

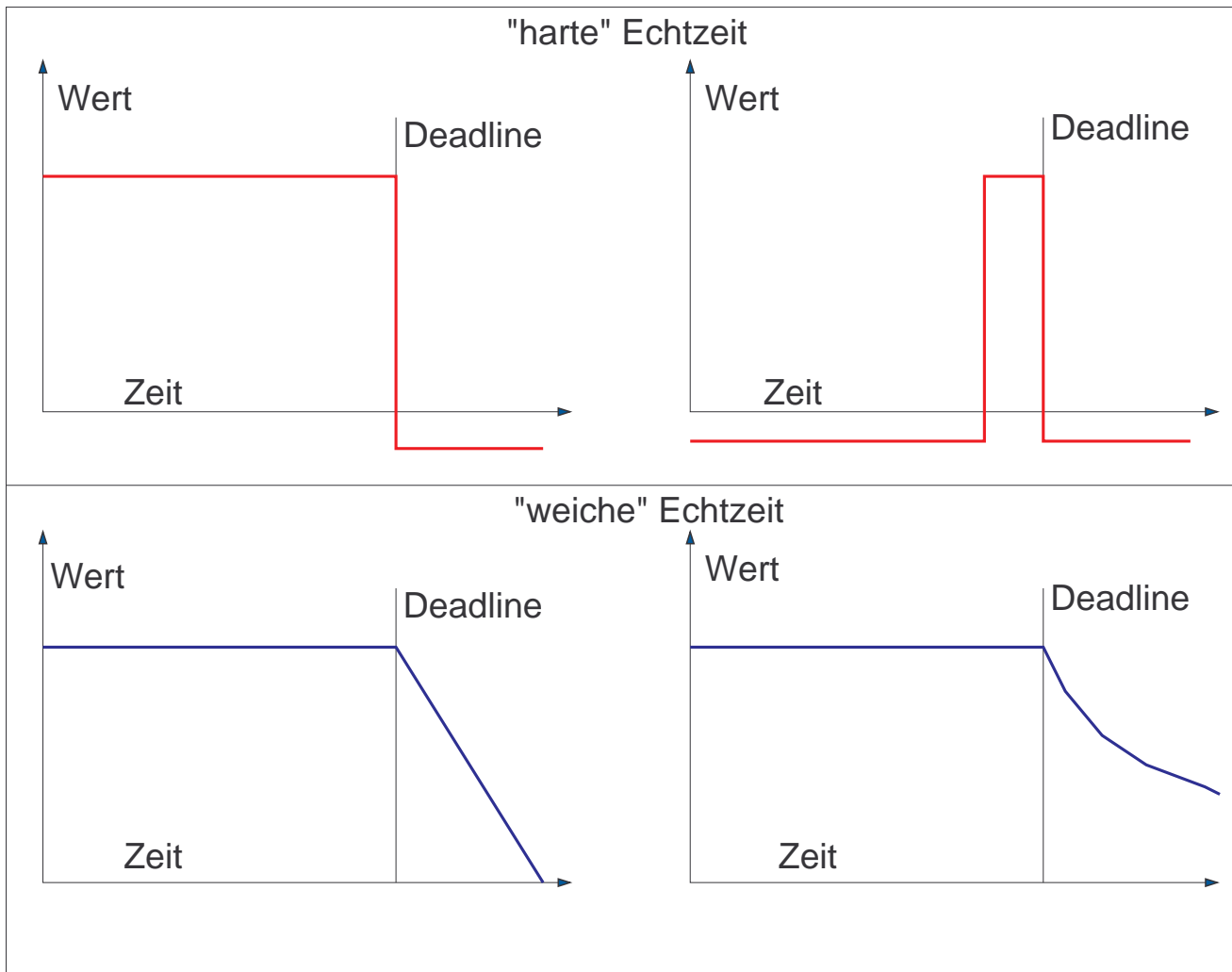
EMES: Eigenschaften mobiler und eingebetteter Systeme

# Echtzeitkommunikation

Dipl. Inf. Jan Richling  
Wintersemester 2004/2005



# Wiederholung - Resultat / Wert-Funktion



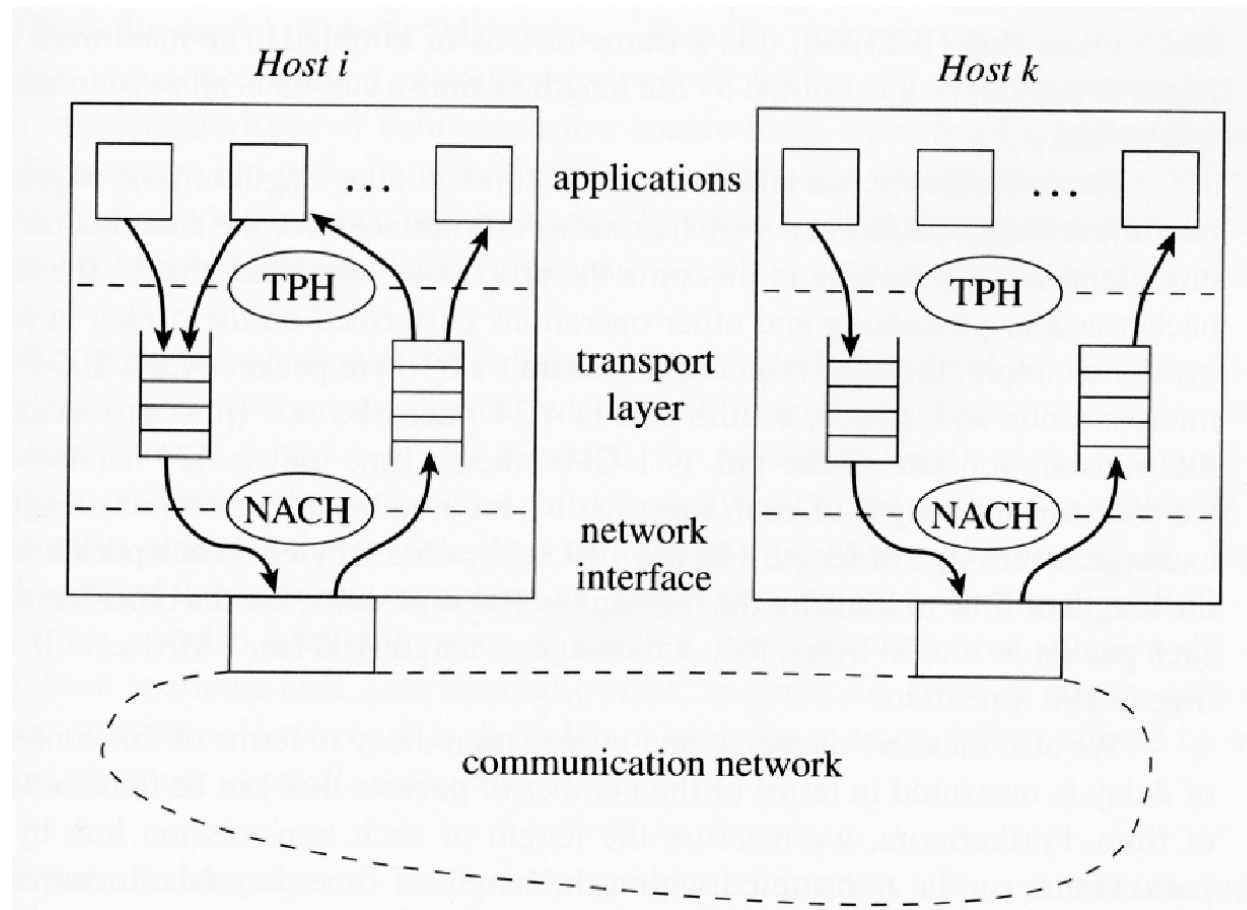
Echtzeitsysteme sind Computersysteme, bei denen der Nutzen eines Resultates nicht nur vom Resultat abhängt, sondern auch vom Zeitpunkt der Auslieferung des Resultats.



# Kommunikation

- Echtzeit erfordert, daß Ergebnisse zu einem bestimmten Zeitpunkt vorliegen
- In einem verteilten System bedeutet das:
  - Reaktion auf eine Eingabe an einem Knoten produziert Ausgabe an einem anderen Knoten
  - Zeitliche Anforderung ist ebenfalls verteilt:
    - \* Lokale Tasks auf Knoten
    - \* Transport auf Netzwerk

# Modell für Echtzeitkommunikation I



Vereinfachung: Alles über der Transportschicht wird als Applikation betrachtet

# Modell für Echtzeitkommunikation II

- Queues für zu sendende und für empfangene Nachrichten
- TPH - Transport Protocol Handler
  - Interface zu lokalen Applikationen
  - Bietet einen Nachrichtentransport-Service
- NACH - Network Access Control Handler
  - Interface zum Netzwerk
  - Bietet TPH Netzwerkzugriff
  - Bietet TPH Nachrichtenübermittlungsservice
- Paket
  - Nachrichten werden für die Übertragung in Pakete einer definierten Größe zerlegt
  - Übertragung eines Paketes ist nicht unterbrechbar
  - Pakete heißen in realen Netzen auch Frames, Segmente, Zellen

# Real-Time Traffic Modell

- Nachrichten sind oft Teil eines Nachrichtenstromes von Nachrichten des entsprechenden Types
  - Periodische Nachrichten
    - \* Übertragung einer periodischen Nachricht ist periodische Task
    - \* Charakterisiert durch  $p$  (Periode),  $e$  (Dauer der Übertragung) und  $D$  (Deadline auf Empfängerseite)
  - Aperiodische Nachrichten
    - \* Kein Wissen über Deadlines und Interarrival-Zeiten
    - \* Ziel: Baldmöglichste Auslieferung
  - Sporadische Nachrichten
    - \* Wissen über Interarrival-Zeiten / Deadlines
    - \* Zusätzlich zu periodischen Tasks:  $\bar{p}$  (durchschnittliche Interarrival-Zeit),  $l$  (Länge des Zeitintervalls für Durchschnitt), und  $p$  als minimale Interarrival-Zeit

# Performance-Maße I

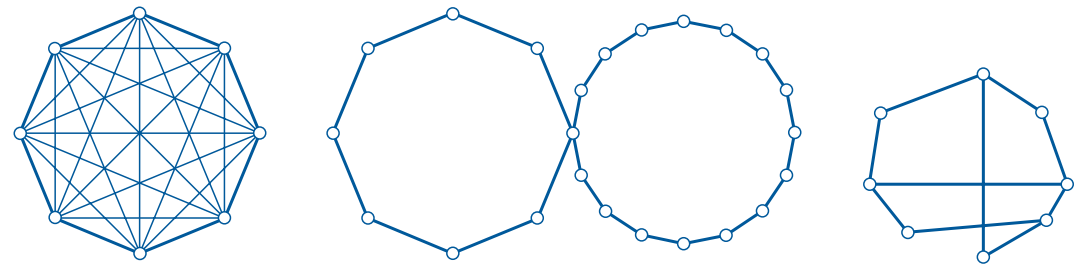
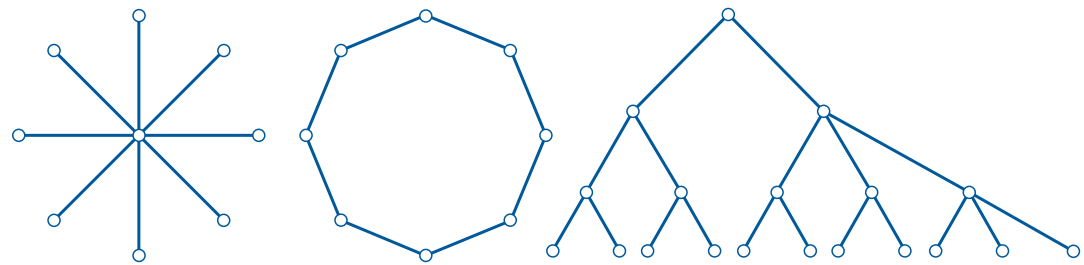
- Raten für Verlust, verpaßte Deadlines und beschädigte Nachrichten
  - Anteil am Gesamtnachrichtenaufkommen
  - Anforderung: Einhaltung bestimmter Mindestwerte
- Durchsatz
  - Anzahl von Nachrichten, die pro Zeiteinheit transportiert werden können
  - Mindestanforderungen an den Durchsatz sind Basis von Scheduling- und Flußsteuerungsalgorithmen

- Delay oder End-zu-End-Zeiten
  - Verzögerung einer Nachricht von sendender Task zu empfangender Task
  - Bei Steuerungssystemen oft essentiell
  - Bei Multimedia-Systemen oft irrelevant
- Delay-Jitter
  - Schwankungen der End-zu-End-Übertragungsdauer
  - Bei Multimedia-Anwendungen möglichst niedrig
  - Abhilfe gegen Jitter: große Puffer, erhöhen aber Delay

# Physikalische Netztopologie

- Echtzeitkommunikationssysteme können in beliebigen Topologien aufgebaut sein:

- Stern
- Ring
- Bus
- eng vermaschtes Netz
- Baum
- Mischformen



- Strukturen mit nicht direkter Verbindung zwischen Knoten erfordern Routing

Nachrichten liegen als Pakete oder Paketstrom vor:

- Packet Switching
  - Pakete einer Nachricht werden individuell zum Empfänger transportiert
  - Einzelne Pakete nehmen unter Umständen andere Wege
- Circuit Switching
  - Verbindung zwischen Sender und Empfänger wird geschaltet
  - Ganzer Nachrichtenstrom nimmt diesen Weg
- Wormhole Routing
  - Pakete werden zerlegt und suchen sich stückweise den Weg durch das Netz
  - Nur Anfangsteil hat Zieladresse
  - Vorteil: Erfordert weniger Pufferplatz

# Nutzung von Verbindungen

- Exklusiv
  - Punkt-zu-Punkt-Netzwerke
  - Erfordern Routing für mehrere Teilnehmer
  - Multicasts oder Broadcasts nur mit hohem Aufwand
  - U.U. hohe Kosten für dedizierte Verbindungen
- Geteilt
  - Broadcast-Netzwerke
  - Mehrere (oder alle) Teilnehmer teilen sich die Netzressource
  - Nachrichten gehen prinzipiell an alle
  - Adressierung auf “logischer Ebene”
  - Multicasts einfach möglich
  - Kostenersparnis in vielen Fällen

# Zugriffsverfahren

Problem: Zugriff auf Broadcastmedium erfordert Protokoll

- Nicht-Echtzeit: z.B. CSMA-CD (Carrier Sense Multiple Access Collision Detection):
  - Station mit Sendewunsch wartet, bis Medium frei, sendet dann
  - Beim Senden wird das Medium beobachtet, wenn eine Kollision festgestellt wird, wird abgebrochen
  - Neuer Versuch nach zufällig festgelegter Wartezeit
- In dieser Form nicht für Echtzeitkommunikation geeignet:
  - Zufällige Komponente verletzt Vorhersagbarkeit
  - Keine Priorisierung möglich
  - Keine garantierbare obere Schranke für Delay

# Zugriffsverfahren — Physikalische Probleme

- Physikalische Nachrichtenübertragung ist nicht beliebig schnell
- Damit: “Gleichzeitigkeit” hängt von der Auflösung der benutzten Uhren und der physikalischen Entfernung zwischen den Knoten ab
- Problem kann vernachlässigt werden, wenn Laufzeit klein gegen die Dauer der Übertragung eines Bits (“Bitzeit”) ist
- Anderenfalls: Propagationdelay - Zeit, die ein Signal benötigt, um alle Teilnehmer zu erreichen
- Beispiel:
  - 100 MBit/s Datenrate, 200 m Kabellänge
  - Bitzeit: 10 ns
  - Propagationdelay: 1000 ns (Ausbreitungsgeschwindigkeit ist  $2/3 c$ )
  - Kanal kann damit 100 Bit “speichern”
- Kollisionen werden erst nach Ablauf des Propagationdelay erkannt

# CSMA/CD-LON

## Carrier Sense Multiple Access Collision Detection im LON-System

- Idee analog zum klassischen Ethernet
- Vor jedem Senden wird zufällig lange gewartet
- Parameter des Zufallsgenerators sind lastabhängig
- Ziel: randomisierter Zugriff soll Kollisionshäufigkeit senken

# CSMA/CA — CAN

Carrier Sense Multiple Access Collision Avoidance im CAN-Bus

CAN: Controller Area Network

- Idee: Bitweise Arbitrierung und damit Vermeidung von Kollisionen
- Funktion:
  - 0 - elektrisch dominant
  - 1 - elektrisch rezessiv
  - Paralleler Start des Sendens der ID erzeugt binäre Priorisierung
  - Voraussetzung: Propagationdelay klein gegen Bitzeit
- Details: Vorlesung über Feldbusse

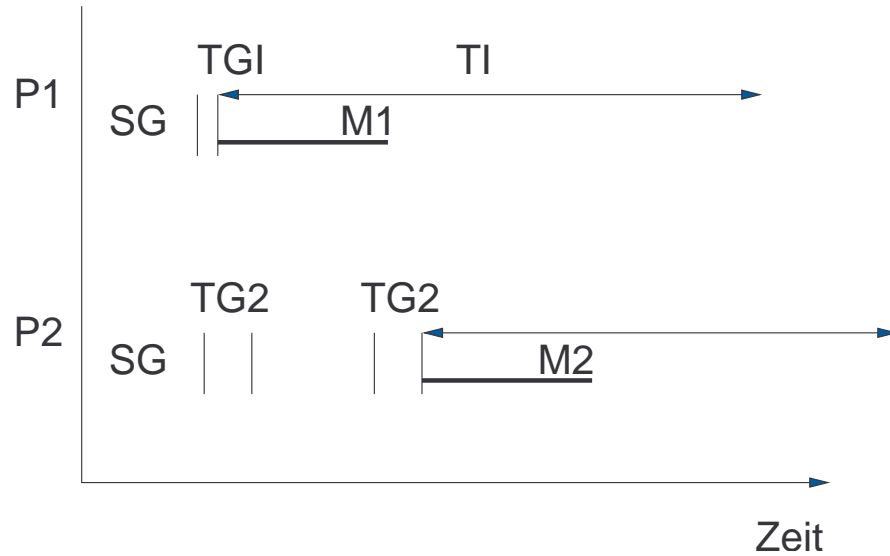
# Token Bus / Token Ring

- Spezielle Steuernachricht beinhaltet das Recht zum Senden: Token
- Token wird in definierter Reihenfolge weitergegeben
- Problem: Tokenverlust erzeugt Stillstand bis zum Erzeugen eines neuen Tokens
- Zeitliche Vorhersagbarkeit durch Festlegung der Zeit, die der Token für eine Runde maximal benötigen darf
- Nicht-Echtzeit-Nachrichten können gesendet werden, wenn der Token “zu früh” ankommt

# Minislotting — ARINC 620

- Idee: Prioritäten werden auf Minislots abgebildet, die eine Nachricht vor dem Senden zu warten hat
- Beispiel: ARINC 620, benutzt in Boeing 777
- Funktionsweise:
  - Task mit Sendewunsch wartet SG (Synchronization Gap) Stille ab
  - TG (prioritätsabhängige Wartezeit) Stille abwarten
  - Nachricht M senden
  - Timer T1 verhindert weitere Sendungen für einen gewissen Zeitraum
    - verhindert Blockade des Mediums durch einen Knoten
  - Da TG alle verschieden, gibt es keine Kollisionen

# Minislottting — ARINC 620



# Central Master — FIP

- Zentraler Master steuert Zugriffe auf Bus
- Ausfall des Master: Anderer Knoten übernimmt Rolle
- Beispiel: FIP-Protokoll
- Funktion:
  - Beim Systemstart wird eine statische Liste zu sendender Nachrichten generiert — enthält Namen und Perioden
  - Im Betrieb sendet der Master periodisch Namen von Nachrichten (Polling)
  - Knoten, der Nachricht zu senden hat, sendet diese
  - Freie Zeit kann für sporadische Daten benutzt werden



# TDMA — TTP

## Time Division Multiple Access

## TTP: Time Triggered Protocol

- Idee: Zur Verfügung stehende Zeit wird statisch auf die Knoten aufgeteilt
- Jeder Knoten kann im eigenen Zeit-Slot senden (einmal pro Runde)
- Wenn keine Daten zu senden: Leeres Paket wird gesendet (damit andere Teilnehmer erkennen, daß der Knoten noch arbeitet)
- Problem: Genaue Uhrensynchronisation erforderlich
- Details: Vorlesung über Feldbusse

# Eigenschaften von Echtzeitkommunikationsprotokollen

- Externe vs. interne Kontrolle
- Flexibilität vs. Fehlererkennung
- Sporadische Daten vs. periodische Daten
- Zentrale Steuerung vs. Fehlertoleranz
- Zufällige Zugriffe vs. Replikat-Determiniertheit
- Explizite vs. implizite Flußsteuerung
- Babbling Idiot-Problem

# Externe vs. interne Kontrolle I

- Externe Kontrolle: Senden einer Nachricht wird von der Anwendung gesteuert und ist für das Kommunikationssystem ein externes Ereignis (Pushing)
  - Vorteil: Flexibilität
  - Vorteil: Einfache Schnittstelle zwischen Anwendung und Kommunikationssystem
  - Vorteil: Trennung zwischen Anwendungslogik und Steuerung des Kommunikationssystems
  - Nachteil: Fehlerhafte Anwendungen können Kommunikationssystem mit Nachrichten überfluten
  - Nachteil: Überlast durch gleichzeitiges Senden vieler Anwendungen

# Externe vs. interne Kontrolle II

- Interne Kontrolle: Kommunikationssystem legt den Zeitpunkt einer Sendung fest und fordert Daten von Anwendung (Polling)
  - Vorteil: Laststeuerung vollkommen unter Kontrolle des Kommunikationssystems
  - Vorteil: Fluten nicht möglich
  - Vorteil: “Komponierbarkeit”
  - Vorteil: Anwendungen und Kommunikationssystem können separat getestet werden
  - Nachteil: Kommunikationssteuerung ist in Anwendungslogik involviert (unflexibel)
  - Nachteil: Schnittstelle komplizierter, da zeitliche Aspekte berücksichtigt werden müssen

# Flexibilität vs. Fehlererkennung

Annahme: System ohne Replikation

- Je weniger a-priori über ein System bekannt ist, desto schwerer ist die Erkennung von Fehlern
- Wenn Vorwissen existiert (z.B. Knoten sendet ein alive in regelmässigen Abständen), ist Fehlererkennung vereinfacht
- Einführung von a-priori-Wissen geht immer auf Kosten der Flexibilität

# Sporadische Daten vs. periodische Daten

Einander widersprechende Zielstellungen

- Periodische Daten
  - Viel a-priori-Wissen (Periode, Nachrichtentypen, Sendezeitpunkte,...)
  - Ziel: Jitterminimierung
  - Schedule ist vorab berechenbar, hohe Auslastung erzielbar
- Sporadische Daten
  - Kein Wissen über Perioden und Sendezeitpunkte
  - Nur geringes Vorabwissen über minimal Interarrival Time
  - Ziel: Geringes Delay

# Zentrale Steuerung vs. Dezentrale Fehlertoleranz

- Zentrale Steuerung vereinfacht Zugriffsverfahren (z.B. Central Master oder auch Token-Verfahren)
- Zentrale Steuerung ist ein “single point of failure”: Ausfall bedeutet Notwendigkeit der Rekonfiguration
  - Verletzung zeitlicher Eigenschaften
  - Erkennung des Ausfalls kostet Zeit
  - Festlegung einer neuen Kontrollinstanz erfordert zusätzliche Kommunikation

# Zufällige Zugriffe vs. Replikat-Determiniertheit

- Replikat-Determiniertheit: Replikate verhalten sich bei gleicher Eingabe stets gleich
- Zufällige Zugriffe: Folge von zeitlich fein granularen Wettlaufbedingungen bei Arbitrierungen oder Kollisionserkennung
- Problem: Replikate müssen sich durch verschiedenes physikalisches Verhalten bei solchen Verfahren nicht gleich verhalten
- Folge: Beide Replikate kommen zu einem verschiedenen korrekten Ergebnis - Replikat-Determiniertheit ist nicht gegeben

# Explizite vs. implizite Flußsteuerung

Flußsteuerung: Steuerung der Übertragungsgeschwindigkeit zwischen Sender und Empfänger

- Explizite Flußsteuerung:
  - Empfänger bestätigt Pakete des Senders
  - Nächstes Paket wird erst nach Bestätigung gesendet
  - Nichtbestätigte Pakete werden wiederholt
  - Voraussetzung: Sender muß vom Empfänger erreichbar sein
  - Erhöhter Jitter
- Implizite Flußsteuerung:
  - Sender und Empfänger einigen sich a-priori auf Sendezeitpunkte
  - Empfänger erkennt Fehler
  - Einfache Realisierung von Redundanz
  - Nicht geeignet für Nachrichten ohne Vorabwissen
  - Voraussetzung: Synchronisation

# Das “Babbling-Idiot” -Problem

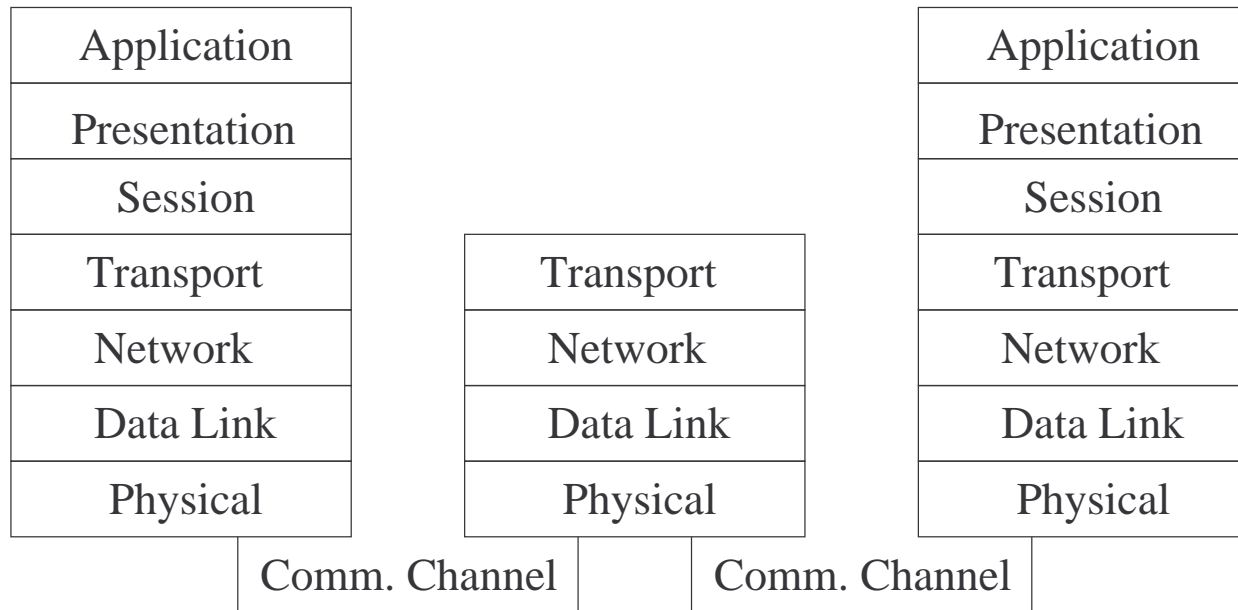
- In Broadcast-Netzen kann ein einzelner Knoten durch Verletzung des Kommunikationsprotokolls sämtlichen Verkehr stören
- Repliziertes Medium hilft nur bedingt: Wenn Problem in der Anwendung liegt, werden beide Medien gestört
- Besser: Mechanismus im Kommunikationssystem
- Möglichkeiten
  - Kommunikationssystem “weiß”, wann gesendet werden darf und läßt es nur zu diesen Zeiten zu (TTP)
  - Kommunikationssystem setzt Obergrenzen für Sendungslängen oder Mindestabstände zwischen Sendungen (ARINC-620)
  - Erkennung von fehlerhaften Knoten und “Sperrung” (CAN)

# Echtzeitkommunikation auf Basis von Standard-Netzen

COTS auch für Kommunikation?

- Viele Anwendungen sind weit verteilt
- Proprietäre oder dedizierte Verbindungen sind teuer und nicht immer realisierbar
- COTS-Netzhardware ist kostengünstig verfügbar
- Für weiche Echtzeit: Mit entsprechenden Maßnahmen benutzbar
- Für harte Echtzeit: Kommunikation basiert in den meisten Fällen auf dem OSI-Modell oder Vereinfachungen davon

# OSI-Modell und Echtzeit I



# OSI-Modell und Echtzeit II

- Sind typische Implementationen nach dem OSI-Modell (z.B. TCP/IP über Ethernet) für harte Echtzeit geeignet?
- Nein, denn:
  - Explizite Flußsteuerung mit Retransmit im Fehlerfall - Delay kann sehr hoch werden
  - Latency und Latency Jitter sind nicht berücksichtigt und nicht parametrisierbar
  - Modell ist auf Punkt-zu-Punkt-Verbindungen beschränkt
  - Ausschließlich externe Steuerung der Kommunikation

# Echtzeit im Internet I

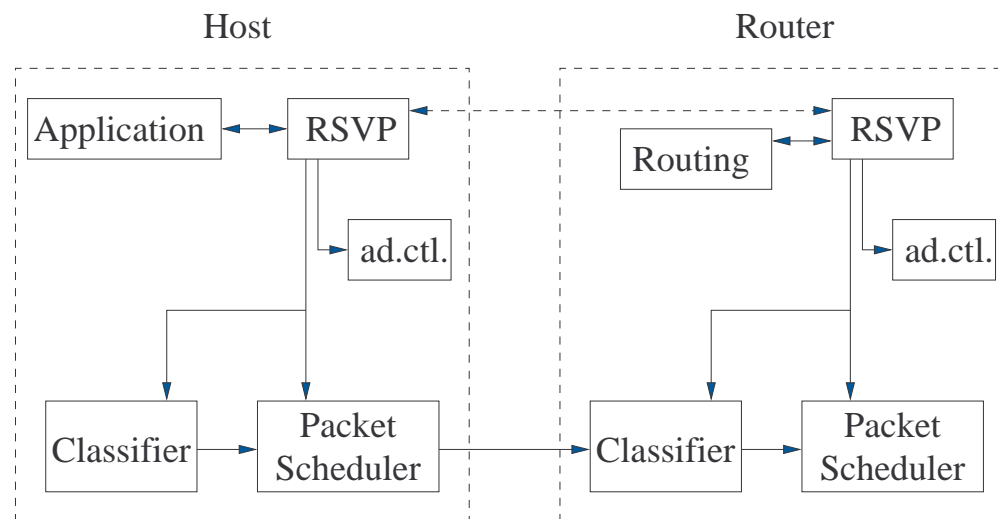
- Internet-Protokolle sind nicht für die Übertragung von Echtzeitdaten geeignet:
  - Vielschichtiges Protokoll
  - Wegwahl von Sender zum Empfänger nicht bekannt und nicht statisch
  - Netzkomponenten auf dem Weg unterliegen externer Kontrolle
  - Garantien können nicht gegeben werden (unzureichendes Wissen)

# Echtzeit im Internet II

- Lösungen für weiche Echtzeit:
  - Benutzung dedizierter Verbindungen im Netz
    - \* Teuer, nur selten realisierbar
    - \* Bietet volle Kontrolle
  - Ressource-Reservierung
    - \* Für eine Datenübertragung wird ein fester Weg gewählt
    - \* Beteiligte Router reservieren Kapazitäten
    - \* Problem: Aushandlung und Abrechnung
    - \* Protokolle: RSVP und ST-II
  - IPv6 mit QoS

# Resource Reservation Protocol

- Empfänger-initiierte Ressourcen-Reservierung
- Baut auf existierender Verbindung oder Multicast-Tree auf
- Verschiedene Reservierungen innerhalb von Multicast-Gruppen
- Aufbau eines Baumes von den Empfängern zum Sender
- Reservierung entsprechender Ressourcen in den beteiligten Routern
- Router können Reservierungen abweisen — Reject wird zum Empfänger zurückgeleitet und Reservierungsbaum abgebaut



# ST-II — Internet Stream Protocol

- Integriert Multicast Routing, Verbindungsaufbau und Datenübertragung
- Reservierung vom Sender initiiert
- Keine Berücksichtigung verschiedener QoS verschiedener Empfänger
- Bietet one-to-many Streams auf IP an
- Beim Aufbau des Streams wird Multicast-Tree zwischen den Routern aufgebaut
- Im Betrieb realisiert ST-II Multicasts (Router an Verzweigungspunkten machen Multicast aus Unicast)
- Unterstützung von priorisierten Streams (bei Reservierung bedeutsam)
- Stream-Spezifikation wird beim Aufbau ausgehandelt

