Komponierbarkeit eingebetteter Echtzeitsysteme

Jan Richling
Institut für Informatik
Humboldt-Universität zu Berlin
richling@informatik.hu-berlin.de

Gliederung

- Motivation / Zielstellung
- Verwandte Arbeiten
- Komponierbarkeit
- Message Scheduled System
- Zusammenfassung
- Ausblick

Einleitung / Motivation

- Schwerpunkt: verteilte Systeme, Middleware, eingebettete System
- Problem: Interoperabilität “horizontal” und “vertikal”
- Problem: Hohe Komplexität verteilter Systeme
  - Zerlegung
  - Wiederverwendung
  - Standardschnittstellen
  - Standard-“Teile“ (Komponenten)
- Fokus: Echtzeitsysteme

Einleitung / Motivation

- Modularisierung und Strukturierung: betrachten *funktionale* Aspekte
- Nicht ausreichend bei Echtzeitsystemen
- *Nichtfunktionale* Aspekte sind bedeutsam: zeitliches Verhalten, Ressourcenbedarf
- Gehen nicht aus dem Code der Komponente hervor, sondern sind von der Ausführungsumgebung und Interaktionen mit anderen Komponenten abhängig
- Ziel: “Komponierbarkeit”

Verwandte Arbeiten

- Komponentensysteme wie CORBA oder DCOM
  - RT-CORBA
  - Minimum CORBA
- Time Triggered Message Triggered Object (TMO)
  - netzwerktransparente RT-Objekte
  - Erweiterung existierender Objekt-Strukturen, Middlewareansatz
- Time Triggered Architecture (TTA)
  - Wartbarkei , Verifizierbarkeit, Testbarkeit als Ziele
  - globales Vorauswissen über zeitliche Abläufe im System

Komponierbarkeit - Definition

- A system architecture is called composable with respect to a set of properties iff system properties can be inferred from component properties and these component properties must be preserved.
- Aspekte:
  - usefully composable: system properties are more than the union of component properties
  - efficiently composable: system properties can be calculated in polynomial time

Eigenschaften

- funktionale Aspekte
  - Kompatibilität der Schnittstellen (Parameter, Datentypen)
  - Semantik
- nichtfunktionale Aspekte
  - zeitliches Verhalten
  - Sicherheit
  - Fehlertoleranz
  - Ressourcenbedarf
- Fokus:
  "Komponierbarkeit in Bezug auf eine Klasse von nichtfunktionalen Eigenschaften"

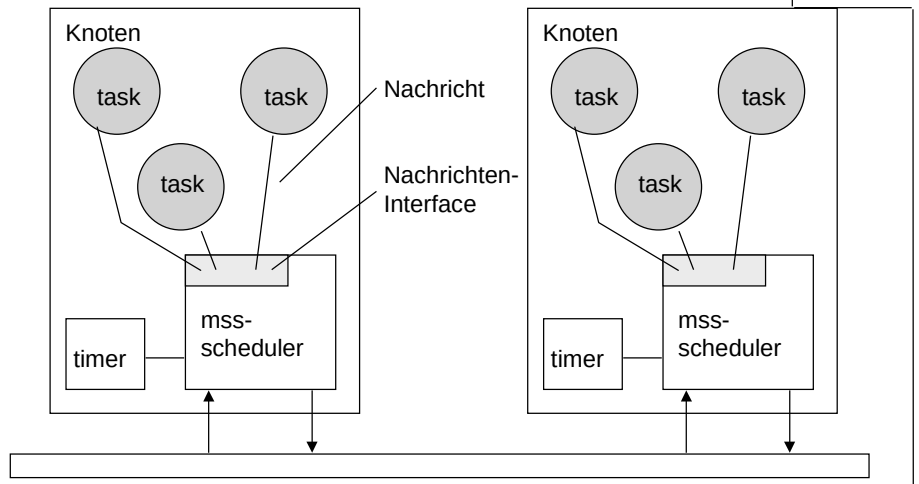
Message Scheduled System (MSS)

- 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

Message Scheduled System (MSS)

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

MSS: Architektur



MSS: Komponierbarkeit

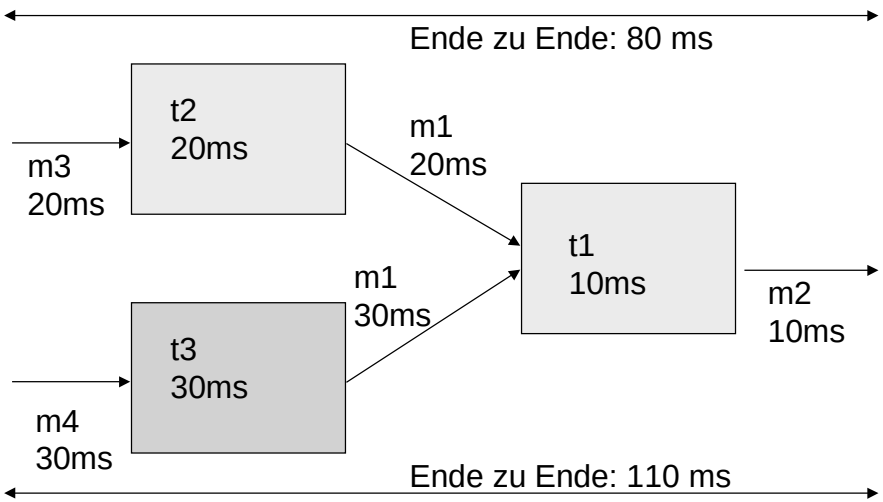
MSS bildet Entscheidungen der Komponierbarkeit auf Schedulingentscheidungen ab:

- 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

- Existenz von Schedules auf allen Ebenen:
  Alle Anforderungen der Komponenten erfüllt
  - Die Berechnung der Schedules ist in konstanter Zeit und unter Nutzung bereits existierenden 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 und TT

MSS - Vorteile

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

TTA - Vorteile

- jitterfreie Ausgaben möglich
- gut geeignet für statische Einsatzgebiete

MSS - Nachteile

- Jitterfreiheit kaum erreichbar

TTA - Nachteile

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

Zusammenfassung

- Komponierbarkeit in Bezug auf eine Klasse nichtfunktionaler Eigenschaften
- MSS: komponierbare Architektur für eingebettete Echtzeitsysteme
- MSS: Abbildung von Komponierbarkeitsent-scheidungen auf Schedulingtests

Ausblick

- Endgültige Definition von “Komponierbarkeit” (Zusammenarbeit mit DaimlerChrysler)
- Formale Spezifikation von MSS
- Fehlertoleranz in MSS
- Entwicklung einer Komponentenbeschreibungssprache
- Implementation eines Simulators
- Implementation eines Prototypen auf Basis von CAN und PURE (Zusammenarbeit mit Universität Magdeburg und FIRST)