

**Eigenschaften mobiler und
eingebetteter Systeme:**

Beispielarchitekturen

EMES

Dr.-Ing. Matthias Werner

Dipl.-Inf. Jan Richling

Wintersemester 2001/2002

Warum Systemarchitekturen?

- Ein Echtzeitsystem ist mehr als die Zusammenhäufung seiner Bestandteile
 - Hardware
 - Betriebssystem
 - Anwendungssoftware
 - Netzwerkhardware
 - Externe Hardware (Sensoren, Aktuatoren)
 - Netzwerksoftware, Netzwerkprotokolle
 - Nutzerschnittstellen
- Architektur überdeckt alle Schichten und alle “Komponenten”
 - Regeln für “Bauteile” eines Systems
 - Regeln zum Bau von Systemen

Ziele

- Architektur soll Eigenschaften von Systemen “per Konstruktion” sicherstellen
 - Erreichbarkeitseigenschaften
 - Architektur erlaubt den Bau von Systemen mit diesen Eigenschaften
 - Sicherheitseigenschaften
- Architektur erlaubt ausschließlich den Bau von Systemen mit diesen Eigenschaften
- Verlagerung der Einhaltung von Eigenschaften von Systementwurf hin zum Architekturentwurf
 - Benutzung einer entsprechenden Architektur sichert Eigenschaften zu, um die sich der Systemdesigner nicht kümmern muß
 - Trennung von funktionalen und nichtfunktionalen Parametern (Aspekten)

Eigenschaften von Architekturen

- Systemeigenschaften: Eigenschaften eines Systems, das nach den Regeln einer Architektur gebaut wurde
- Eigenschaften einer Architektur
 - Eigenschaften, die alle konstruierbaren Systeme betreffen
 - Eigenschaften, die mit den Regeln der Architektur erzielbar sind
 - Eigenschaften der Konstruktionsregeln
 - Eigenschaften der von der Architektur erlaubten “Bauteile”

Komponentenorientierte Sicht

- Komponente oder Element: Element einer Menge von Bauteilen
 - Erlaubte Menge von Bauteilen ist durch Architektur definiert
 - “Atomare” Komponenten sind nicht aus anderen Komponenten komponiert
- Komposition: Zusammenfügen von Komponenten
 - Architektur definiert Kompositionoperator(en) (was darf zusammengefügt werden?)
 - Ergebnis der Komposition ist wieder eine Komponente
 - System als “Ende” der Komposition ist ebenfalls eine Komponente
 - Architektur gibt über den Kompositionoperator die Schnittstellen der Komponenten vor

Arten von Architekturen (Beispiele)

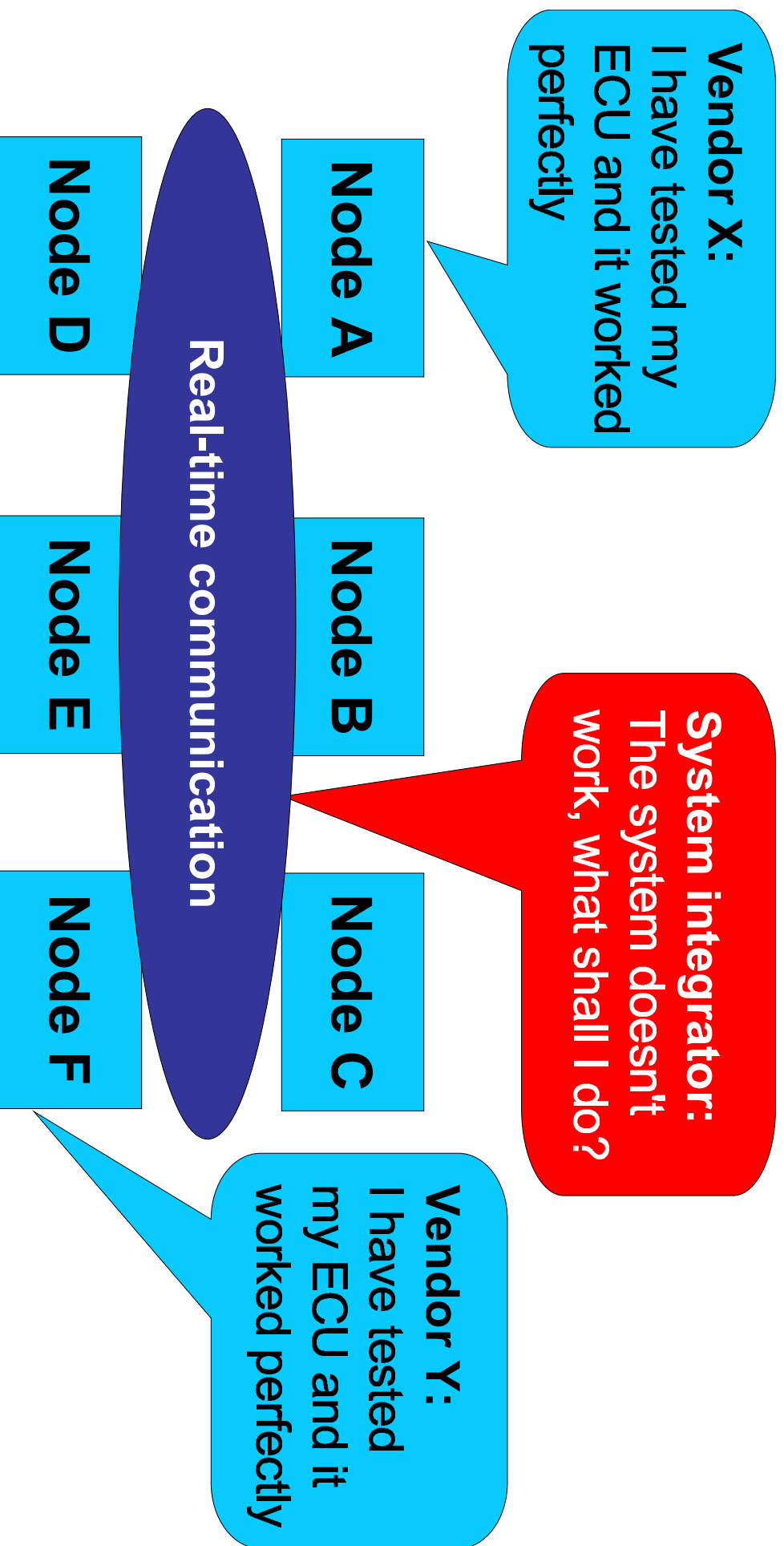
- LEGO-Architektur
 - Jede atomare Komponente kann mit jeder anderen atomaren Komponente verbunden werden
- “Plumbing”-Architektur
 - Unterscheidung “Konnektor” und “Komponente”
 - Konnektoren können mit Konnektoren oder Komponenten verbunden werden
 - Komponenten können nur mit Konnektoren verbunden werden
- “Powergrid”-Architektur
 - Unterscheidung “Konnektor” und “Komponente”
 - Konnektoren können nur mit Komponenten verbunden werden
 - Komponenten können nur mit Konnektoren verbunden werden
- “Backplane”-Architektur
 - Wie “Powergrid”, aber
 - Nur ein einziger Konnektor: Backplane

Komponierbarkeit I

- Eigenschaft einer Architektur
- Intuitiv: “Zusammenfügbar sein”
- Definition:
 - Eine Systemarchitektur ist komponierbar in Bezug auf eine Sicherheitseigenschaft, wenn sie ausschließlich die Komposition von Systemen erlaubt, die diese Eigenschaft besitzen
 - Eine Systemarchitektur ist komponierbar in Bezug auf eine Erreichbarkeitseigenschaft, wenn sie die Komposition von wenigstens einem System erlaubt, das diese Eigenschaft besitzt.
- Beispiel:

Komponierbarkeit in Bezug auf die Erhaltung von zeitlichen Eigenschaften

Komponierbarkeit II



Komponierbarkeit III

- Bei Nicht-Echtzeitsystemen erlauben Objektorientierung und Komponentensysteme wie CORBA o.ä. derartige Kompositionen
- Nicht übertragbar auf Echtzeitsysteme!
 - Ergebnisse hängen nicht nur vom funktionalen Verhalten ab
 - Ergebnisse sind auch durch das nichtfunktionale Verhalten bestimmt
 - Funktionale Komposition (Interface A ist kompatibel zu Interface B) sagt nichts über nichtfunktionale Eigenschaften aus
- Ansätze
 - Anpassung von Komponentensystemen
 - * RT-Corba
 - * FT-Corba
 - Entwicklung von komponierbaren Architekturen
 - Aspektorientierte Programmierung (AOP)

Beispielarchitekturen

- Architekturen mit Unterstützung für Komponierbarkeit
 - Verständnis von Komponierbarkeit ist verschieden, da keine allgemeine Definition
 - Erfüllung des intuitiven Ziels “Zusammenfügbarkeit”
- Beispiele
 - Time Triggered Architecture (TTA)
 - * Forschungsgruppe von Prof. Kopetz an der TU Wien
 - * Nachfolger des MARS Projektes (Maintainable Real-Time System)
 - Message Scheduled System (MSS)
 - * Rechnerorganisation und -kommunikation der HU Berlin

Time Triggered Architecture (TTA)

- Ziel: Komponierbare Architektur für verteilte Echtzeitsysteme in sicherheitskritischen Applikationen
 - brake-by-wire
 - steer-by-wire
 -
- Adressierte Probleme:
 - Fehlertolerante Uhrensynchronisation
 - Echtzeit-Membership-Service
 - Management von Rekonfigurationen
 - Fail-silence in der Zeitdomäne — Vermeidung von “Babbling Idiot”

Komponierbarkeit in TTA I

- Eigenschaft einer Komponente

- Definition:

A component is said to be composable with respect to a certain property if and only if the system integration will not invalidate this property once it has been established at the component level.

- Unterschiede zur anderen Definition
 - Keine Unterscheidung Systemeigenschaft/Komponenteneigenschaft
 - Keine Berücksichtigung von neu auftauchenden (System-) Eigenschaften
 - Komponierbarkeit ist einer Komponente zugeordnet, nicht der Architektur
 - Blick auf Komponente unterstellt dem System ein der Komponente angepaßtes Verhalten

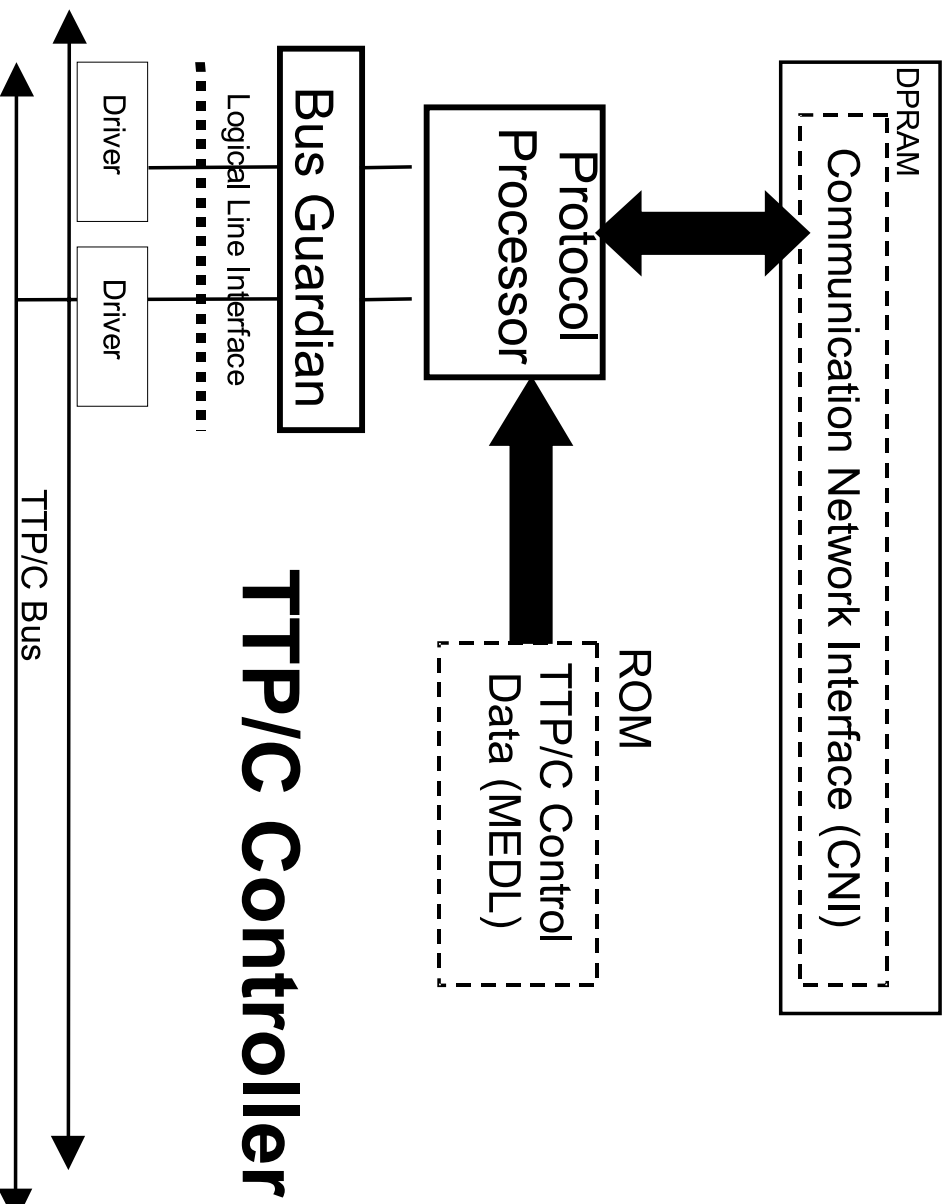
Komponierbarkeit in TTA II

- Komponierbarkeit beschränkt sich auf Erhaltung existierender Eigenschaften sowie deren “Übertragung” auf Systemebene
- Komponierbarkeit wird durch Vorwissen über mögliche Kompositionen erreicht
 - Ohne Neukonfiguration nur bekannte Komponenten möglich
 - Ohne Neukonfiguration nur bekannte Interfaces möglich
 - Unflexibel
 - Sicher (safe)
 - Replikat-Determiniertheit
- Trennung der funktionalen Interfaces von den nichtfunktionalen
- Übertragung von State-Informationen statt Event-Informationen

Architektur eines TTA-Knotens I

- Anwendungsprogramm und Kommunikationssystem vollständig getrennt
- Kontrollsignale können Interfaces nicht passieren
- Kontrolle des Kommunikationssystems unterliegt vollständig dem Kommunikationscontroller
 - Kommunikationssystem sendet Daten time triggered
 - Anwendungsprogramm hat keine "send"-Primitive
 - Zeitliche Fehler des Anwendungsprogrammes können sich nicht über das Interface hinaus ausbreiten (error containment regions, temporal firewall)
- Jede Komponente kann anhand ihres Interfaces in der Zeit- und Wertedomäne isoliert getestet werden

Architektur eines TTA-Knotens II

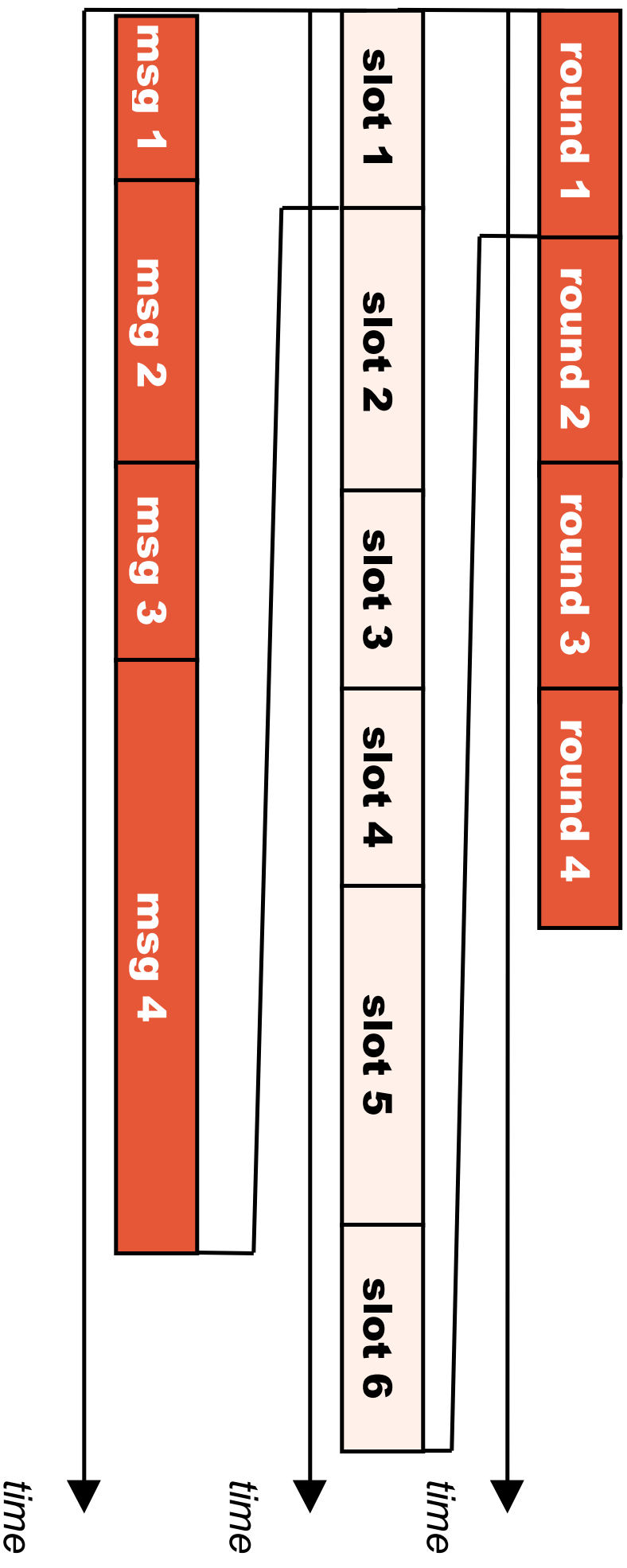


TTP/C Controller

TTA — Buszugriff I

- TDMA (Time Division Multiple Access)
- Verfügbare Zeit ist in Slots für Nachrichten eingeteilt
- Slots sind Nachrichten fest und a priori zugeordnet
- Zuordnung ist in MEDL (Message Descriptor List) gespeichert in allen Kommunikationscontrollern
- Verschiedene Runden mit verschiedenen MEDLs für Mode-Changes
- Eigenschaften
 - Jitterfrei
 - Präzise in der Zeit-Domäne definiert
 - Verletzungen des zeitlichen Verhaltens leicht "sichtbar"
- Unterstützung für
 - Fehlertoleranz (duplizierter Bus)
 - Uhrensynchronisation mit bis zu einer Mikrosekunde Genauigkeit
 - Membership

TTA — Buszugriff II



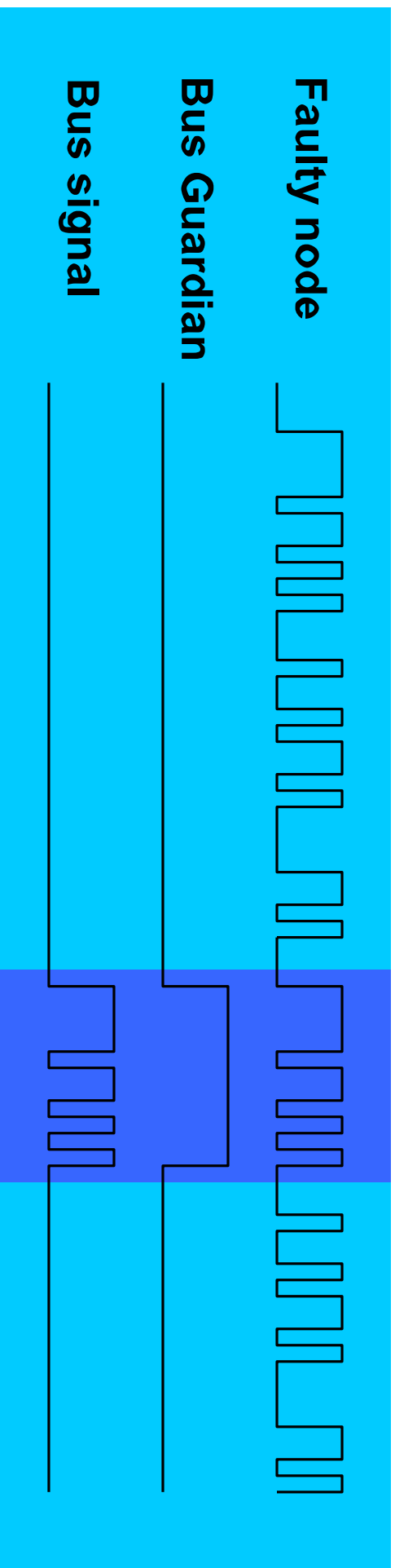
Echtzeit-Membership-Service

- TTP erfordert konsistente Sicht auf Membership
- “Alive”-Signale sind überflüssig
 - Verschwendung von Bandbreite
 - A priori Wissen erlaubt Ausfallerkennung ohne Overhead
- Protokoll:
 - Senden von C-States (Teil einer Nachricht, enthält Bitvektor für alle Knoten)
 - Ankunft von korrekten Nachrichten führt zur Aufnahme/Erhalt des entsprechenden Knotens in C-State
 - Fehlerhafte oder fehlende Nachrichten führen zur Löschung des entsprechenden Bits im C-State
 - Cliquen-Bildung:
 - * z.B. durch defekten Receiver
 - * wird behandelt durch Majorität — Knoten mit Minorität wird als ausgefallen betrachtet

Vermeidung von “Babbling Idiot” I

- Problem: Fehlerhafter Knoten sendet ständig Nachrichten und stört damit die Kommunikation
 - Fehler im Anwendungsprozeß werden durch temporalen Firewall aufgefangen
 - Bleibt: Behandlung von Fehlern des Kommunikationscontrollers
- Idee: Bus Guardian überwacht elektrischen Buszugriff des Knotens
 - Zeitpunkte des Zugriffes sind a priori bekannt
 - Kontrolle erfordert kein Wissen der Anwendungslogik
 - MEDL genügt als Datenbasis
- Funktion: Bus Guardian erlaubt Sendungen nur im Slot, der für den Knoten reserviert ist

Vermeidung von “Babbling Idiot” !!



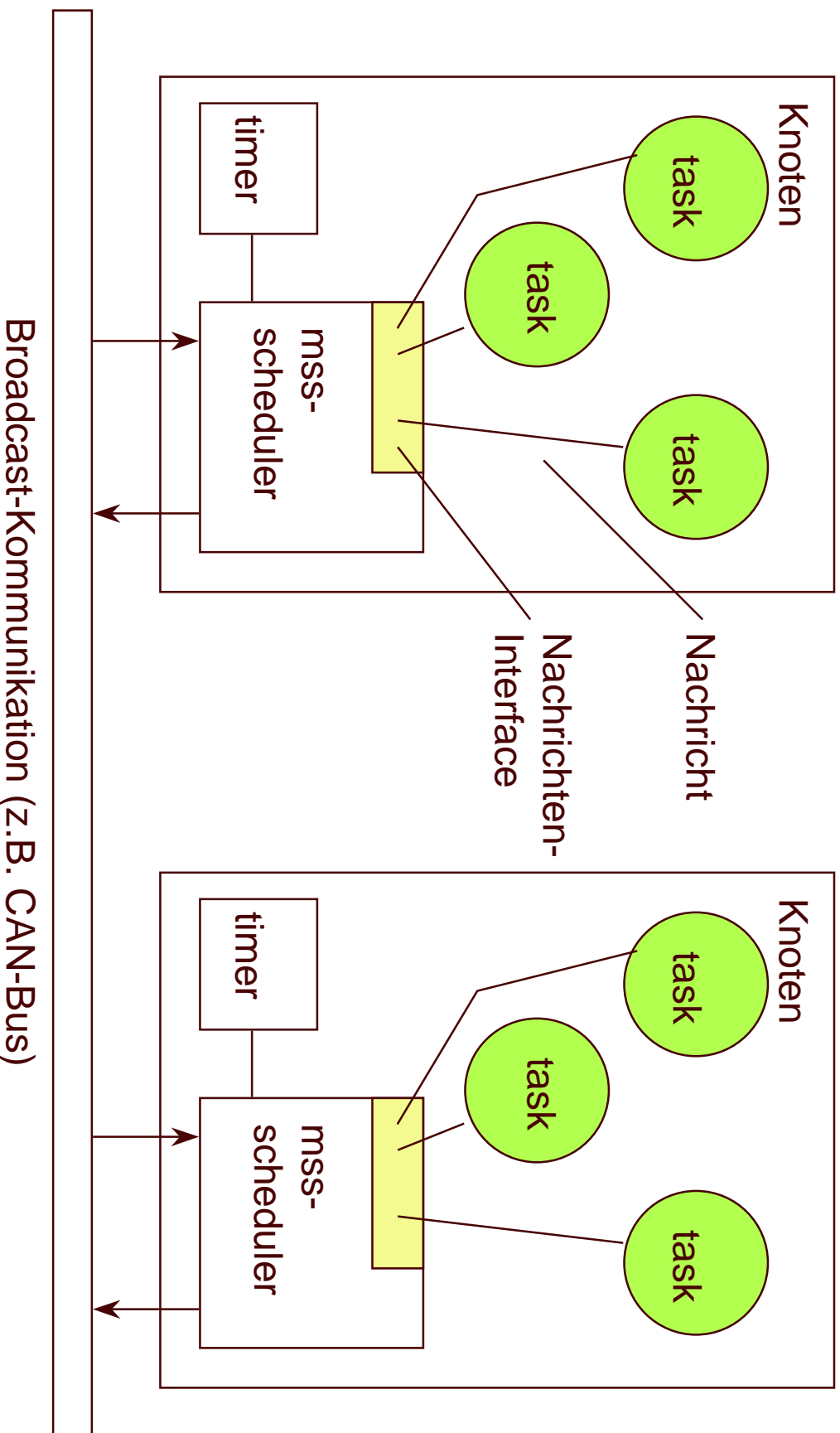
Message Scheduled System (MSS)

- **Ziel:**
 - Eingebettete Echtzeitsysteme (Fahrzeuge, Flugzeuge, Automatisierung)
 - Entwicklung einer Architektur mit Unterstützung von Komponierbarkeit in Bezug auf das zeitliche Verhalten
- **Idee:**
 - globales Scheduling mit lokalem Wissen
 - Mehrstufige Abbildung von Komponierbarkeits- auf Schedulingentscheidungen
- **Voraussetzungen:**
 - Echtzeitfähige Knoten an einem Echtzeitkommunikationsmedium
 - Globale Prioritäten

MSS — Arten von Komponenten

- **Tasks**
erzeugen aus einem Satz von periodischen Eingangsnachrichten einen Satz von Ausgangsnachrichten (mit Deadline)
- **Knoten**
führt Tasks aus; verfügt über “Nachrichtenmanager”, der den Nachrichtentrffic der Tasks verwaltet und über die Nutzung des Kommunikationsmediums entscheidet
- **Kommunikationsmedium**
echtzeitfähiger Bus, der die Knoten verbindet

MSS — Architektur



MSS — Komponierbarkeit I

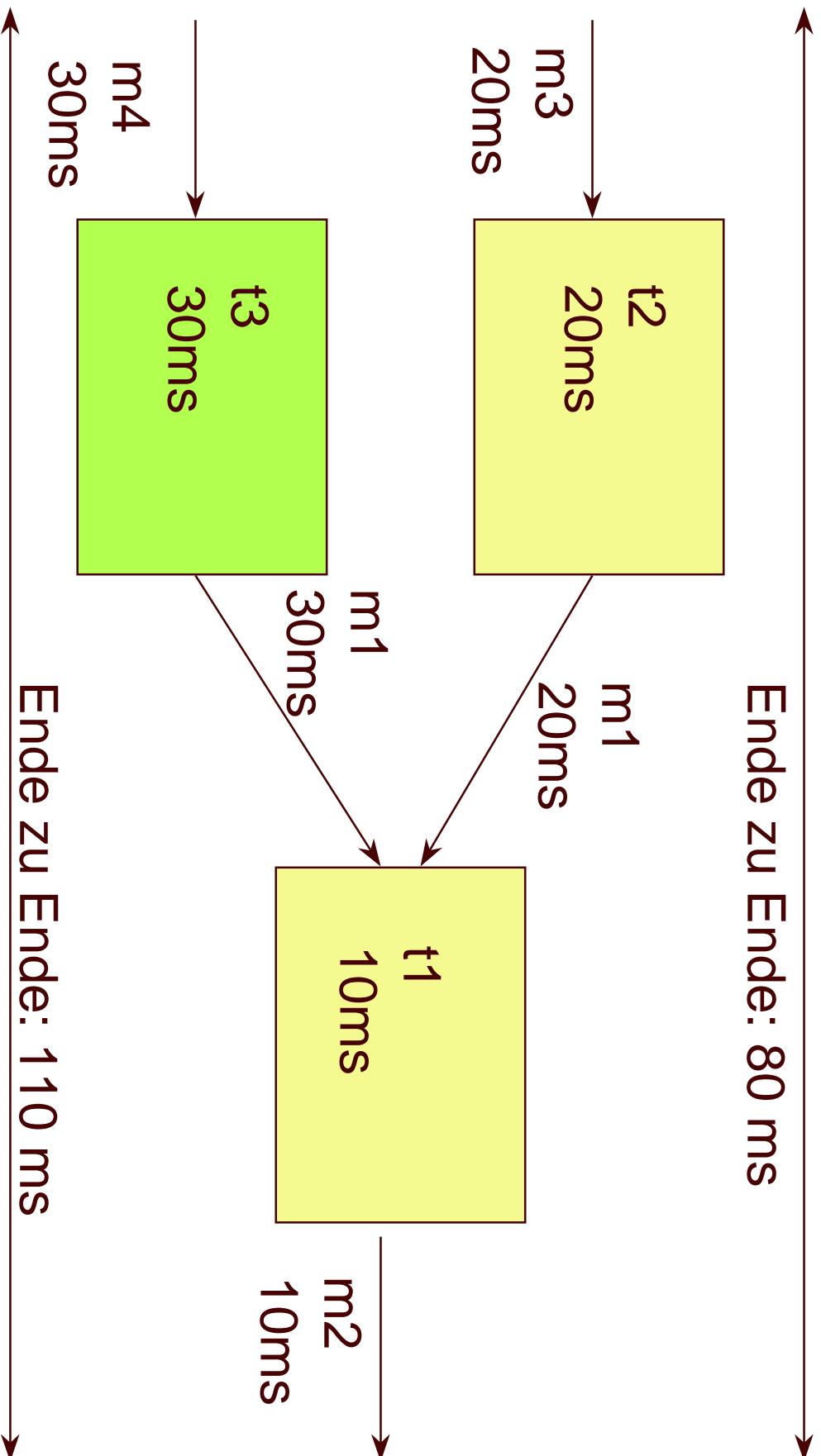
MSS bildet Entscheidungen der Komponierbarkeit auf Schedulingentscheidungen ab:

- **Lokales Scheduling aller Tasks auf einem Knoten**
 - betrachtet die lokalen Ressourcen eines Knoten wie CPU-Zyklen
- **Globales Scheduling auf dem Echtzeit-Netzwerk**
 - zwischen allen Nachrichten verschiedener Typen
 - betrachtet die Nachrichten-Slots auf dem Kommunikationsmedium
 - **aller Nachrichten gleichen Typs an den gleichen Empfänger**
 - betrachtet die Inanspruchnahme eines Empfängers, an den mehrere Sender Nachrichten senden

MSS — Komponierbarkeit II

- Existenz von Schedules auf allen Ebenen:
Alle Anforderungen der Komponenten erfüllt
 - Die Berechnung der Schedules ist in konstanter Zeit und unter Nutzung bereits existierendes Wissens (Parameter wie Last) möglich
 - Die Berechnungen setzen auf eingeführten Verfahren der Echtzeit-Forschung auf (RMA, EDF)
 - Bandbreite und Last als Abstraktion
- Systemeigenschaften können aus den Komponenteneigenschaften berechnet werden (beispielsweise End-zu-End-Zeiten)

MSS — Beispiel



Vergleich MSS vs. TTA

MSS - Vorteile

- einfache Erweiterbarkeit
- unanfälliger gegen Störungen auf dem Medium
- geringes globales Wissen

TTA - Vorteile

- jitterfreie Ausgaben möglich
- gut geeignet für statische Einsatzgebiete
- hohe Auslastung erzielbar

MSS - Nachteile

- Jitterfreiheit kaum erreichbar
- erreichbare Auslastung niedriger

TTA - Nachteile

- Erweiterbarkeit aufwendig
- anfällig gegen Störungen auf dem Medium
- viel globales Wissen