Schedulability Analysis and Optimization for the Synthesis of Multi-Cluster Distributed Embedded Systems

Paul Pop, Petru Eles, Zebo Peng
Embedded Systems Lab (ESLAB)
Linköping University, Sweden
Heterogeneous Networks

Multi-Cluster Systems

NoCs

Factory Systems

Automotive Electronics
Applications distributed over heterogeneous networks are difficult to...

- Analyze (e.g., guaranteeing timing constraints)
- Design (e.g., efficient implementation)

- Applications distributed over the heterogeneous networks
  - Reduce costs: use resources efficiently
  - Requirements: close to sensors/actuators

Unsolved problems
Contributions

- **Analysis and design** of Multi-Cluster Embedded Systems
  - **Analysis**
    - Proposed a schedulability analysis for safety-critical **hard real-time** applications mapped on multi-cluster distributed embedded systems
      - Is the application schedulable? (Are deadlines satisfied?)
      - Bounds on the communication delays and communication buffer sizes
  
- **Design optimization**
  - In this paper we have addressed communication synthesis and priority assignment for
    - Improving the degree of schedulability of an application
    - Minimizing communication buffer sizes needed to run a schedulable application
Outline

- Motivation
- Contributions
  - System architecture and application model
    - Schedulability analysis for multi-clusters
    - Optimization strategies
    - Experimental results
    - Message and future work
Time Triggered Protocol (TTP)
- Bus access scheme: time-division multiple-access (TDMA)
- Schedule table located in each TTP controller: message descriptor list (MEDL)

Controller Area Network (CAN)
- Priority bus, collision avoidance
- Highest priority message wins the contention
- Priorities encoded in the frame identifier

Hardware Architecture

- **Time-triggered** cluster
  - Static cyclic scheduling
  - Time-triggered protocol

- **Event-triggered** cluster
  - Fixed priority preemptive scheduling
  - Controller area network protocol

Gateway
Software Architecture

Time-triggered cluster

Event-triggered cluster

Gateway
Problem Formulation

- **Input**
  - An application modeled as a set of process graphs
  - Each process has an worst case execution time, a period, and a deadline
  - Each message has a known size
  - The system architecture and the mapping of the application are given

- **Output**
  - Worst case response times and bounds on the buffer sizes
  - Design implementation such that the application is schedulable and buffer sizes are minimized
    - Schedule table for TT processes
    - Priorities for ET processes
    - Schedule table for TT messages
    - Priorities for ET messages
    - TT bus configuration (TDMA slot sequence and sizes)
Schedulability Analysis

- Scheduling **time-triggered** activities:
  - Building a schedule table:
    static cyclic scheduling (e.g., list scheduling)

- Scheduling **event-triggered** activities:
  - Response time analysis:
    calculate worst case response times for each process
  - Schedulability test: response times smaller than the deadlines
  - Response times depend on the communication delay between sending and receiving a message
  - Communication delays depend on the type of message passing
    1. TTC → TTC
    2. TTC → ETC
    3. ETC → ETC
    4. ETC → TTC
      - Communication delays
      - Bounds on the buffer sizes
Scheduling cannot be addressed separately for each type of cluster

The inter-cluster communication creates a **circular dependency**:  
- TTC static schedules (offsets) ⇒ ETC response times  
- ETC response times ⇒ TTC schedule table construction

**Offset**: earliest possible start time for an *event-triggered* activity
Optimization Example

Transformation: $P_2$ is the high priority process on $N_2$
Optimization Strategies

- **OptimizeSchedule**
  - Synthesizes the communication and assigns priorities to obtain a schedulable application
  - Based on a greedy approach
    - Cost function: degree of schedulability

- **OptimizeBuffers**
  - Synthesizes the communication and assigns priorities to reduce the total buffer size
  - Based on a hill-climbing heuristic
    - Cost function: total buffer size

- **Straightforward solution**
  - Finds a schedulable application
  - Does not consider the optimization of the design
Can We Improve Schedulability?

Cost function: degree of schedulability

- **Straightforward solution**
  - Does not perform optimizations

- **OptimizeSchedule**
  - Near-optimal values for the degree of schedulability

- **Simulated Annealing**

![Graph showing the relationship between number of processes and average percentage deviation.](image)
Can We Reduce Buffer Sizes?

Cost function: total buffer size

Number of processes vs. Average total buffer size [k]

- OptimizeBuffers
  - Case study
    - Vehicle cruise controller
    - Distributed over the TT and ET clusters

- OptimizeSchedule
  - Does not optimize the total buffer size
Analysis and optimization methods are needed for the efficient implementation of applications distributed over interconnected heterogeneous networks.

- Future Work
  - Explore more design problems
    - Mapping for multi-clusters
    - How to partition an application in ET and TT activities?