Analysis and Optimization of Distributed Real-Time Embedded Systems

Paul Pop, Traian Pop, Sorin Manolache, Petru Eles, Zebo Peng

Department of Computer and Information Science (IDA)
Linköping University
http://www.ida.liu.se/~eslab/
Outline

- Distributed Embedded Systems
- System-level Design Flow
- Application Model
- Distributed Hard Real-Time Systems
  - Heterogeneous Distributed Systems
  - Static and Dynamic Communication
  - Analysis & Optimization
  - Fault Tolerance
- Distributed Soft Real-Time Systems
  - Analysis Approaches and Optimization
  - Optimization
Distributed Embedded Systems
Distributed Embedded Systems

I/O Interface

CPU

RAM
ROM
ASIC

Comm. Controller

Sensors & Actuators
Distributed Embedded Systems

Why?

- Physical constraints
  - Operation close to sensor;
- Modularity constraints
- Safety Constraints
- Performance
System Level Design Flow

System model

Mapping
Scheduling
Bus access

Mapped and scheduled model

OK

Analysis

Software Synthesis Hardware Synthesis

Exec. time estimation

System platform
System Level Design Flow

- System platform
- Exec. time estimation
- System model
- Mapping
- Scheduling
- Bus access
- Mapped and scheduled model
- Analysis
- Software Synthesis
- Hardware Synthesis

Design space exploration loop

Analyze and optimize distributed real-time embedded systems.

Petru Eles
System Level Design Flow

- System platform
- Exec. time estimation
- System model
- Mapping
- Scheduling
- Bus access
- Mapped and scheduled model
- Design space exploration loop
- Analysis
- Software Synthesis
- Hardware Synthesis

Driven by
Category of applications:

- Hard RT: WCET, hard deadlines to be satisfied
- Soft RT: exec. time probability distributions, perc. of missed deadlines
**Category of applications:**
- Hard RT: WCET, hard deadlines to be satisfied
- Soft RT: exec. time probability distributions, perc. of missed deadlines

**Constraints:**
- Time constraints (hard, soft)
- Cost (amount of resources)
- Fault tolerance
System Level Design Flow

- **Category of applications:**
  - Hard RT: WCET, hard deadlines to be satisfied
  - Soft RT: exec. time probability distributions, perc. of missed deadlines

- **Constraints:**
  - time constraints (hard, soft)
  - cost (amount of resources)
  - fault tolerance

- **Optimization problems:**
  - schedule generation/priority assignment
  - task mapping
  - bus configuration
  - scheduling policy assignment
  - fault tolerance policy assignment
An application is modelled as a set of task graphs:

\[ \Gamma_1 \]
\[ \text{Period: } T_{\Gamma_1} \]
\[ \text{Deadline: } D_{\Gamma_1} \]

\[ \Gamma_2 \]
\[ \text{Period: } T_{\Gamma_2} \]
\[ \text{Deadline: } D_{\Gamma_2} \]

\[ \Gamma_3 \]
\[ \text{Period: } T_{\Gamma_3} \]
\[ \text{Deadline: } D_{\Gamma_3} \]
An application is modelled as a set of task graphs:
An application is modelled as a set of task graphs:

\[ \tau_0 \]

\[ \tau_1 \]

\[ \tau_2 \]

\[ \tau_3 \]

\[ \tau_4 \]

\[ \tau_5 \]

\[ \tau_6 \]

\[ \tau_7 \]

\[ \tau_8 \]

\[ \tau_9 \]

\[ \tau_{10} \]

\[ \tau_{11} \]

\[ \tau_{12} \]

\[ \tau_{13} \]

\[ \tau_{14} \]

\[ \tau_{15} \]

\[ \tau_{16} \]

\[ \tau_{17} \]

\[ \tau_{32} \]

\[ C \tau_3 \] or \[ \varepsilon \tau_3 \]

\[ \delta \tau_3 \]

\[ C: WCET \]

\[ \varepsilon : ETPDF \]

\( \Gamma_1 \) Period: \( T_{\Gamma_1} \)
Deadline: \( D_{\Gamma_1} \)

\( \Gamma_2 \) Period: \( T_{\Gamma_2} \)
Deadline: \( D_{\Gamma_2} \)

\( \Gamma_3 \) Period: \( T_{\Gamma_3} \)
Deadline: \( D_{\Gamma_3} \)
Outline

- Distributed Embedded Systems
- System-level Design Flow
- Application Model
- Distributed Hard Real-Time Systems
  - Heterogeneous Distributed Systems
  - Static and Dynamic Communication
  - Analysis & Optimization
  - Fault Tolerance
- Distributed Soft Real-Time Systems
  - Analysis Approaches and Optimization
  - Optimization
Distributed Hard Real-Time Systems

- Given:
  - WCET and BCET of each task
  - Dimension of messages
  - Deadlines for tasks/task graphs

☞ All deadlines have to be satisfied!
Distributed Hard Real-Time Systems

- Given:
  - WCET and BCET of each task
  - Dimension of messages
  - Deadlines for tasks/task graphs

☞ All deadlines have to be satisfied!

- Particular issues taken into consideration
  - Heterogeneous nature of the system
  - Various communication protocols

☞ Try to satisfy constraints at low cost!
Application Area

- Automatic Cruise Control
- Anti-lock Breaking System
- Engine Control
- Powertrain Control
- Electronic Stability Program
- Acceleration Skid Control
Heterogeneous Distributed Embedded Systems

■ RT Design Approach

■ Network Protocol
Heterogeneous Distributed Embedded Systems

- RT Design Approach
  - Time triggered
  - Event triggered

- Network Protocol
Heterogeneous Distributed Embedded Systems

- RT Design Approach
  - Time triggered
  - Event triggered

- Network Protocol
  - Static
  - Dynamic
Heterogeneous Distributed Embedded Systems

☞ Heterogeneous Nature of Implemented Functions

[Diagram showing a heterogeneous system with various components and connections]
Heterogeneous Distributed Embedded Systems

Engine Control
- hard real-time
Heterogeneous Distributed Embedded Systems

- Engine Control
  - hard real-time

- Power Train (break-by-wire, ABS)
  - hard real-time
  - highly safety-critical
Heterogeneous Distributed Embedded Systems

- Engine Control
  - hard real-time

- Power Train (break-by-wire, ABS)
  - hard real-time
  - highly safety-critical

- Air Conditioning
  - soft real-time
Static Communication: TTP

I/O Interface

CPU
- RAM
- ROM
- ASIC

TTP Controller

Sensors & Actuators

N₀ writes N₁ writes

left empty in this round

TTP cycle

TTP round 1
- S₀
- S₁
- \( \cdots \)\n- Sₙ

TTP round 2
- S₀
- S₁
- \( \cdots \)\n- Sₙ
Static Communication: TTP

Over a TTP cycle

<table>
<thead>
<tr>
<th>Time</th>
<th>Action</th>
<th>length (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>rd-frame</td>
<td>16</td>
</tr>
<tr>
<td>$t_2$</td>
<td>wr-frame</td>
<td>32</td>
</tr>
<tr>
<td>$t_3$</td>
<td>rd-frame</td>
<td>32</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$t_x$</td>
<td>wr-frame</td>
<td>16</td>
</tr>
</tbody>
</table>
Dynamic Communication: CAN

- Priority bus with collision avoidance mechanism:
  - The node that transmits the highest priority frame wins the contention.
Mixed Communication: UCM

- Universal Communication Model
  - Static phase: TDMA
  - Dynamic phase: CAN

Bus cycle
Mixed Communication: UCM

Universal Communication Model

- Static phase: TDMA
- Dynamic phase: CAN

Bus cycle

\[ N_0 \quad N_1 \quad N_n \]
Mixed Communication: FlexRay

- Static phase: TDMA
- Dynamic phase: Flexible TDMA

Bus cycle

Static phase
Dynamic phase
The Traditional Approach

One node ↔ One function

[Diagram showing a network of nodes with one function each]
The Traditional Approach

One node ↔ One function

- Purchase node & function and integrate into system
- Sophistication grew quickly

More than 100 nodes

Resources have to be used more efficiently
What Comes

One node ↔ Several functions

Several nodes ↔ One function

Flexibility!

- Reduce cost
- Improve resource usage
- Function close to sensor
What Comes

One node ↔ Several functions
Several nodes ↔ One function

Flexibility!
- Reduce cost
- Improve resource usage
- Function close to sensor

Needed:
- Middleware software
- New analysis
- New system optimization
Navet et al.
*Trends in Automotive Communication Systems*
The different domains are isolated.

Time triggered cluster:
- TT tasks
- Static communication

Event triggered cluster:
- ET tasks
- Dynamic communication
The TT & ET, Static & Dynamic domains are interacting because:

- They share resources (node, bus)

and/or

- They communicate
Multi-cluster Heterogeneous Distributed Architecture

Analyse this!
Single-cluster Heterogeneous Distributed Architecture

N₀  N₁  Nₙ

⋯
Single-cluster Heterogeneous Distributed Architecture

Hierarchical Scheduler

N₀

TT

ET f.p. edf

N₁

TT

ET f.p. edf

Nₙ

TT

ET f.p. edf

...
Single-cluster Heterogeneous Distributed Architecture

Hierarchical Scheduler

- UCM (TTP&CAN)
- FlexRay
Isolated Domains

ET tasks
DYN messages

Schedulability Analysis

Response Times

Static Cyclic Scheduling

Activity | Time
--- | ---

TT tasks
ST messages
Resource sharing, no communication

ET tasks
DYN messages

TT tasks
ST messages

Schedulability Analysis
Response Times

Static Cyclic Scheduling

Slacks

Activity Time

Resource sharing, no communication
Analysis

Resource sharing and communication

ET tasks
DYN messages

TT tasks
ST messages

Schedulability Analysis
Fixpoint
Slacks/Offsets
Constraints
Static Cyclic Scheduling

Response Times

Activity | Time
---------|-------

Analysis and Optimization of Distributed Real-Time Embedded Systems
Petru Eles
Leiden, 2005
Analysis

Static Cyclic Scheduling

Schedulability Analysis

Constraints

Fixpoint

Slacks/Offsets

Response Times

Find a schedule as “friendly” as possible towards the ET tasks!

ET tasks
DYN messages

TT tasks
ST messages

Analysis and Optimization of Distributed Real-Time Embedded Systems
Petru Eles
Leiden, 2005
Once the analysis approach is in place, several new, specific optimization tasks can be performed (in addition to “classical” ones, like task mapping):

- Task partitioning into TT, ET
- Cluster mapping
- Bus access optimization (static, dynamic phases)
- Buffer minimisation
Optimization

System model
- Exec. time estimation
- Mapping and scheduling
  - Bus access
  - Mapped and scheduled model

Analysis
- OK

Software synthesis
Hardware synthesis

In addition:
- Partition into TT and ET (f.p., edf)
- Buffer space

System platform
- OK
Mapping & Task Partitioning

\[ \tau_1 \rightarrow \tau_2 \rightarrow \tau_3 \rightarrow \tau_4 \rightarrow \tau_5 \rightarrow \tau_6 \rightarrow \tau_7 \]
\[ \tau_8 \rightarrow \tau_9 \rightarrow \tau_{10} \rightarrow \tau_{11} \rightarrow \tau_{12} \rightarrow \tau_{13} \]
\[ \tau_{14} \rightarrow \tau_{15} \rightarrow \tau_{16} \rightarrow \tau_{17} \]

\[ N_0 \quad N_1 \quad \ldots \quad N_n \]

ET tasks

TT tasks
Mapping & Task Partitioning
Outline

- Distributed Embedded Systems
- System-level Design Flow
- Application Model
- Distributed Hard Real-Time Systems
  - Heterogeneous Distributed Systems
  - Static and Dynamic Communication
  - Analysis & Optimization
  - Fault Tolerance
- Distributed Soft Real-Time Systems
  - Analysis Approaches and Optimization
  - Optimization
Another Issue: Fault Tolerance

- Transient faults
  - Their number can be much larger than that of permanent faults.

- Find cost-effective implementations that are fault tolerant and satisfy time constraints.

Some Interesting trade-offs!
Decide which fault tolerance technique to apply:

- re-execution
- re-exution&checkpointing
- replication

Different techniques can be applied to different tasks
Fault Tolerance

- Decide which fault tolerance technique to apply:
  - re-execution
  - re-execution & checkpointing
  - replication

- Map the tasks (including eventual replicas)
Fault Tolerance

- Decide which fault tolerance technique to apply:
  - re-execution
  - re-exution&checkpointing
  - replication
- Map the tasks (including eventual replicas)
- Decide on transparency

**Transparent:** The schedule of outgoing messages does not depend on occurrence of faults (faults are not visible to the outside).
Fault Tolerance

- Decide which fault tolerance technique to apply:
  - re-execution
  - re-execution & checkpointing
  - replication

- Map the tasks (including eventual replicas)

- Decide on transparency

- Do the analysis/scheduling, considering worst case number of faults (re-executions).

  Are time constraints satisfied? If not, go back!
Fault Tolerance

- Decide which fault tolerance technique to apply:
  - re-execution
  - re-execution & checkpointing
  - replication

- Map the tasks (including eventual replicas)

- Decide on transparency

- Do the analysis/scheduling, considering worst case number of faults (re-executions).
  Are time constraints satisfied? If not, go back!

- Which is the optimal number of check-points?
Outline

- Distributed Embedded Systems
- System-level Design Flow
- Application Model
- Distributed Hard Real-Time Systems
  - Heterogeneous Distributed Systems
  - Static and Dynamic Communication
  - Analysis & Optimization
  - Fault Tolerance
- Distributed Soft Real-Time Systems
  - Analysis Approaches and Optimization
  - Optimization
Distributed Soft Real-Time Systems

Given:

- WCET and BCET of each task
- Dimension of messages
- Deadlines for tasks/task graphs

All deadlines have to be satisfied!
Distributed Soft Real-Time Systems

Given:

- Execution Time Probability Distribution Function (ETPDF) of each task
- Dimension of messages
- Deadlines for tasks/task graphs

A certain percentage of deadlines have to be satisfied!
Distributed Soft Real-Time Systems

■ Given:
  ■ Execution Time Probability Distribution Function (ETPDF) of each task
  ■ Dimension of messages
  ■ Deadlines for tasks/task graphs

☞ A certain percentage of deadlines have to be satisfied!

■ Problems:
  ■ Determine the percentage of missed deadlines for each task/task graph.
  ■ Optimize task mapping and priority assignment, such that a required percentage of satisfied deadlines is achieved.
Input:

- Set of mapped task graphs, periods, deadlines, priorities.
Analysis

Input:

- Set of mapped task graphs, periods, deadlines, priorities.
- Set of execution times probability density functions.
- Scheduling policy.
Output:

- Ratio of missed deadlines per task/task graph
The main problem is **Complexity** (in terms of analysis time and memory).

- Applying straight-forward solutions is possible
  - only for very(!) small applications.
  - and/or
  - for very particular cases (e.g. exponential distributions).
Approaches

- An exact method for schedulability analysis, efficiently applicable (but not limited) to monoprocessor systems.

- An method for schedulability analysis, trading analysis efficiency for result accuracy; efficiently applicable to multiprocessors.

- A very fast, approximate analysis, to be used inside an optimization loop.
Stochastic Process Construction

- Analyse of the underlying stochastic process.
  - A state of the process should capture enough information to be able to generate the next states and to compute the corresponding transition probabilities.
Stochastic Process Construction

Task A: period=3

Task B: period=5

EDF scheduling
Stochastic Process Construction

Task A: period=3

Task B: period=5

A, 0, \{B\}

B, t_0, \{\}

B, t_1, \{\}

B, t_k, \{A\}

B, t_{k+1}, \{A\}
Stochastic Process Construction

Task A: period=3

Task B: period=5

Infinite number of states!
Stochastic Process Construction

Task A: period=3

Task B: period=5

Infinite number of states!
Stochastic Process Construction

Task A: period=3

Task B: period=5

States that have their start time inside the same PMI are grouped together.
The Stochastic process is constructed and probabilities corresponding to individual states are calculated by convolution.
The Stochastic process is constructed and probabilities corresponding to individual states are calculated by convolution.

**Good News:** Process construction and analysis are performed simultaneously. Only a (very small) sliding window of the states has to be stored in memory at any time.
Limitation

- The number of states increases very much in the case of *interacting tasks* implemented on multiprocessor systems.

- We need to avoid
  - storing explicitly the distribution of residual exec. time in each state;
  - calculation of the convolutions.
The generalised ETPDFs are approximated by weighted sums of exponential functions (Coxian approximation).
The generalised ETPDFs are approximated by weighted sums of exponential functions (Coxian approximation).

The initial generalised semi-Markov process is transformed into a continuous time Markov chain.

Larger, but much easier to analyse.
Multiprocessor Approach

The generalised ETPDFs are approximated by weighted sums of exponential functions (Coxian approximation).

- **Good News:** The whole matrix, corresponding to the complete model, has *not* to be stored in memory; it is generated on the fly, from terms expressed as small matrices.

- **Accuracy can be traded for analysis time/memory.**
Multiprocessor Approach

- Task graphs
- Stochastic Petri Net Model
- Reachability Graph/GSMP
- CTMC
- Analysis
- Results (percentage of missed deadlines)
- Coxian Distr. for ETPDFs
System Optimization

Problem:
- Optimize task mapping and priority assignment, such that a required percentage of satisfied deadlines is achieved.
Problem:

- Optimize task mapping and priority assignment, such that a required percentage of satisfied deadlines is achieved.
The presented analysis approach works for large multiprocessor applications.

It is too slow to be used inside an optimization loop.
An approximate analysis method of polynomial complexity has been developed.

Basic idea:

- Weak dependencies among the random variables have been neglected.

Very fast analysis, sufficiently accurate to guide the design space exploration.
If you optimize your system to minimise average or worst case execution time
If you optimize your system to minimise average or worst case execution time and you hope that the resulted design is close to optimal with regard to the percentage of missed deadlines.
If you optimize your system to minimise average or worst case execution time and you hope that the resulted design is close to optimal with regard to percentage of missed deadlines then

YOU ARE VERY WRONG!!!
Conclusions

- Distributed embedded systems are becoming common.
- They are communication dominated and heterogeneous.
- They have to be safe (fault tolerance).
- Hard distributed real-time systems.
- Soft distributed real-time systems.

Analysis (in particular timing) is difficult, but possible.

Simulation is not the right solution!

It is not sufficient to analyse! Help the designer to implement a cost effective system that satisfies the imposed constraints!

Efficient optimization tools are needed.

How to guarantee timing constr.?