EMES: Eigenschaften mobiler und eingebetteter Systeme

Systemarchitekturen

Dr. Felix Salfner, Dr. Siegmar Sommer

Wintersemester 2010/2011

Motivation

- Eingebettete Systeme sind mehr als die Summe ihrer Teile
  - Hardware, Betriebssystem, Anwendungssoftware, Externe Hardware (Sensoren, Aktuatoren), Nutzerschnittstellen
  - Entwurf und Implementation haben hohe Komplexität
  - Nicht-funktionale Eigenschaften: Zeitverhalten, Ressourcenbedarf, Verlässlichkeit

- Typische Idee: Teile-und-herrsche
  - Zerlegung, Wiederverwendung, standardisierte Schnittstellen, standardisierte Komponenten
  - Aber: Funktionale Komposition meist orthogonal zu nicht-funktionalen Eigenschaften

Motivation

- Idee: Anwendung einer Systemarchitektur
  - überdeckt alle Schichten und alle “Komponenten”
  - Regeln für “Bauteile” eines Systems
  - Regeln zum Bau von Systemen
- 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"
  - Erreichbarkeitseigenschaft: Wird von mindestens einem Element der Architektur erfüllt
  - Sicherheitseigenschaft: Wird von allen Elementen der Architektur erfüllt

Anforderungen an das Architekturmodell

- Möglichkeit der Darstellung der gewünschen Eigenschaften
- Allgemeingültigkeit
  - Betrachtung nicht nur einer Eigenschaft
  - Betrachtung nicht nur bestimmter Arten von Möglichkeiten, Systeme zu bauen
  - Domänenübergreifend (nicht nur Bau von Computersystemen)
- Im Speziellen bei Computersystemen:
  - Komponentenorientierte Sicht
  - Aufhebung der Trennung in Hard- und Software
- Unterstützung von Konzepten der Komponierbarkeit

Komponentenorientierte Sicht

- Komponente oder Element: Element einer Menge von Bauteilen
  - Erlaubte Menge von Bauteilen ist durch Architektur definiert
  - Konzept der “atomaren” Komponente
- Komposition: Zusammenfügen von Komponenten
  - Architektur definiert Kompositionsoperator(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 Kompositionsoperator 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

Eigenschaften und Komposition

Komposition von Elementen kann Eigenschaften verändern:

- Invariante
  Eigenschaft bleibt unverändert und identifizierbar (also an eines der Elemente gebunden)
- Gebundene Eigenschaft
  Eigenschaft wird an einen neuen Wert oder Wertebereich gebunden, bleibt aber einem der Elemente zuordenbar
- Verschwindende Eigenschaft
  Eigenschaft kann im neuen System nicht mehr identifiziert werden
- Auftauchende Eigenschaft
  Eine neue Eigenschaft wird generiert
- Übertragene Eigenschaft
  Eigenschaft wird aus einer Komponenteneigenschaft zu einer Systemeigenschaft

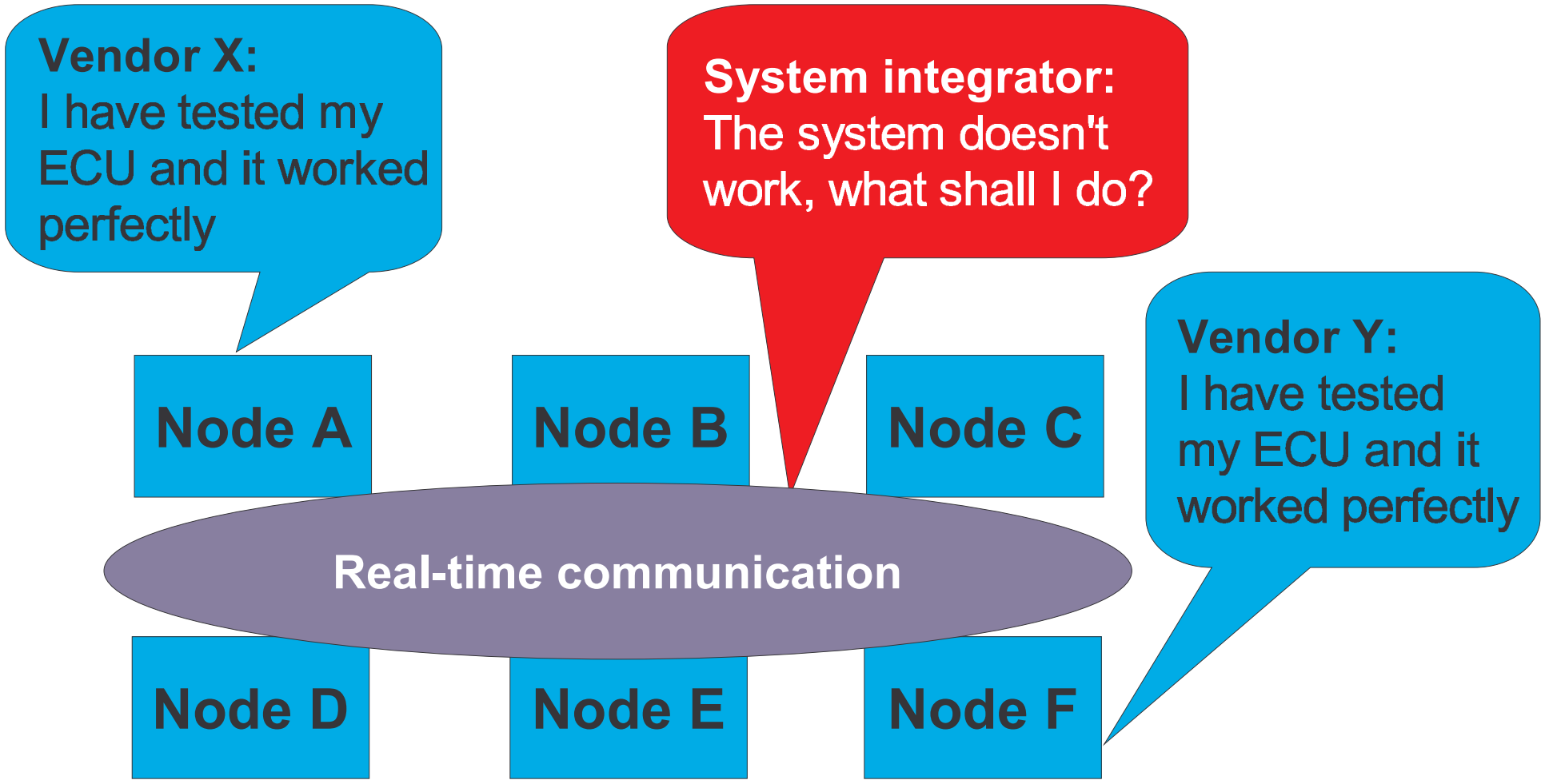
Komponierbarkeit

- Bergmans: Composability allows for the modular specification of modules with multiple independent concerns
- Kopetz: An architecture is said to be composable with respect to a specified property if the system intergration will not invalidate this property, once the property has been established at the subsystem level.
- Malek: A set of elements with given properties is composable if the calculation of composition's properties needs polynomial time.

Komponierbarkeit

- Kann auch Eigenschaft einer Architektur sein !
- 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



Komponierbarkeit

- 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)

- Kopetz et al., TU Wien
- Ziel: Komponierbare Architektur für verteilte Echtzeitsysteme in sicherheitskritischen Applikationen (x-by-wire)
- Adressierte Probleme:
  - Fehlertolerante Uhrensynchronisation
  - Echtzeit-Membership-Service
  - Management von Rekonfigurationen
  - Fail-silence in der Zeitdomäne — Vermeidung von "Babbling Idiot"
- Praxis:
  - Hardware verfügbar (eigene Controller, Integration in μController)
  - Praktisch im Einsatz (Luftfahrt, Automobilbau, Industrieanlagen, Antriebssysteme, Kabinendrucksteuerung des A380)

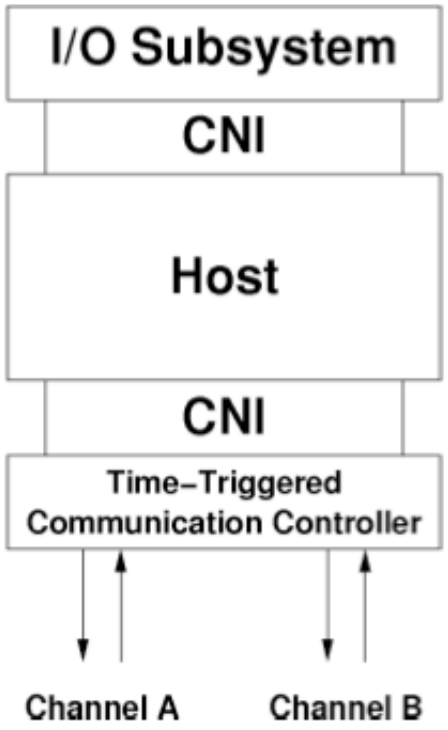
Architekturkonzept: Temporale Firewall

- Problem der temporalen Genauigkeit von Beobachtungen:
  'Die Ampel ist grün.'

- Temporale Firewall
  - Unidirektionale zustandsbehaftete Schnittstelle zu mehreren Seiten
  - Spezifikation von funktionalem und temporalen Verhalten
  - Wenigstens eine Seite greift unter einem vorher festgelegten Zeitverhalten schreibend bzw. lesend zu
  - Temporale Genauigkeit der abgelegten Daten in der Firewall ist vorher festgelegt
  - Wenn gespeicherte Daten ihre Gültigkeit verloren haben, wird Information verworfen
  - Grenze für Fehlerausbreitung

Architektur eines TTA-Knotens

- Knoten
  - Prozessor und Speicher, Betriebssystem, Anwendung
  - Communication Network Interface (CNI)
  - Communication Controller (CC)
  - I/O Subsystem
- TTA-Knoten werden zu einem Cluster durch TTP/C gekoppelt
  - Replizierter Broadcast-Kanal

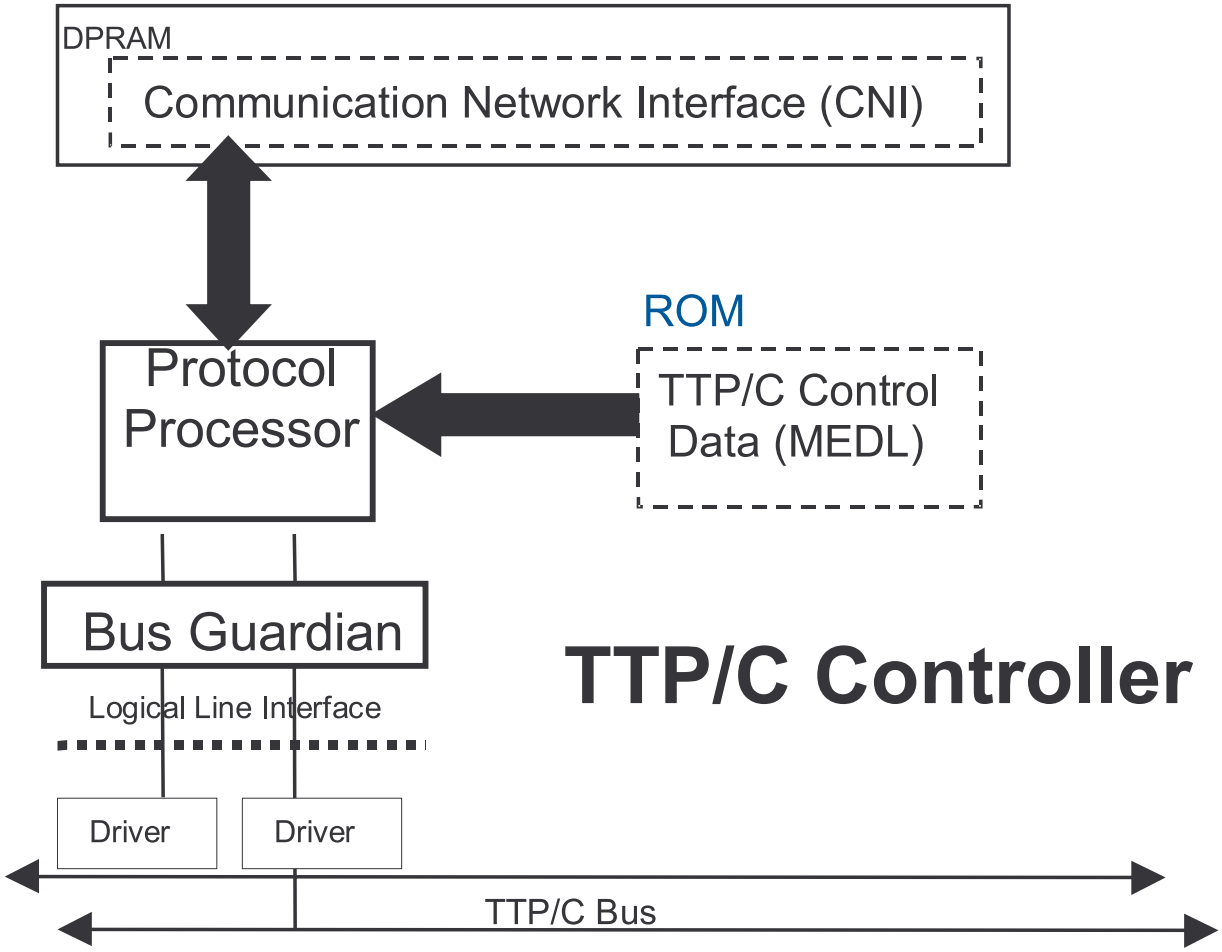


- CNI bietet unidirektionalen Sende- und Empfangskanal
  - Sender übergibt Daten an das CNI, Empfänger holt Daten ab

Architektur eines TTA-Knotens

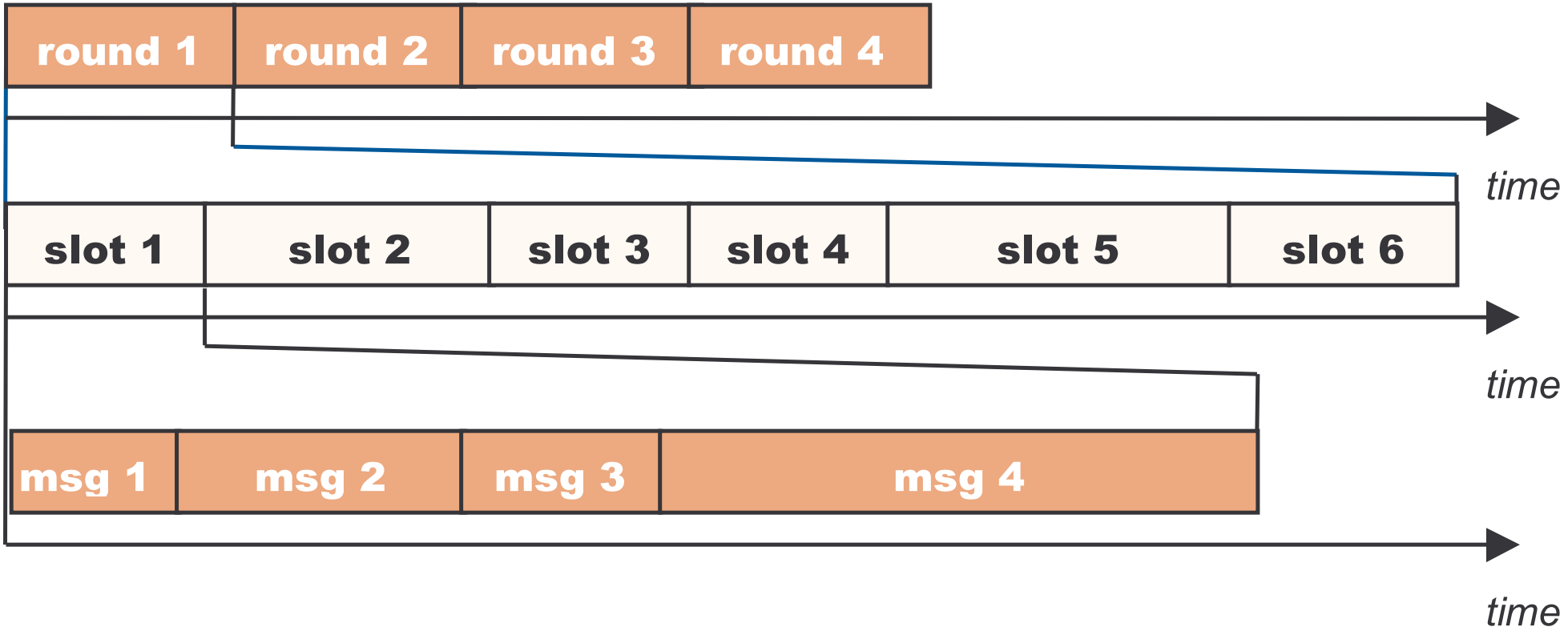
- Anwendungsprogramm und Kommunikationssystem sind durch die Architektur vollständig getrennt
- Kontrolle der Kommunikation unterliegt vollständig dem Kommunikationscontroller (CC)
  - Kommunikationssystem sendet zeitgesteuert Daten
  - Anwendungsprogramm hat keine “send”-Primitive
  - Zeitliche Fehler des Anwendungsprogrammes können sich nicht ausbreiten (error containment regions)
- Jede Komponente kann an der Grenze ihrer Schnittstelle in der Zeit- und Wertedomäne isoliert getestet werden

Architektur des CC in TTA



TTP/C

- Time-Triggered Architecture: Zeitverhalten 'a priori' bekannt
- Nutzung der Informationen durch TDMA - Ansatz
  - Verfügbare Zeit ist in Slots für Nachrichten eingeteilt
  - Slots sind Nachrichten fest und a priori zugeordnet
  - Zuordnung ist in MEDL (Message Descriptor List) gespeichert
  - 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


round 1
round 2
round 3
round 4
time
slot 1
slot 2
slot 3
slot 4
slot 5
slot 6
time
msg 1
msg 2
msg 3
msg 4
time

Echtzeit-Membership-Service

- TTP erfordert konsistente Sicht auf Gruppenmitgliedschaft
- Wissen über Zeitverhalten erlaubt Ausfallerkennung ohne Overhead, keine 'Alive'-Nachrichten nötig
- Ansatz: Senden von 'controller states'
  - Sender und Empfänger müssen sich über C-State einig sein - aktueller Mode, TDMA-Slot, globale Zeit, Membership-Status
  - Jeder Knoten besitzt Liste der als fehlerfrei angenommenen Knoten. Bei jeder Übertragung wird die Liste aktualisiert.
  - Es gibt verschiedene Memberschip-Protokolle (weiterentwickelt über die Jahre).

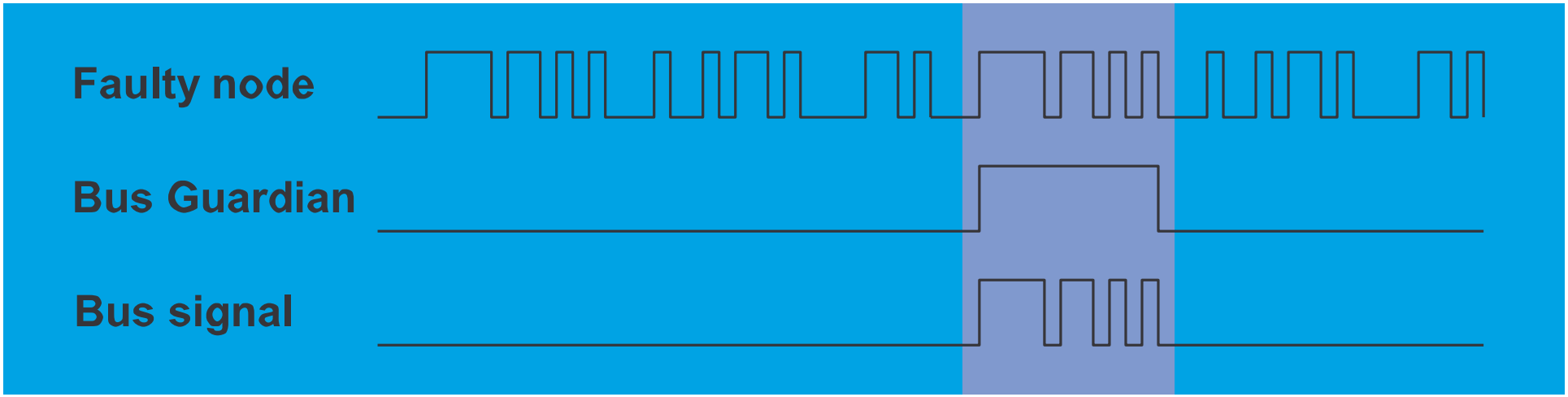
TTA-Membership II

- Im Wesentlichen funktioniert das Protokoll folgendermaßen:
  - Prüefsumme des C-State wird übertragen, bei Unterschied zu lokalem C-State: Annahme eines Fehlers bei sich selbst $\longrightarrow$ Passive Mode. Sendet Sender einen fehlerhaften Frame, wird er aus lokalem C-State ausgeschlossen.
  - Ankunft von korrekten Nachrichten führt zur Aufnahme/Erhalt des entsprechenden Knotens in lokalen C-State
  - Cliquen-Bildung (kommuniziert nur untereinander und hält sich gegenseitig für fehlerfrei) wird durch Voting behandelt

Vermeidung von “Babbling Idiot” I

- Problem: Fehlerhafter Knoten sendet ständig Nachrichten und stört damit die Kommunikation
  - Fehler im Anwendungsprozeß werden durch temporale Firewall aufgefangen
  - Bleibt: Behandlung von Fehlern des Kommunikationscontrollers
- Idee: Bus Guardian überwacht eletrischen 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” II



Top-Down-Entwurf in TTA

- Aufteilung des Systems in nahezu unabhängige vernetzte Komponenten
- Spezifikation der funktionalen Eigenschaften
- Spezifikation der Interaktionsmuster, inkl. zeitlicher Ablauf
  - Liefert implizit die Deadlines für die Echtzeitkomponente
- Definition der MEDL's der Kommunikationscontroller

Komponierbarkeit in TTA

- Unabhängige Entwicklung von Knoten aufgrund funktionaler und zeitlicher Spezifikationen
- CNI realisiert notwendige Schnittstelle
- Existierende Knoten werden durch Integration neuer Knoten nicht beeinflusst
- Aktive Redundanz und Voting für transparente Fehlertoleranz

Komponierbarkeit in TTA

- 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.
- Aspekte der Definition
  - Keine Unterscheidung zwischen System- und Komponenteneigenschaft
  - Komponierbarkeit ist einer Komponente zugeordnet
  - Blick auf Komponente unterstellt dem System ein der Komponente angepaßtes Verhalten
  - Komponierbarkeit beschränkt sich auf Erhaltung existierender Eigenschaften sowie deren “Übertragung” auf Systemebene

TTA – Weitere Aspekte

- Fehlertoleranz
  - Replizierter Bus
  - Separate Sendeteile
  - Separater Bus-Guardian
- Replikat-Determiniertheit
  - Gleiche MEDLs für beide Busse
  - Gleiche Zeitbasis für beide Busse
  - Exakter Parallelbetrieb
- Uhrensynchronisation
  - Vorherwissen über zeitliches Verhalten des Systems
  - Genaue Kenntnis, wann was geschehen muß
  - Leicht zu früh oder leicht zu spät eintreffende Ereignisse ermöglichen Anpassung der Uhren

Message Scheduled System (MSS)

- Entwickelt am Lehrstuhl für Rechnerorganisation und Kommunikation der HU Berlin, Dissertation von Dr. Jan Richling

- Ziele:
  - Architektur für eingebettete Echtzeitsysteme
  - Unterstützung von Komponierbarkeit in Bezug auf das zeitliche Verhalten
  - Generalisierbarkeit für andere Eigenschaften
  - „Nebenziele“:
    * Architekturbegriff
    * Komponierbarkeitsbegriff
    * Verifikationen
    * Simulationen

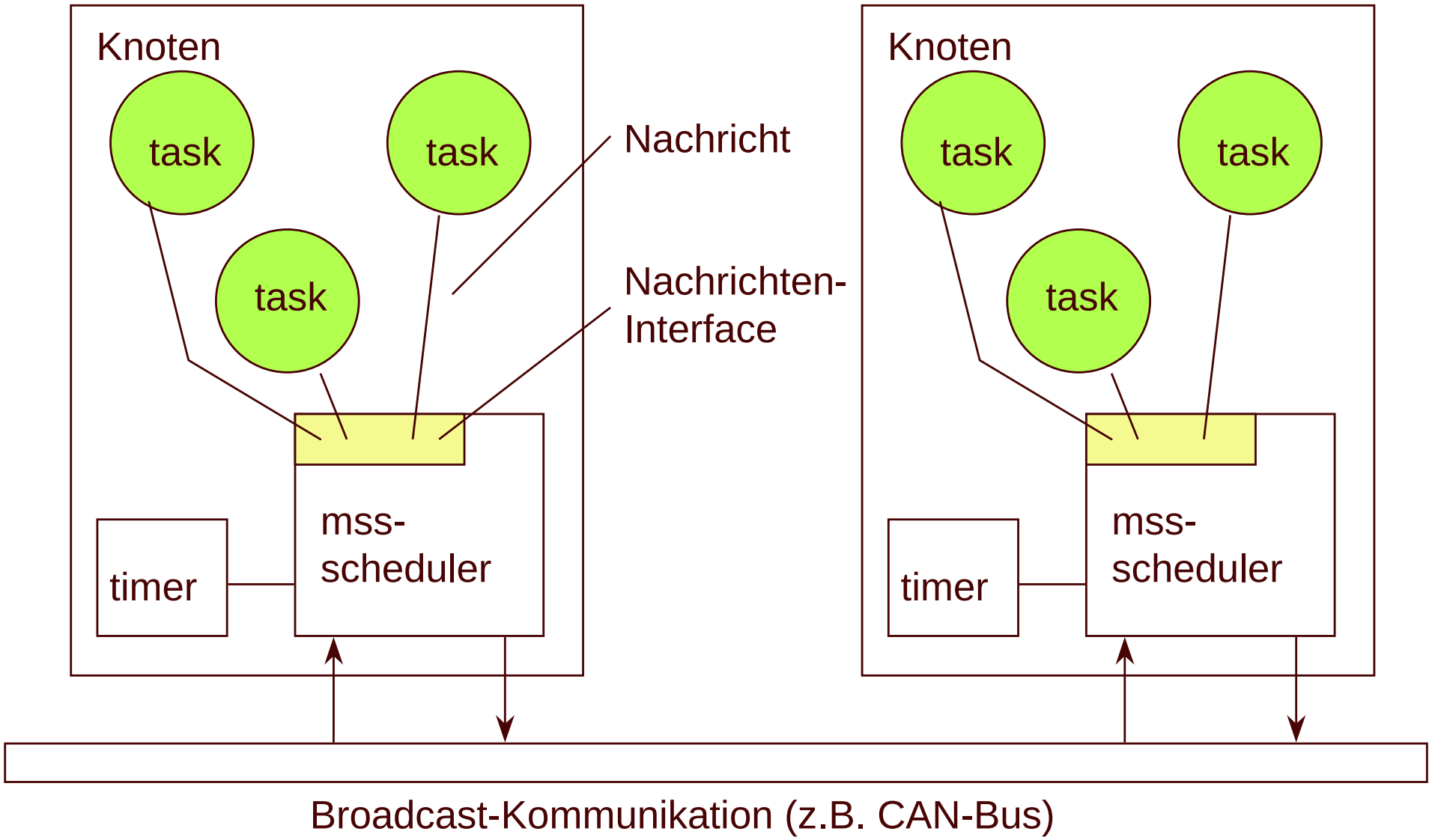
Message Scheduled System (MSS)

- Zielstellung:
  - Höhere Flexibilität durch Komponierbarkeit
- Zieldomäne:
  - Eingebettete Echtzeitsysteme (Autos, Flugzeuge, . . .)
  - Entwicklung einer Architektur, die komponierbar in Bezug auf das zeitliche Verhalten ist:
    - Einhaltung von Task-/Nachrichtendeadlines im komponierten System (invariante Eigenschaft)
    - Einhaltung von End-zu-End-Laufzeiten (auftauchende E.)
- Ideen:
  - Beschränkung des Design-Raumes
  - Globale Kontrakte auf Basis von lokalem Wissen
  - Mehrstufige Abbildung von Komponierbarkeitsentscheidungen auf Schedulingprobleme

MSS — Arten von Komponenten

- Tasks
  - Erzeugen aus einem Satz von perioden Eingangsereignissen einen Satz von Ausgangsnachrichten (mit Deadline)
  - Menge von ausgezeichneten Ereignissen löst Ausführung aus
- Knoten
  - Ausführungsumgebung für Tasks
  - Verwaltung des Nachrichtenverkehrs der lokalen Tasks
  - Scheduling des globalen Nachrichtenverkehrs
- Kommunikationsmedium
  - Echtzeitfähiger Bus
  - Verbindet die Knoten

MSS — Architektur



MSS — Komponenteneigenschaften

Beispiele, nicht vollständig

- Task:
  - WCET (Worst Case Execution Time)
  - Periode (bzw. Minimal Interarrival Time) — entspricht Deadline
  - Eingangsnachrichten mit Perioden und Typen, „Wake-Up-Message“
  - Ausgangsnachrichten mit Perioden und Typen
- Knoten:
  - Zusammengefaßte Parameter der Tasks
- System:
  - Zusammengefaßte Parameter der Knoten

MSS — Komponierbarkeit

Idee: Mehrstufige Abbildung von Komponierbarkeitsentscheidungen auf Schedulingentscheidungen

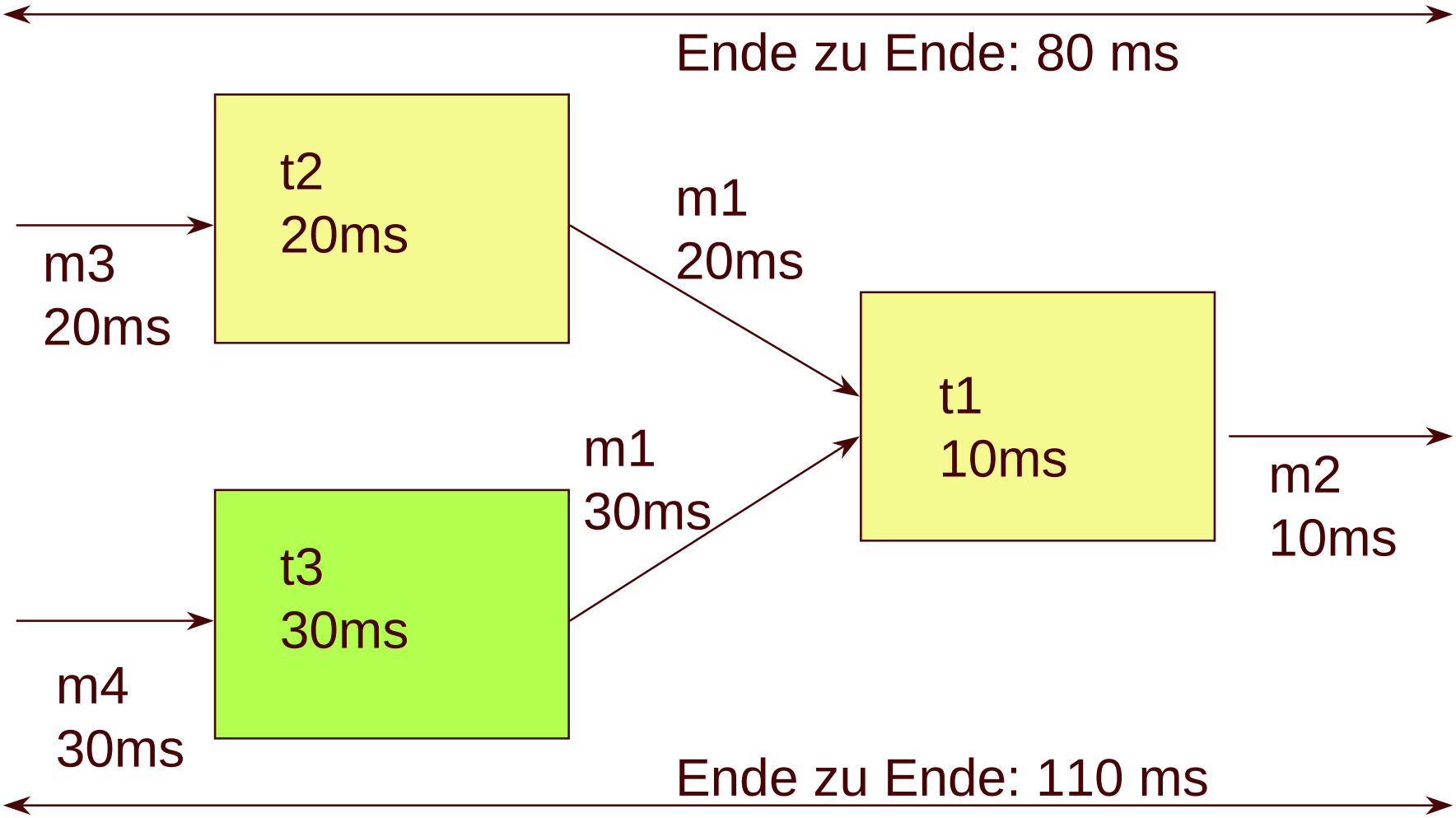
- Lokales Scheduling aller Tasks auf einem Knoten
  - Ressourcen: CPU-Zeit, Speicher, . . .
- Globales Scheduling auf dem echtzeitfähigen Bus
  - Zwischen Nachrichten verschiedener Typen
    - Ressource: Nachrichtenslots auf dem Medium
  - Nachrichten gleichen Typs an den gleichen Empfänger
    - Ressource: Fähigkeit des Empfängers, eine ankommende Nachricht zu verarbeiten

MSS — Komposition

- Existenz von Schedules auf allen Ebenen: Anforderungen der Komponenten sind erfüllt
  - Berechnung ist mit geringem Aufwand durchführbar
  - Bereits berechnetes Wissen (Last) kann benutzt werden
  - Berechnung setzt auf bekannten Verfahren auf (RMS)
  - Systemeigenschaften können aus den Komponenteneigenschaften berechnet werden

- Ablauf einer Komposition:
  - Neue Komponente stellt Anfrage mit Garantien und Bedingungen
  - Beliebiger MSS-Scheduler kann antworten (positiv/negativ)
  - Wenn positiv, dann kann wird Komponente Teil des Systems und der Kontrakt wird gültig

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 erzielbatr

MSS - Nachteile

- Jitterfreiheit kaum erreichbar
- erreichbare Auslastung niedriger

TTA - Nachteile

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