitstellung anwendungsspezifischer Übertragungs- und Kommunikationsdienste
  - Beispiele:
    - Datenübertragung
    - E-Mail
    - Virtual Terminal
    - Remote Login
    - Video-On-Demand
    - Voice-over-IP

Schichten in Echtzeitsystemen

- Die Nachrichtenübertragungszeit setzt sich aus folgenden Komponenten zusammen:
  - Umsetzung der Protokolle der einzelnen Schichten durch den Sender
  - Wartezeit auf Medienzugang
  - Übertragungszeit auf Medium
  - Entpacken der Nachricht in den einzelnen Schichten durch den Empfänger

⇒ Jede zu durchlaufende Schicht verlängert die Übertragungszeit und vergrößert die zu sendenden Daten.

⇒ in Echtzeitsystemen wir die Anzahl der Schichten zumeist reduziert auf:

  - Anwendungsschicht
  - Sicherungsschicht
  - Physikalische Schicht

Echtzeitfähige Kommunikation

Medienzugriffsverfahren

Problemstellung

- Zugriffsverfahren regeln die Vergabe des Kommunikationsmediums an die einzelnen Einheiten.
- Das Kommunikationsmedium kann in den meisten Fällen nur exklusiv genutzt werden, Kollisionen müssen zumindest erkannt werden um Verfälschungen zu verhindern.
- Zugriffsverfahren können dabei in unterschiedliche Klassen aufgeteilt werden:
  - Erkennen von Kollisionen, Beispiel: CSMA/CD
  - Vermeiden von Kollisionen, Beispiel: CSMA/CA
  - Ausschluss von Kollisionen, Beispiel: token-basiert, TDMA

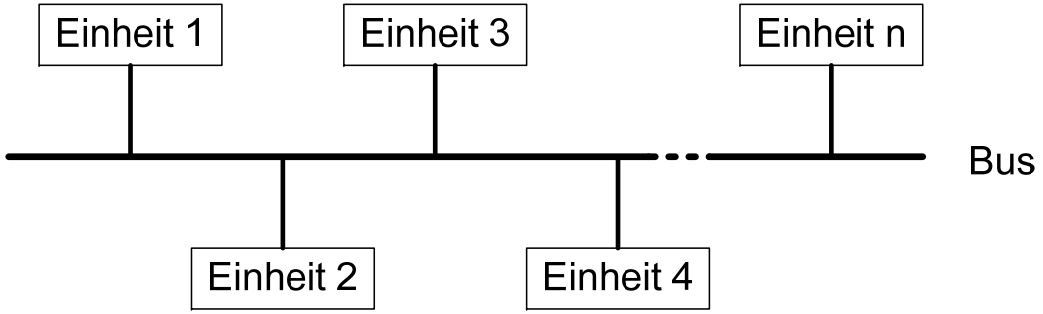
Echtzeitfähige Kommunikation

Carrier Sense Multiple Access/Collision Detection (CSMA/CD)

Vertreter: Ethernet (nicht echtzeitfähig!)

CSMA/CD

- CSMA/CD: Carrier Sense Multiple Access - Collision Detection
  - alle am Bus angeschlossenen Einheiten können die aktuell versendeten Daten lesen (**Carrier Sense**).
  - mehrere Einheiten dürfen Daten auf den Bus schreiben (**Multiple Access**).



  - Während der Übertragung überprüft der sendende Knoten gleichzeitig das Resultat auf dem Bus, ergibt sich eine Abweichung, so wird eine Kollision angenommen (**Collision Detection**)

CSMA/CD: Ablauf

- Beschrieben wird im Folgenden das 1-persistente CSMA/CD- Verfahren (Spezifikation in der Norm IEEE 802.3)
- Ablauf zum Senden eines Paketes:
  1. Test, ob Leitung frei ist (**carrier sense**)
  2. Falls Leitung für die Zeitdauer eines IFS (**inter frame spacing**) frei ist, wird die Übertragung gestartet, ansonsten Fortfahren mit Schritt 5.
  3. Übertragung der Daten inklusive Überwachung der Leitung. Im Fall einer Kollision: senden eines **JAM**-Signals, fortfahren mit Schritt 5.
  4. Übertragung erfolgreich beendet: Benachrichtige höhere Schicht, Beendigung
  5. Warten bis Leitung frei ist
  6. Sobald Leitung frei: weitere zufälliges Warten (z.B. **Backoff-Verfahren**) und Neustarten mit Schritt 1, falls maximale Sendeversuchsanzahl noch nicht erreicht.
  7. Maximale Anzahl an Sendeversuchen erreicht: Fehlermeldung an höhere Schicht.