

# Echtzeitsysteme

## Verteilte Echtzeitsysteme

Lehrstuhl Informatik 4

12. Januar 2012

## Fragestellungen

- Warum sind Echtzeitsysteme häufig auch verteilte Systeme?
  - Wären zentralisierte Rechensysteme nicht einfacher handhabbar?
  - Welche Vorteile bietet eine verteilte Lösung?
- Welche Anforderungen stellen wir an verteilte Echtzeitsysteme?
  - Wie helfen uns verteilte Systeme, um die Rechenleistung eines Echtzeitsystems zu steigern und dessen Komplexität zu beherrschen?
- Wie sind verteilte Echtzeitsysteme grundlegend aufgebaut?
  - Welche Elemente verteilter Echtzeitsysteme werden unterschieden?
- Was zeichnet echtzeitfähige Kommunikationssysteme aus?

## Gliederung

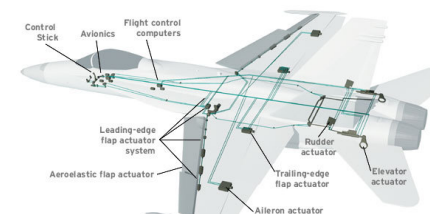
- 1 Überblick & Motivation
- 2 Anforderungen an verteilte Echtzeitsysteme
  - Zusammensetzbarkeit
  - Skalierbarkeit
  - Verlässlichkeit
- 3 Aufbau verteilter Echtzeitsysteme
  - Kommunikationssystem
  - Netzwerkschnittstelle
  - Topologie
- 4 Kommunikationssysteme
  - Anforderungen
  - Zeitliche Kapselung
  - Flusskontrolle
  - Netzzugangsprotokolle
- 5 Zusammenfassung

## Echtzeitsysteme sind inhärent verteilt!

Orientierung an der physikalischen Verteilung des zu kontrollierenden Objekts

Elemente wie Sensoren, Aktoren und Bedienpulte unterliegen häufig einer natürlichen Verteilung, die durch das physikalische Objekt vorgegeben ist.

**Beispiel:** Fly-By-Wire-Systeme moderner Flugzeuge



Quelle: IEEE Spectrum [6]

weitere Beispiele:

- Automobile, Schienenverkehr (Züge und Signalanlagen), Industrieanlagen (Kraftwerke, Fertigungsstraßen, ...), ...

- Bedienpult → Cockpit
- Triebwerke → Heck, Flügel
- Leitwerke → Heck, Flügel
- Fahrwerk → Rumpf
- Steuerrechner → Rumpf

## Echtzeitsysteme sind komplex!

Echtzeitrechnungssysteme übernehmen eine Vielzahl von Aufgaben:

- von einfach Fensterhebern (aber mit Einklemmschutz),
- über zeitkritische Motorsteuerungen (bei bis zu 10000 U/min),
- bis zu komplexen Fahrerassistenzsystemen

Die **Vielfalt** der abzuarbeitenden Aufgaben ist enorm, ihr **Rechenzeitbedarf** ist durch monolithische Systeme nicht zu erfüllen.

**Verteilung** der Aufgaben auf mehrere Knoten erhöht die Rechenleistung

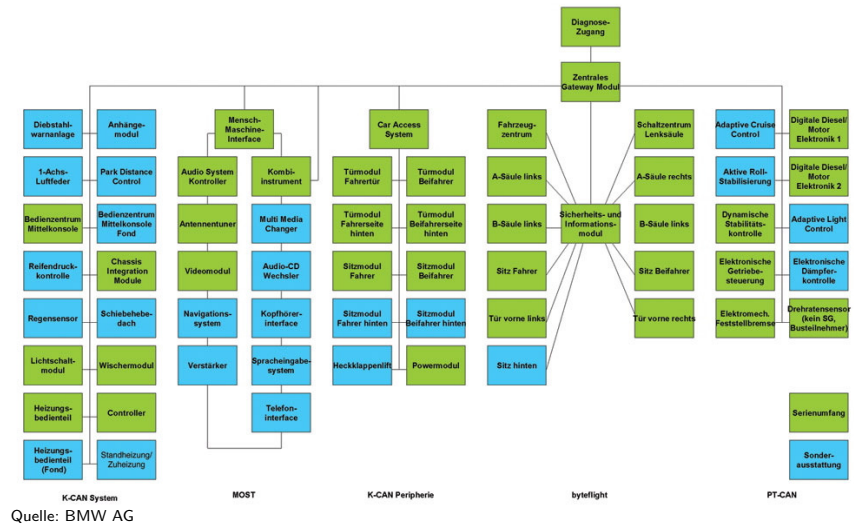
- die Motorsteuerung wird z.B. oft exklusiv von einem einzigen Steuergerät übernommen

**Aufteilung** in Subsysteme reduziert die Komplexität

- eigene Bereiche für Antriebssteuerung, Komfortfunktionen (z.B. Klimaanlage) oder Infotainment im verteilten System „Automobil“

## Verteiltes System auf Rädern

Vernetzung beim 7er BMW, Baureihe E65

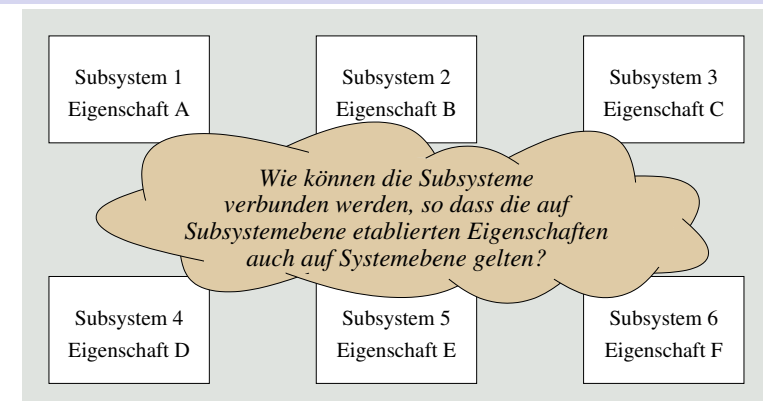


## Gliederung

- 1 Überblick & Motivation
- 2 Anforderungen an verteilte Echtzeitsysteme
  - Zusammensetzbarkeit
  - Skalierbarkeit
  - Verlässlichkeit
- 3 Aufbau verteilter Echtzeitsysteme
  - Kommunikationssystem
  - Netzwerkschnittstelle
  - Topologie
- 4 Kommunikationssysteme
  - Anforderungen
  - Zeitliche Kapselung
  - Flusskontrolle
  - Netzzugangsprotokolle
- 5 Zusammenfassung

## Kompositionsproblem

Zentrale Rolle von Echtzeitkommunikationssystemen bzw. der Netzwerkschnittstelle

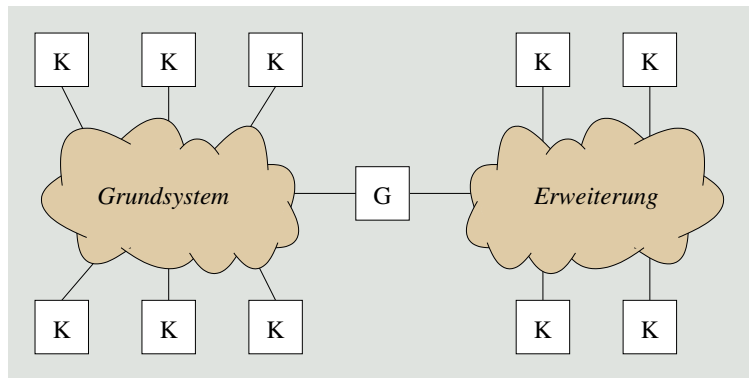


Architekturen sind zusammensetzbar (engl. *composable*) hinsichtlich einer spezifizierten Eigenschaft, wenn die Systemintegration diese vorher für ein Subsystem festgelegte Eigenschaft weiterhin aufrecht erhält

- **Rechtzeitigkeit** (engl. *timeliness*), Testbarkeit (engl. *testability*)

## Transparenter Ausbau einer Gerätegruppe

Hinzunahme von Knoten  $\leadsto$  Andocken neuer Gerätegruppen



- neue Anforderungen sind keine Ausnahme, sondern die Regel
- eine **skalierbare Architektur** ist offen für Änderungen...

## Erweiterbarkeit

Graduelle Leistungszunahme — bzw. Leistungsabnahme, bei Schrumpfung

Architekturen dürfen **keine zentralen Flaschenhälse** aufweisen, um skalierbar zu sein in Bezug auf Rechen- und Kommunikationsleistung

- eine Hinzunahme von Knoten richtet sich nach der noch freien Kommunikationskapazität der Gerätegruppe
  - lediglich die Rechenleistung des Systems wird erhöht
- ist die Kommunikationskapazität einer Gerätegruppe erschöpft, so eröffnet der neue Knoten eine neue Gerätegruppe
  - ein Knoten der alten Gruppe „mutiert“ zum Netzübergangsknoten
  - der „geopferte“ und der neue Knoten bilden eine neue Gruppe
  - der Netzübergang ist transparent für andere Knoten
- die Zuordnung von Funktion zu Knoten muss weiterhin einer globalen Verteilungsdisziplin gehorchen
  - **Lastausgleich** (statisch, dynamisch)

☞ nur eine **verteilte Architektur** ermöglicht unbegrenztes Wachstum

## Komplexität

Komponentenanzahl, Anzahl und Art der Komponenteninteraktionen

*The partitioning of a system into subsystems, the encapsulation of the subsystem, the preservation of the abstractions in case of faults, and most importantly, a strict control over the interaction patterns among subsystems, are thus the key mechanisms for controlling the complexity of a large system. [9, S. 37]*

Komplexität eines großen Systems kann reduziert werden, wenn...

- das **innere Verhalten** der Subsysteme verborgen/gekapselt ist,
- zur Abkapselung **stabile Schnittstellen** Verwendung finden und
- diese Schnittstellen der Subsysteme „einfach und verständlich“ sind

## Reaktionsfähiges System (engl. *responsive system*)

↪ Verteilung + Echtzeitperformanz + Fehlertoleranz [13]

**Fehlereingrenzung** durch eine **Sicherheitshülle** (engl. *containment*)

- fehlertolerante Systeme sind in Partitionen strukturiert
  - in „fehlereingrenzende Regionen“ (engl. *error-containment regions*)
- Fehler, die in einer Partition auftreten, bleiben isoliert...
  - sie werden lokal erkannt und korrigiert oder maskiert
  - sie können das restliche System nicht beschädigen
- Fehlererkennung betrifft den Wert- wie auch den Zeitbereich

**Replikation** (der Funktionen) von Knoten

- erfordert **Replikdeterminismus** (engl. *replica determinism* [15])
  - aktiv replizierte Knoten sehen denselben Zustand *zur selben Zeit*
  - mit Zugeständnis zur endl. Genauigkeit der *Uhrensynchronisation*
- Maßnahme zur Fehlermaskierung: *hot/warm/cold stand-by*
  - ohne Replikdeterminismus ist Abstimmung (engl. *voting*) sinnlos...
- Fehlereingrenzung ist weiterhin unabdingbar
  - ansonsten droht eine Beeinflussung durch ein fehlerhaftes Replikat

## Fehlertransparenz

Zuschnitt und Platzierung der fehlereingrenzenden Regionen und die Schnittstellen zwischen ihnen bilden wesentliche Aktivitäten des Entwurfs eines verteilten Echtzeitrechensystems

- bestimmen die Struktur und Eigenschaften des Systems als Ganzes

Herausforderung dabei ist, dass die Abstraktionen an den Schnittstellen auch im **Fehlerfall** ihre Gültigkeit haben sollten

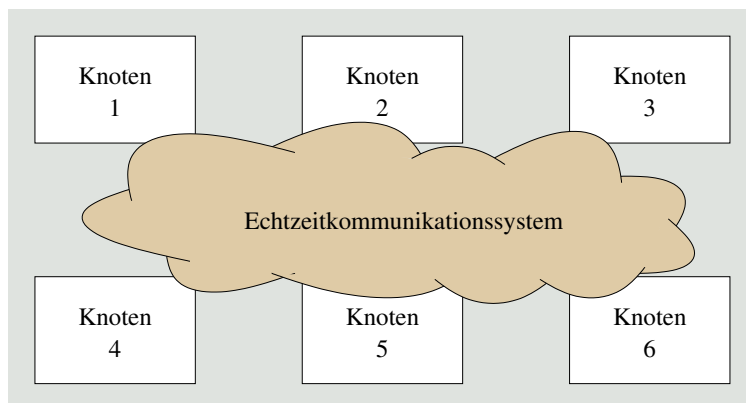
- das Ideal ist ein **verteiltes Rechensystem** mit einer 1-zu-1-Beziehung zwischen Funktion und Rechenknoten
  - (a) Ursache bzw. Knoten einer Fehlfunktion ist „leicht“ identifizierbar
  - (b) Auswirkungen eines fehlerhaften Knotens sind „gut“ vorhersehbar
- ein **zentralisiertes Rechensystem** erschwert die Fehlerdiagnose enorm, wie auch die Analyse von Fehlereffekten von Subsystemen

## Gliederung

- 1 Überblick & Motivation
- 2 Anforderungen an verteilte Echtzeitsysteme
  - Zusammensetzbarkeit
  - Skalierbarkeit
  - Verlässlichkeit
- 3 Aufbau verteilter Echtzeitsysteme
  - Kommunikationssystem
  - Netzwerkschnittstelle
  - Topologie
- 4 Kommunikationssysteme
  - Anforderungen
  - Zeitliche Kapselung
  - Flusskontrolle
  - Netzzugangsprotokolle
- 5 Zusammenfassung

## Verteiltes Echtzeitrechensystem

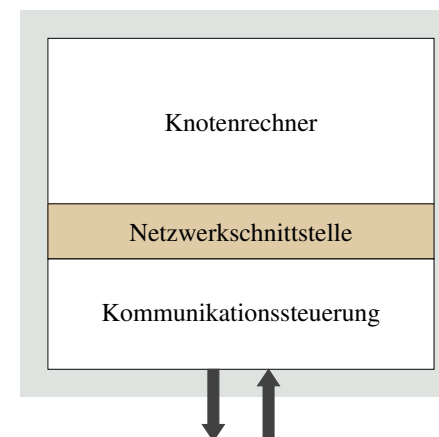
Rechenbetonte Gerätegruppe (engl. *computational cluster*)



- jeder Knoten erbringt eine Teilfunktion des Gesamtsystems
- ein **Kommunikationssystem** (KS) sorgt für die enge/lose Kopplung

## Grobstruktur eines Knotens

Partitionierung in zwei Subsysteme: Knotenrechner und Kommunikationssteuerung



### Echtzeitkommunikationssystem

- Menge der Subsysteme zur Kommunikationssteuerung der Knoten der Gerätegruppe
- zusammen mit dem jeweiligen phys. Verbindungsmedium

### Netzwerkschnittstelle

- **Transportschicht** des ISO OSI Referenzmodells [7]
- wichtigster Bestandteil des Echtzeit-KS

### Kommunikationssteuerung (engl. *communication controller*)

- Gerätetreiber und Netzsteuerung (Hardware und Software)

## Netzwerkschnittstelle

Semantik von Daten und Strategie der Steuerung

Abstraktion von den Details der Protokolllogik und der physikalischen Struktur des Kommunikationsnetzwerks. . .

- einerseits in Bezug auf die **Datensemantik**, die Nachrichteninhalte als Ereigniseintritt oder Zustandswert versteht
  - (a) da jedes Ereignis signifikant ist, sind alle Nachrichten entsprechend ihrer Ereigniszeitpunkte zwischenspeichern  $\mapsto$  sortierte Schlange
    - Nachrichtenverlust bedeutet ggf. Synchronisationsverlust
  - (b) da nur aktuelle Zustandswerte signifikant sind, ist immer nur die zuletzt empfangene Nachricht zu speichern  $\mapsto$  überschreiben
- andererseits in Bezug auf die **Steuerungsstrategie**, die zwischen zwei Kontrollbereichen differenziert
  - (a) **externe Kontrolle** im Knotenrechner, die vom Kommunikationssystem die Anzeige von Kontrollsignalen erfordert  $\leadsto$  **Ereignissteuerung**
  - (b) **autonome Kontrolle** im Kommunikationssystem, die Knotenrechner ununterbrochen weiter arbeiten lässt  $\leadsto$  **Taktsteuerung**

## Nachrichten besonderer Bedeutung

Ereigniseintritt vs. Zustandswert

**Ereignisnachricht** (engl. *event message*)

- kombiniert Ereignissemantik mit externer Kontrolle:
  - jede eingehende Nachricht wird beim Empfänger gepuffert
  - Entsorgung erfolgt durch Konsumierung (d.h., bei Verarbeitung)
- erfordert 1-zu-1-Synchronisation zwischen Sender und Empfänger
  - zur Vermeidung von Pufferüberlauf bzw. Empfängerblockaden
- korrespondiert zum „klassischen“ Botschaftenaustausch  $\mapsto$  IPC

**Zustandsnachricht** (engl. *state message*)

- kombiniert Zustandswertsemantik mit autonomer Kontrolle
  - entspricht der Semantik globaler Variablen, jedoch. . .
    - (a) das Kommunikationssystem garantiert unteilbares schreiben
    - (b) es gibt nur 1 Schreiber (engl. *multiple reader, single writer; MRSW*)
  - gestattet eine losere Kopplung zwischen Sender und Empfänger
- korrespondiert zu den Anforderungen von Steuerungsanwendungen

## Interagierende Gerätegruppen

Netzübergang (engl. *gateway*)

**Netzübergangsknoten** (engl. *gateway nodes*) schlagen Brücken zwischen verschiedenen Gerätegruppen (engl. *cluster*)

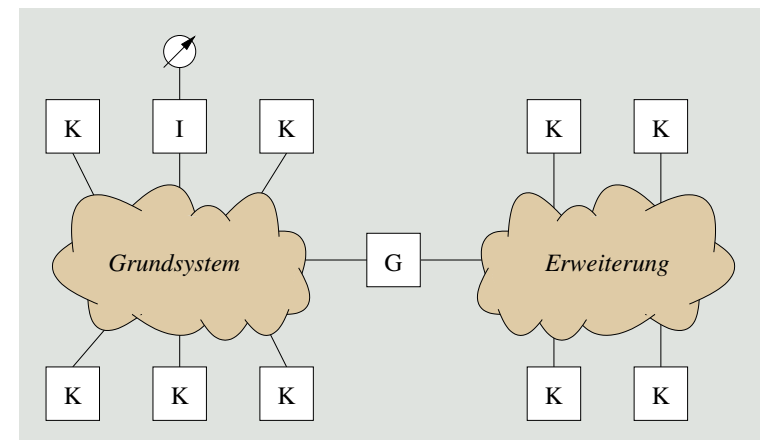
- Netzübergänge kommen mit zwei Ausprägungen von Schnittstellen:
  - ① eine Instrumenten- und eine Kommunikationsschnittstelle
    - auch als **Schnittstellenknoten** (engl. *interface node*) bezeichnet
  - ② zwei Kommunikationsschnittstellen (d.h., Netzwerkschnittstellen)
- sie bilden einen „Umschlagplatz für relevante Informationen“
  - in nicht allen Gerätegruppen ist jede Information signifikant
  - Datenformate bzw. -repräsentationen können verschieden sein
  - Nachrichtenweiterleitung bedingt Transformationsvorgänge

Netzübergangsknoten stellen **stabile Schnittstellen** zur Verfügung, ihr **funktionales Verhalten**, d.h. Typ und Semantik der umgeschlagenen Nachrichten, und ihr

**zeitliches Verhalten**, d.h. zu welchen Zeitpunkten, diese Nachrichten weitergeleitet werden,

sind idealerweise eindeutig spezifiziert.

## Netzübergangsknoten

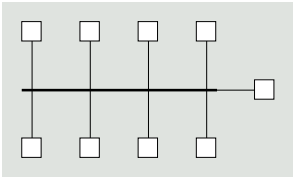


**Knoten G** verbindet zwei Subsysteme  $\mapsto$  Netzübergangsknoten

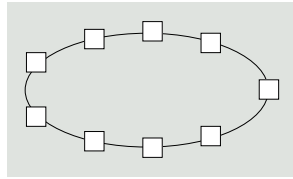
**Knoten I** bindet einen Sensor ein  $\mapsto$  Schnittstellenknoten

## Netzwerktopologie

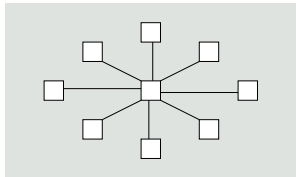
Physikalische Struktur des Kommunikationssystems



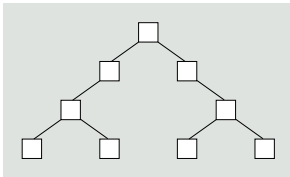
**Bus** (engl. *bus*), **passive Topologie**  
zentrales Medium



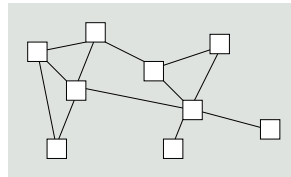
**Ring** (engl. *ring*), **aktive Topologie**  
verteilte Steuerung



**Stern** (engl. *star*)  
zentrale Verteilung



**Baum** (engl. *tree*)  
dezentrale Steuerung



**Masche** (engl. *mesh*)  
~ Internet

## Netzwerktopologie (Forts.)

Bus *versus* Ring — in Echtzeitkommunikationssystemen verbreitete Techniken

**Bus**  $\mapsto$  ein Hauptkabel, an dem alle Teilnehmer über spezielle T-Stücke (z.B. BNC, engl. *bayonet navy connectory*) angeschlossen sind

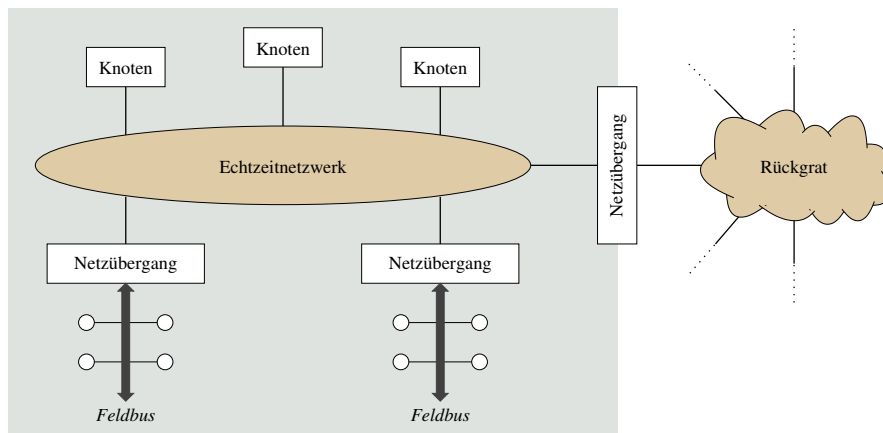
- Teilnehmer hören „gleichzeitig“, was auf dem Bus geschieht
  - **Simultanzustellung** von Nachrichten  $\leadsto$  *Multicast*
- Zugriffsverfahren koordinieren „gleichzeitige“ Sendevorgänge
  - Bus ist **gemeinsames** und teil- bzw. unteilbares **Betriebsmittel**
- unabhängig von Knotenfunktionen; Problem: Kabelbruch

**Ring**  $\mapsto$  Zweipunktverbindungen zwischen zwei Teilnehmern

- Nachrichten werden bis zum Bestimmungsort weitergeleitet
  - jeder Teilnehmer agiert als **Zwischenverstärker** (engl. *repeater*)
- Adressierung sorgt für überschneidungsfreies Senden
  - der Ring ist ein **abschnittsweise teilbares Betriebsmittel**
- abhängig von Knotenfunktionen; Problem: Teilnehmerausfall
  - Fehlerfall: ggf. Umschaltung der Drehrichtung des Arbeitswegs

## Netzwerkföderation

Organisation von Echtzeitnetzen [9, S. 156]



## Netzwerkföderation (Forts.)

Typen von Echtzeitnetzen

**Echtzeitnetzwerk** (engl. *real-time network*) Kern einer Gerätegruppe

- zuverlässige und zeitlich vorhersagbare Nachrichtenübertragung
  - insb. periodische Zustandsnachrichten mit impliziter Flusskontrolle
- Unterstützung für Fehlertoleranz: replizierte Knoten/Kanäle
  - Mitgliedsdienst (engl. *membership service*), Knotenausfallerkennung
- **Uhrensynchronisation** mit Auflösung im Mikrosekundenbereich

**Feldbus** (engl. *field bus*) Anschluss von Sensoren und Aktoren

- Netz von Mikrocontrollern ( $\mu C \mapsto$  Sensor und/oder Aktor)
- periodisch übertragene, kurze Nachrichten mit Zustandsdaten
- strikte Echtzeitanforderungen an Latenz und Latenzschwankungen
  - zieht meist präzise Uhrensynchronisation auf Busebene nach sich

**Rückgrat** (engl. *backbone network*) Verbindung zur „Außenwelt“

- Austausch zeitunkritischer Daten mit anderen Rechensystemen

## Gliederung

- 1 Überblick & Motivation
- 2 Anforderungen an verteilte Echtzeitsysteme
  - Zusammensetzbarkeit
  - Skalierbarkeit
  - Verlässlichkeit
- 3 Aufbau verteilter Echtzeitsysteme
  - Kommunikationssystem
  - Netzwerkschnittstelle
  - Topologie
- 4 Kommunikationssysteme
  - Anforderungen
  - Zeitliche Kapselung
  - Flusskontrolle
  - Netzzugangsprotokolle
- 5 Zusammenfassung

## Unterstützung für Zusammensetzbarkeit

### Architektonische Gesichtspunkte

#### Abkapselung des Zeitverhaltens von Knoten (engl. *temporal isolation*)

- Zeitverhalten an der Netzwerkschnittstelle ist vollständig bekannt
- **Brandmauer** (engl. *firewall*) gegen Steuerfehlerausbreitung
  - isolierte Prüfung der zeitl. Randbedingung von Anwendungssoftware

~> dadurch werden präzise Aussagen über einzelne Nachrichtenlaufzeiten im Kommunikationssystem möglich


#### Verpflichtungen der Klienten nachkommen $\mapsto$ Schutz des Anbieters

- **Überlastung** der Anbieter (engl. *server*) **vermeiden**
  - zuviele oder unkoordinierte Anforderungsnachrichten unterbinden
- Flusskontrolle der Dienstanforderungen der Klienten
  - Klienten helfen, ihre zeitlichen Verpflichtungen erfüllen zu können
- Anbietern ermöglichen, ihre Termine einhalten zu können
  - d.h. Nachrichten rechtzeitig zum Versand bereitstellen zu können

## Ereignisgesteuerte Kommunikationssysteme

Kommunikation basiert auf **Ereignisnachrichten** (s. Folie VIII/18)

- ~> der genaue Zeitpunkt des Nachrichtenversands ist **nicht bekannt**
  - der Knotenrechner entscheidet über die Versandoperation
  - dieser Zeitpunkt hängt von der dort gegenwärtigen Lastsituation ab
- ~> die Lastsituation im Kommunikationssystem ist **nicht bekannt**
  - sie hängt von den Versandoperationen einzelner Knoten ab
- ~> nimmt man neue Rechenknoten in das verteilte Systeme auf, können sie die Kommunikation existierender Knoten **erheblich beeinflussen**
  - auch wenn das Kommunikationssystem noch genügend Kapazität für neue Knoten besitzt und keine Überlastsituation erzeugt wird

 **Ereignisgesteuerte Kommunikationssysteme** sind nicht für die Kapselung des Zeitverhaltens geeignet!

## Zeitgesteuerte Kommunikationssysteme

Kommunikation basiert auf **Zustandsnachrichten** (s. Folie VIII/18)

- zumindest ist das Kommunikationssystem **autonom gesteuert**
- **Uhrensynchronisation** zwischen allen Kommunikationsteilnehmern
- ~> Nachrichten können nur an **definierten Zeitpunkten** versandt werden
  - das Kommunikationssystem unterbindet den unkoordinierten Nachrichtenversand und schützt somit den Kommunikationskanal
  - festgelegt durch **statische Sende- und Empfangstabellen**
- ~> im Kommunikationssystem ist die **Auslastung bestimmbar**
  - die Auslastung hängt lediglich von der Kommunikationssteuerung und nicht von den teilnehmenden Knoten ab
- ~> neu hinzukommende Knoten können die Kommunikation existierender Knoten nicht beeinflussen
  - nur die verfügbare Kommunikationsbandbreite entscheidet, ob ein neuer Knoten in das verteilte System aufgenommen werden kann

 **Zeitgesteuerte Kommunikationssysteme** kapseln das zeitliche Verhalten eines Knotens!

## Regelung des Datenflusses

Aufgabe der Netzwerkschicht des ISO OSI Referenzmodells [7]

Steuerung der Geschwindigkeit des Informationsflusses zwischen Sender Empfänger, so dass der Empfänger mit dem Sender Schritt halten kann

- Sender werden veranlasst, nur so viele Nachrichten zu übertragen, wie der Empfänger auch aufnehmen kann
- Empfänger bestimmen **maximale Kommunikationsgeschwindigkeit**

Zweck der Maßnahme ist es, eine Überschreitung der Aufnahmekapazität des Empfängers zu vermeiden und diesen nicht zu überlasten

- ereignisgesteuerte Systeme sind besonders von Überlast bedroht:
  - Nachrichtenversand/-empfang verursacht Unterbrechungen
  - Pufferplatz für zu sendende/empfangende Nachrichten ist begrenzt
  - Nachrichten werden von einzuplanenden/-lastenden Jobs verarbeitet
- die Steuerung des Informationsflusses geschieht **explizit** oder **implizit**

## Explizite Flusskontrolle (Forts.)

Bedeutung (für Echtzeitsysteme) haben Protokolle, die nach dem Schema „sende und warte“ (engl. *send and wait*, auch *stop and wait*) arbeiten

**PAR** (engl. *positive acknowledgement and retransmission*)

**senderseitige Schritte**  $\leadsto$  **Fehlermaskierung**

- die Quelle sendet ein Paket, startet einen Zeitgeber und erwartet eine Empfangsbestätigung, bevor ein neues Paket gesendet wird
- bleibt die Empfangsbestätigung aus, läuft der Zeitgeber ab und das Paket wird wiederholt gesendet
- ist die maximale Anzahl von Wiederholungen (desselben Pakets) erreicht, wird der Sendevorgang abgebrochen  $\mapsto$  **Exception**

**empfangsseitige Schritte**  $\leadsto$  **Duplikatunterdrückung**

- nimmt die Senke ein eingetroffenes Paket an, sendet sie eine Empfangsbestätigung an die Quelle zurück
- gleicht die Laufnummer des Pakets der des von derselben Quelle zuletzt empfangenen Pakets, wird das Paket verworfen

## Explizite Flusskontrolle

Voraussetzung — die jedoch oft übersehen wird — ist, dass sich ein Sender im Kontrollbereich eines Empfängers befindet

- ein Empfänger kann **Gegendruck** (engl. *back pressure*) auf den Sender ausüben, indem er die Übertragungsrate kontrolliert
  - **Flusskontrolle durch Gegendruck** (engl. *back-pressure flow control*)
- der Gegendruck des Empfängers äußert sich dadurch, dass beim Sender die Übertragung weiterer Daten hinausgezögert wird
  - empfangene Nachrichten werden ohne weitere Behandlung verworfen
  - Empfangsbestätigungen werden bewusst und gezielt zurückgehalten
- das weitere Vorankommen des Senders hängt ab vom Zustand und vom Verhalten des Empfängers

Protokolle mit **1-zu-1-Synchronisation** zwischen Sender und Empfänger bilden die Grundlage für **Ereignisnachrichten**

- Maximierung der Bandbreitenausnutzung ist nebensächlich (in EZS)

## Explizite Flusskontrolle (Forts.)

Gefahr vor Überlast durch „Flattern“ (engl. *thrashing*)

**Wiederholungen** von Nachrichten bei **Zeitüberschreitung** (engl. *timeout*)

- Ursache kann sein, dass das Kommunikationssystem die gegebene Last kaum noch bzw. nicht mehr bewältigen kann
  - anfällig sind Systeme, deren Normallast nahe der Maximallast liegt
  - Wiederholungen wegen Übertragungsfehler sind dann bes. kritisch
  - ein abrupter Leistungsabfall (Durchsatz) kann die Folge sein
- Überlast erhöht das Risiko von Zeitüberschreitungen, woraufhin zusätzliche Last anfällt. . .
  - die die bereits vorhandene Überlast weiter ansteigen lässt
  - die Zeitüberschreitungen dadurch noch wahrscheinlicher macht
- ebenso abrupt, wie die Überlastsituation aufgetreten ist, wird sie auch wieder verschwinden
  - ggf. muss nur eine einzige Kommunikation erfolgreich abschließen

**Thrashing ist unbedingt zu vermeiden** ( $\mapsto$  *rare-event situation*) und d.h.:

- kontinuierliche Überwachung der Betriebsmittelanforderungen
- Flusskontrolle durch Gegendruck, bei beobachtetem Leistungsabfall



## Implizite Flusskontrolle

Voraussetzung ist globale Zeit (engl. *global time*)

Sender und Empfänger treffen vorher (z.B. beim Systemstart) eine Übereinkunft über die Sendezeitpunkte von Nachrichten

- der Sender verpflichtet sich, Nachrichten nur zu den vereinbarten Zeitpunkten zum Empfänger zu versenden
- der Empfänger verpflichtet sich, alle Nachrichten des Senders zu empfangen, solange dieser seine Verpflichtung einhält

**unidirektionale Kommunikation**  $\mapsto$  Bestätigungen für eingetroffene

Nachrichten entfallen, Fehlererkennung ist Aufgabe des Empfängers:

- er weiß, wann eine erwartete Nachricht nicht mehr eintreffen kann
- globale Zeit erlaubt ihm, den Zeitpunkt auf  $t_s + d_{max}$  festzulegen
  - für jeden ihn betreffenden Sendezeitpunkt  $t_s$
  - und für die maximale Protokolllatenz  $d_{max}$

**Fehlertoleranz** (aktive Redundanz) durch *Multicast* ist gut umsetzbar

- gleichzeitige Übertragung von  $k$  Kopien derselben Nachricht
- bevorzugt über mehrere Kanäle, soweit verfügbar und möglich

## Gegenüberstellung

Vor dem Hintergrund *Hard Real-Time System* (HRTS, [9, S. 153])

Charakteristik	Flusskontrolle		HRTS
	explizit	implizit	
Steuersignal	Der Empfänger muss in der Lage sein, die Sendeereignisse des Senders steuern zu können.	Die Signale werden bei Fortschreiten der Echtzeit mit konstanter Rate generiert.	Der Empfänger kann die Ereignisse im Kontrollbereich des Senders nicht völlig kontrollieren.
Fehlererkennung	Sender	Empfänger	Empfänger
<i>Thrashing</i>	anfällig	nicht anfällig	ist zu vermeiden
<i>Multicast</i>	schwer	einfach	gefordert

Flusskontrolle macht die **Prozessschnittstelle** zwischen kontrolliertem Objekt und Echtzeitsystem besonders kritisch

- nicht alle Ereignisse, die im kontrollierten Objekt anfallen, werden im Kontrollbereich des Echtzeitsystems liegen
- ein **Alarmschauer** kann die Folge sein, wenn mehr Ereignisse im kontrollierten Objekt anfallen, als im Entwurf angenommen wurde

## Netzzugangsprotokolle

Verwalten den gemeinsam benutzten physikalischen Übertragungskanal

**Zugriffskontrolle** für die **exklusive Vergabe** eines Übertragungskanals leistet das **Netzzugangsprotokoll** (engl. *media-access protocol*)

- unterstützt (externe Kontrolle) bzw. implementiert (autonome Kontrolle) dabei die Kommunikationssteuerung
- vermeidet bzw. behandelt **Kollisionen** durch **gleichzeitiges Senden**
  - um einen Mindestdurchsatz zu garantieren und
  - Latenz(schwankungen) zu begrenzen
- als Verfahren kommen allgemein zum Einsatz:
  - CSMA (CA/CD/CR)
  - *Token*
  - *Minislotting*
  - *Master/Slave*
  - TDMA

## CSMA

(engl. *carrier sense multiple access*)

Klasse von verteilt arbeitenden Zugriffsverfahren, die keine zentrale Kontrolle (beim Schreiben) erfordern

- jeder Teilnehmer hört beim Schreiben gleichzeitig den Bus auf kollidierende Schreibzugriffe ab  $\mapsto$  *carrier sense*

**CA** (engl. *collision avoidance*)

- Kollisionsvermeidung durch zufällige Wartezeit zum Schreiben

**CD** (engl. *collision detection*)

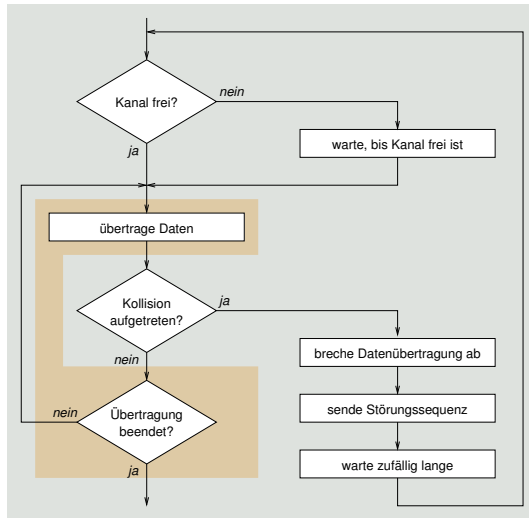
- im Konfliktfall werden die Schreibzugriffe zurückgenommen
- jeder betroffene Teilnehmer wartet zufällig lang
- nach den Wartephasen werden die Schreibzugriffe wiederholt

**CR** (engl. *collision resolution*) z.B. durch Bitarbitrierung

- Nachrichten gehen eindeutige Identifikationen/Prioritäten voran
- im Konfliktfall geben „Verlierer“ ihre Schreibzugriffe auf
- nach endlich vielen Schritten/Takten bleibt ein Gewinner übrig

# CSMA/CD

Standardprotokoll für Halbduplexbetrieb



## Ethernet (Xerox [14])

- lokales Netzwerk
  - quittierungsfrei
- Varianten:
  - thick/thin
  - fast/gigabit
- Probleme:
  - Wartezeiten
  - Fairness

# CSMA/CR

Kollisionenauflösung durch Arbitrierung: Bitarbitrierung

## CAN (engl. control area network, [8]) Automobilindustrie

- jeder gesendeten Nachrichten geht ein **Nachrichtentyp** voran:
  - beschreibt den Inhalt einer Nachricht, verwendet zur Arbitrierung
  - je kleiner der Wert des Typs, desto höher die Priorität der Nachricht
- die Übertragung beginnt mit dem höchstwertigsten Bit des Typs
  - sendet ein Knoten eine 1 (rezessives Bit), empfängt er aber eine 0 (dominantes Bit), so bricht er seine Übertragung ab
  - erneuter Sendeversuch, nachdem **Busruhe** erkannt wurde: 11 Bitzeiten auf Ruhepotential (rezessiver Buspegel)

Beispiel: vier Knoten senden gleichzeitig...

Nachrichtentyp 1431 → ~~1 0 1 1 0 0 1 0 1 1 1~~  
 Nachrichtentyp 1337 → ~~1 0 1 0 0 1 1 1 0 0 1~~  
 Nachrichtentyp 1335 → ~~1 0 1 0 0 1 1 0 1 1 1~~  
 Nachrichtentyp 1332 → 1 0 1 0 0 1 1 0 1 0 0  
**Bus** 1 0 1 0 0 1 1 0 1 0 0 ~ 1332 hat Vorrang

# CSMA/CD (Forts.)

Verringerung der Wahrscheinlichkeit von Kollisionen

## LON (engl. local operating network, [4]) Gebäudeautomatisierung

- Busteilnehmer (Knoten): **Neuron-Chip** mit drei 8-Bit Prozessoren
  - network CPU → Schichten 2–6 des OSI-Referenzmodells
  - media-access CPU → Netzzugangsprotokoll (Schicht 1)
  - application CPU → Anwendung, Knotenfunktion
- Knoten greifen nach zufällig langer Verzögerung auf den Bus zu
  - am Anfang von (regulären) Übertragungen
  - während Übertragungswiederholungen als Folge von Kollisionen
  - nach Rücknahme des Trägers (engl. carrier) des vorherigen Transfers
- die Größe des „Wartefensters“ ist eine **Funktion der Kanallast**
  - speziell ausgelegt, um die Wahrscheinlichkeit von Kollisionen bei hoher Last zu minimieren → **p-persistentes CSMA**
  - Flusskontrolle kommt durch **stochastischen Gegendruck** zustande (engl. stochastic back-pressure flow control)

# Token

Vergabe von Übertragungsrechten

Knoten im Besitz eines Übertragungsrechtes (→ **Token**) können den Bus zum Schreiben verwenden

- das Bussystem ist als **Ring** ausgelegt (engl. token-ring bus)
  - das **Token** umläuft den Ring von Knoten zu Knoten: **aktive Topologie**
- das Antwortverhalten bestimmt sich durch zwei Zeitparameter:
  - THT** (engl. token-hold time)
    - längste Zeit, für die ein Knoten den **Token** halten darf
  - TRT** (engl. token-rotation time)
    - längste Zeit, die ein **Token** zum kompletten Umlauf benötigt
- erstes Problem: Ausfall des Knotens, der den **Token** besitzt
  - Verlust des **Token** wird durch Zeitüberschreitung festgestellt
  - einer der anderen Knoten (welcher?) erzeugt einen neuen **Token**

## Profibus (engl. process field bus, [3]) Prozessautomatisierung

## Minislotting

Zeitkontrolliertes Zugangsverfahren

Zeit wird in eine Folge von „Minischlitzen“ (engl. *mini slot*) unterteilt, jeder länger als die Ausbreitungsverzögerung des Kanals

- jedem Knoten ist eine eindeutige Anzahl von *Minislots* zugeordnet
  - zu verstreichende Zeit der Ruhe (auf dem Kanal) vor dem Schreiben

ARINC 629 (*Aeronautical Radio Incorporated*, [1]) Flugzeugindustrie

- ein **Warteraumprotokoll** ähnlich zum „Bäckereialgorithmus“ [11]
  - in einem ersten Zeitintervall finden sich Prozesse, die schreiben wollen, in einen (verteilten) Warteraum ein
  - im nachfolgenden Zeitintervall ( $\mapsto$  Epoche) können alle wartenden Prozesse schreiben, bevor neue den Warteraum betreten dürfen
- kein (böswilliger) Knoten kann den Bus monopolisieren
  - **Busschutz** (engl. *bus protection*)

## Minislotting (Forts.)

ARINC 629 — Zeitparameter zur Kontrolle des Zugangs zum Übertragungsmedium

SG (engl. *synchronization gap*)

- kontrolliert Zutritt zum Warteraum: identisch für alle Knoten

TG (engl. *termination gap*)

- kontrolliert den Buszugriff: unterschiedlich für jeden Knoten

TI (engl. *transmit interval*)

- verhindert Monopolisierung des Busses: identisch für alle Knoten

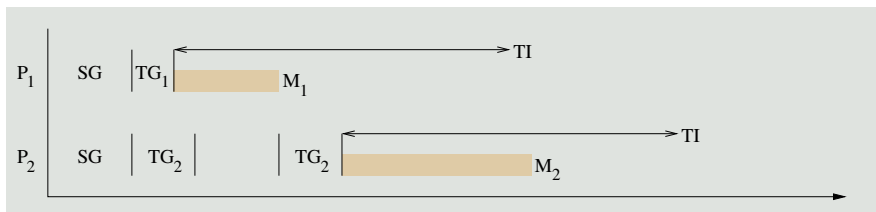
Relationen, die zwischen den Zeitparametern (*Timeouts*) definiert sind:

- $SG > \max(TG_i)$ , für alle Knoten bzw. Prozesse  $i$
- $TI > SG$

## Minislotting (Forts.)

ARINC 629 — Protokollverlauf

Prozesse  $P_1$  und  $P_2$  wollen gleichzeitig senden,  $TG_1 < TG_2$ :



- 1  $P_1$  und  $P_2$  warten SG Zeiten Busruhe ab, betreten den Warteraum
- 2 jeder Prozess  $i$  wartet zusätzlich noch seine  $TG_i$  Zeiten Busruhe ab
- 3 wegen  $TG_1 < TG_2$  sendet  $P_1$  zuerst seine Nachricht  $M_1$
- 4  $P_2$  erkennt Verkehr auf den Bus; wartet, bis  $M_1$  übertragen wurde
- 5  $P_2$  wartet  $TG_2$  Zeiten Busruhe ab und sendet seine Nachricht  $M_2$
- 6  $P_1$  und  $P_2$  können frühestens nach TI Zeiten erneut senden

## Master/Slave

Zentraler *Master* kontrolliert Buszugriffe

FIP (*factory instrumentation protocol*, [12]) Prozessautomatisierung

- arbeitet nach dem Modell „Produzent-Verteiler-Konsument“
  - Produzent**  $\mapsto$  pro Transaktion genau ein Sender
  - Verteiler**  $\mapsto$  Schiedsrichter (engl. *bus arbitrator*, BA)
  - Konsument**  $\mapsto$  pro Transaktion ggf. mehrere Empfänger
- Transaktionen verlaufen periodisch und in vier Schritten ab:
  - 1 BA gibt einen „Variablennamen“ per Sammelaufruf bekannt
  - 2 Produzent und Konsumenten erkennen die aufgerufene Variable
  - 3 der Produzent gibt den Variablenwert an die Konsumenten ab
  - 4 die Konsumenten nehmen den Variablenwert an, sofern benötigt
- Variablennamen sind systemweit eindeutige Bezeichner für...
  - *Boolean*, *Integer*, *Bitstring*, *Bytestring*, *General Time*, Verbünde
- freie Zeit kann für **sporadische Daten** genutzt werden
  - abfragen (engl. *polling*) der Knoten durch den BA

LIN (*local interconnect network*, [16]) Automobilindustrie

# TDMA

(engl. *time division multiple access*)

Übertragungsrechte werden durch Voranschreiten der Echtzeit vergeben:

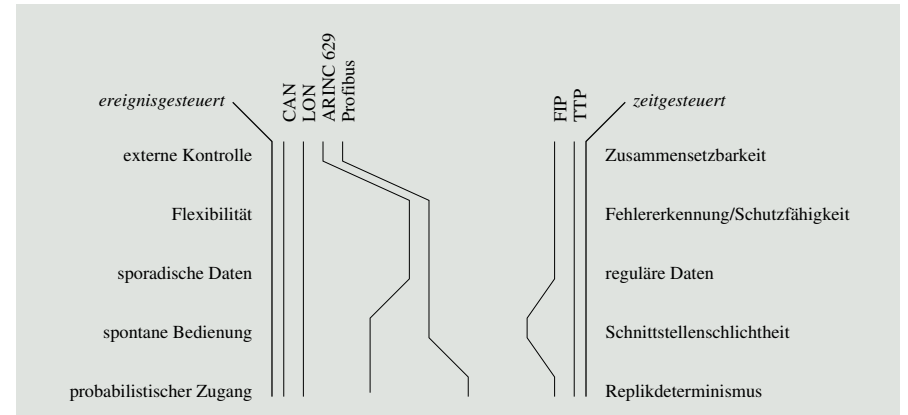
- Voraussetzung: **fehlertolerante globale Zeitbasis** in allen Knoten
- statische Aufteilung der gesamten Kanalkapazität in **Zeitschlitz**
  - jeder Knoten (Busteilnehmer) hat einen eindeutigen Sendeschlitz
  - **TDMA Runde** → Sequenz von Sendeschlitz einer Knotengruppe
    - in jeder Runde kann ein Knoten eine Nachricht übertragen
    - ist nichts zu versenden, bleibt ein Rahmen (engl. *frame*) leer
  - Runden wiederholen sich → **Gruppentakt** (engl. *cluster cycle*)
    - Sequenz verschiedener TDMA-Runden
  - die Gruppentaktlänge bestimmt die Periodizität des TDMA-Systems

TTP (engl. *time-triggered protocol* [10])

- Varianten, die in der Automobilindustrie Verwendung finden:
  - *byeflight* ([2], Sicherheits- und Informationsbussystem: **SI-BUS**)
  - *FlexRay* ([5], zeit- und ereignisgesteuerter Bus)

# Vergleich

Entwurfsentscheidungen [9, S. 164]



Flexibilität } **versus** { Zusammensetzbarkeit  
 sofortige Antwort }  
 sporadische Daten } reguläre Daten

# Ereignisgesteuerte Kommunikationssysteme

Transport von Ereignisnachrichten

Rechtzeitigkeit bzw. zeitliche Kontrolle ist eine **globale Angelegenheit** des gesamten verteilten Rechensystems

- bei ereignisgesteuerten Protokollen ist zeitliche Kontrolle an der Netzwerkschnittstelle undefiniert; das bedeutet:
  - ein einziger gemeinsamer Transportkanal schürt **Zugriffskonflikte**
    - Lösungsansätze sind zufällige Zugriffe (Ethernet), vorgegebene Zugriffsreihenfolgen (*token ring*) und priorisierte Nachrichten (CAN)
    - ohne jedoch das Grundproblem vom Tisch zu bekommen...
  - bei getrennten Transportkanälen droht **Überlastung** des Empfängers
- diese Kontrolle ist weiter oberhalb (des KS) sicherzustellen
  - in der Diensteschicht (engl. *middleware*) bzw. verteilten Anwendung
  - sie muss **nicht deterministisches Systemverhalten** „kaschieren“
  - vergleichsweise leicht bei weicher Echtzeit, schwer bis unmöglich sonst

die Architektur ist **nicht zusammensetzbar** bzgl. Rechtzeitigkeit

# Zeitgesteuerte Kommunikationssysteme

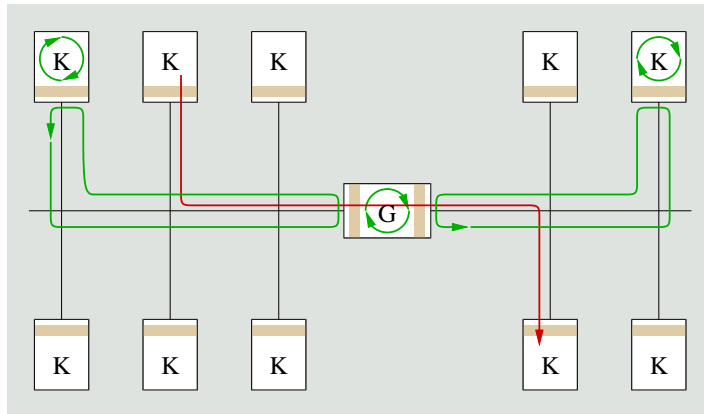
Transport von Zustandsnachrichten

Rechtzeitigkeit bzw. zeitliche Kontrolle ist eine **lokale Angelegenheit** des Kommunikationssystems

- bei zeitgesteuerten Protokollen ist die zeitliche Kontrolle an der Netzwerkschnittstelle wohl definiert
  - Nachrichten werden zu festen, vorgegebenen Zeitpunkten transferiert
    - auf Basis einer **Ablaufabelle** in der Kommunikationssteuerung
  - Knotenrechner haben keinen Einfluss auf das Zeitverhalten des KS
    - die Netzwerkschnittstelle ist frei von Steuersignalen
    - sie hat eine **Daten teilende** (engl. *data sharing*) **Semantik**
    - **Steuerfehlerausbreitung** (*control-error propagation*) ist **unmöglich**
- sämtliche zeitlichen Eigenschaften wurden beim Entwurf festgelegt
  - Knoten sind unabhängig von der Netzwerkschnittstelle testbar
  - Systemintegration verändert nicht das Zeitverhalten der Schnittstelle

die Architektur ist **zusammensetzbar** in Bezug auf Rechtzeitigkeit

## Zeitgesteuerte vs. Ereignisgesteuerte Kommunikation



**Ereignissteuerung**  $\mapsto$  Steuersignale passieren die Netzwerkschnittstelle!

**Zeitsteuerung**  $\mapsto$  Steuersignale passieren die Netzwerkschnittstelle nicht!

## Gliederung

- 1 Überblick & Motivation
- 2 Anforderungen an verteilte Echtzeitsysteme
  - Zusammensetzbarkeit
  - Skalierbarkeit
  - Verlässlichkeit
- 3 Aufbau verteilter Echtzeitsysteme
  - Kommunikationssystem
  - Netzwerkschnittstelle
  - Topologie
- 4 Kommunikationssysteme
  - Anforderungen
  - Zeitliche Kapselung
  - Flusskontrolle
  - Netzzugangsprotokolle
- 5 Zusammenfassung

## Resümee

**Erscheinungsform**  $\mapsto$  **verteiltes Echtzeitrechnungssystem**

- physikalische Verteilung, steigende Komplexität

**Anforderungen** an verteilte Echtzeitsysteme

- Erhöhung der Rechenleistung, Beherrschung von Komplexität
- Zusammensetzbarkeit, Skalierbarkeit, Verlässlichkeit

**grundlegender Aufbau** verteilter Echtzeitrechnungssystem

- Netzwerkschnittstelle, Kommunikationssteuerung, Topologie
- externe vs. autonome Kontrolle; Ereignis- vs. Zustandsnachrichten

**Kommunikationssysteme** für verteilte Echtzeitsysteme

- **Ereignissteuerung**: externe Kontrolle + Ereignisnachricht
- **Zeitsteuerung**: autonome Kontrolle + Zustandsnachricht
- **zeitliche Kapselung** + **implizite Flusskontrolle**  $\leadsto$  zusammensetzbar
- **Netzzugangsprotokolle** bestimmen Kommunikationssteuerung

## Literaturverzeichnis

- [1] AUDSLEY, N. C. ; GRIGG, A. :  
Timing Analysis of the ARINC 629 Databus for Real-Time Applications.  
In: *Proceedings of the ERA Avionics Conference and Exhibition*.  
Heathrow, UK : ERA Technology Ltd, Nov. 20–21, 1996, S. 10.1.1–10.1.11
- [2] BERWANGER, J. ; PELLER, M. ; GRIESSBACH, R. :  
*byflight* — A New Protocol for Safety Critical Applications.  
In: *Proceedings of the 28th FISITA World Automotive Congress*.  
Seoul, Korea : FISITA, Jun. 12–15, 2000
- [3] *Kapitel 3*.  
In: DEUTSCHES INSTITUT FÜR NORMUNG:  
*Der Profibus*.  
Berlin, Köln : Beuth-Verlag, 1991 (DIN 19245)
- [4] ECHELON CORPORATION:  
*Enhanced Media Access Control with LonTalk Protocol*.  
Jan. 1995
- [5] FLEXRAY CONSORTIUM:  
*FlexRay protocol specification 2.1 Revision A*.  
FlexRay Consortium, 2005. –  
<http://www.flexray.com>

## Literaturverzeichnis (Forts.)

- [6] HESS, R. K. ; BASS, D. I. ; BACA, J. B.:  
Control systems: crashproof code.  
In: *IEEE Spectr.* 41 (2004), Sept., S. 48–53.  
<http://dx.doi.org/10.1109/MSPEC.2004.1330810>. –  
DOI 10.1109/MSPEC.2004.1330810. –  
ISSN 0018–9235
- [7] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION:  
*Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model.*  
ISO, 1994 (ISO/IEC 7498-1)
- [8] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION:  
*Road vehicles — Control area network (CAN) — Parts 1–4.*  
ISO, 2003 (ISO 11898)
- [9] KOPETZ, H. :  
*Real-Time Systems: Design Principles for Distributed Embedded Applications.*  
Kluwer Academic Publishers, 1997. –  
ISBN 0–7923–9894–7

## Literaturverzeichnis (Forts.)

- [15] POLEDNA, S. :  
*Replica Determinism in Fault-Tolerant Distributed Real-Time Systems.*  
Vienna, Austria, Technical University of Vienna, Diss., 1995. –  
Research Report 28/95
- [16] SPECKS, J. W. ; RAJNÁK, A. :  
LIN—Protocol, Development Tools, and Software Interfaces for Local Interconnect Networks in Vehicles.  
In: *Proceedings of 9th International Conference on Electronic Systems for Vehicles.*  
Baden-Baden, Germany, Okt. 5/6, 2000

## Literaturverzeichnis (Forts.)

- [10] KOPETZ, H. ; GRÜNSTEIDL, G. :  
TTP—A Time-Triggered Protocol for Fault-Tolerant Real-Time Systems.  
In: *Proceedings of the Twenty-Third Annual International Symposium on Fault-Tolerant Computing (FTCS-23).*  
Toulouse, France : IEEE, Jun. 22–24, 1993, S. 524–533
- [11] LAMPORT, L. :  
A New Solution of Dijkstra's Concurrent Programming Problem.  
In: *Communications of the ACM* 8 (1974), Nr. 7, S. 453–455
- [12] LETERRIER, P. :  
The FIP Protocol / WorldFIP Europe.  
Nancy, France, 1992. –  
Forschungsbericht
- [13] MALEK, M. :  
Responsive Computer Systems.  
In: *Real-Time Systems* 7 (1994), Nr. 3. –  
Special Issue
- [14] METCALFE, R. M. ; BOOGS, D. R.:  
Ethernet: Distributed Packet Switching for Local Computer Networks.  
In: *Communications of the ACM* 19 (1976), Jul., Nr. 5, S. 395–404