Task- and Network-level Schedule Co-Synthesis of Ethernet-based Time-triggered Systems

Licong Zhang\textsuperscript{1}, Dip Goswami\textsuperscript{2}, Reinhard Schneider\textsuperscript{1}, Samarjit Chakraborty\textsuperscript{1}

\textsuperscript{1}Institute for Real-Time Computer Systems, TU Munich, Germany
\textsuperscript{2}Electrical Engineering Department, TU Eindhoven, Netherlands

ASP-DAC 2014
Jan. 21, 2014
Overview

- **Problem**
  - Ethernet-based time-triggered system
  - Co-synthesis of task and communication schedule
  - Application-level (multi-)objectives
Overview

- **Problem**
  - Ethernet-based time-triggered system
  - Co-synthesis of task and communication schedule
  - Application-level (multi-)objectives

- **Approach**
  - Formulation of the problem in Mixed Integer Programming model
    - System description, constraints and objectives formulation

**Configurations**
- applications
- tasks
- communication
- network topology
- device performance

**Objectives**
- timing requirements
- timing objectives

**MIP model**
- system description
- constraints formulation
- objectives formulation

**Synthesized schedules**
- task schedules
- communication schedules
- results of objectives

21.01.2014, ASP-DAC
Licong Zhang/ RCS,TUM
Outline

- Motivation
- Ethernet-based Time-triggered System
- Constraints Formulation
- Multi-objective Optimization
- Experimental Results
- Concluding Remarks
Motivation

- **Ethernet in safety-critical domain**
  - Safety-critical domains: avionics, automotive, industrial automation
  - Increased complexity and load on communication
  - Conventional buses reaching limits (e.g. CAN, FlexRay in automotive)
  - Progress in Ethernet offers better determinism and QoS
Motivation

- **Ethernet in safety-critical domain**
  - Safety-critical domains: avionics, automotive, industrial automation
  - Increased complexity and load on communication
  - Conventional buses reaching limits (e.g. CAN, FlexRay in automotive)
  - Progress in Ethernet offers better determinism and QoS

- **Need for performance guarantees**
  - Safety-critical applications (e.g. vehicle/plane dynamics control)
  - Need for ultra-low latency, jitter and determinism
Motivation

Ethernet in safety-critical domain
- Safety-critical domains: avionics, automotive, industrial automation
- Increased complexity and load on communication
- Conventional buses reaching limits (e.g. CAN, FlexRay in automotive)
- Progress in Ethernet offers better determinism and QoS

Need for performance guarantees
- Safety-critical applications (e.g. vehicle/plane dynamics control)
- Need for ultra-low latency, jitter and determinism

Time-triggered systems
- Offers determinism
- Schedules can be synthesized to minimize latency
Motivation

- Task- and communication-level schedule co-synthesis
  - Application-level timing more important (e.g. feedback control loop)
  - Schedules of tasks and communication must be synchronized
  - Separate task or communication schedule synthesis
    -> not leading to optimal application-level timing properties
Motivation

- **Task- and communication-level schedule co-synthesis**
  - Application-level timing more important (e.g. feedback control loop)
  - Schedules of tasks and communication must be synchronized
  - Separate task or communication schedule synthesis
    -> not leading to optimal application-level timing properties

- **Related work**
  - On general time-triggered architecture [6]
  - Schedule synthesis of FlexRay-based time-triggered system [7,8,9]
  - Communication schedule synthesis of time-triggered Ethernet [10,11,12,13]
Motivation

- **Task- and communication-level schedule co-synthesis**
  - Application-level timing more important (e.g. feedback control loop)
  - Schedules of tasks and communication must be synchronized
  - Separate task or communication schedule synthesis
    -> not leading to optimal application-level timing properties

- **Related work**
  - On general time-triggered architecture [6]
  - Schedule synthesis of FlexRay-based time-triggered system[7,8,9]
  - Communication schedule synthesis of time-triggered Ethernet [10,11,12,13]

- **Contributions**
  - Task and communication schedule co-synthesis in Ethernet-based time-triggered system (problem formulation in Mixed Integer Programming)
  - Multi-objective optimization according to application-level objectives
Distributed system

- Task partition and mapping onto different processing units
- Data sent through a network (e.g. CAN, Ethernet)
- Application-level timing -> interplay between tasks and communication

\[ \alpha \]

application
Distributed system

- Task partition and mapping onto different processing units
- Data sent through a network (e.g. CAN, Ethernet)
- Application-level timing -> interplay between tasks and communication

![Diagram of a time-triggered distributed system]

- Application
- Tasks/messages

Licong Zhang/ RCS,TUM
Time-triggered Distributed System

- **Distributed system**
  - Task partition and mapping onto different processing units
  - Data sent through a network (e.g. CAN, Ethernet)
  - Application-level timing -> interplay between tasks and communication
Time-triggered Distributed System

- **Distributed system**
  - Task partition and mapping onto different processing units
  - Data sent through a network (e.g. CAN, Ethernet)
  - Application-level timing -> interplay between tasks and communication

- **Time-triggered non-preemptive task scheduling**
  - Pre-defined static schedule / a task can not be preempted (e.g. eCos)

- **Time-triggered communication scheduling**
  - Pre-defined static schedule for message transmission (e.g. FlexRay static seg., TTP)
Time-triggered Ethernet Communication

- **Switched Ethernet**
  - Processing units connected through switches
  - Commonly with full-duplex links
  - Ethernet frames forwarded switch by switch

![Switched Ethernet Diagram]

- Licong Zhang/ RCS,TUM
- 21.01.2014, ASP-DAC
Time-triggered Ethernet Communication

- **Switched Ethernet**
  - Processing units connected through switches
  - Commonly with full-duplex links
  - Ethernet frames forwarded switch by switch

- **Network latency**
  - Propagation delay (negligible)
  - Transmission delay
  - Switch delay
Time-triggered Ethernet Communication

- **Switched Ethernet**
  - Processing units connected through switches
  - Commonly with full-duplex links
  - Ethernet frames forwarded switch by switch

- **Network latency**
  - Propagation delay (negligible)
  - Transmission delay
  - Switch delay
    - Processing delay
    - Queuing delay
    - not deterministic
    - can be relatively large
Time-triggered Ethernet Communication

- Time-triggered Ethernet communication
  - Frames are scheduled to avoid queuing delay
  - Frames are not queued at the output port
  - Frame transmission on each link according to static schedule
Time-triggered Ethernet Communication

- **Time-triggered Ethernet communication**
  - Frames are scheduled to avoid queuing delay
  - Frames are not queued at the output port
  - Frame transmission on each link according to static schedule

![Diagram of network communication](image)

- **End Station (Processing Unit)**
- **Switch**

- **Transmission Time Schedule**
Time-triggered Ethernet Communication

- **Time-triggered Ethernet communication**
  - Frames are scheduled to avoid queuing delay
  - Frames are not queued at the output port
  - Frame transmission on each link according to static schedule

![Diagram of a network showing the transmission schedule for different links.](attachment:image.png)

- End Station (Processing Unit)
- Switch

**Transmission Time Schedule**

- Link 1 (1->5)
- Link 2 (2->5)
- Link 2 (5->2)
- Link 3 (5->6)
- Link 4 (6->3)
- Link 5 (6->4)
Time-triggered Ethernet Communication

- **Time-triggered Ethernet communication**
  - Frames are scheduled to avoid queuing delay
  - Frames are not queued at the output port
  - Frame transmission on each link according to static schedule

- Frame 1
  - Link 1
  - Link 3
  - Link 5

- Frame 2
  - Link 1 (1->5)
  - Link 2 (2->5)
  - Link 2 (5->2)
  - Link 3 (5->6)
  - Link 4 (6->3)
  - Link 5 (6->4)

- **End Station (Processing Unit)**
- **Switch**
Time-triggered Ethernet communication

- Frames are scheduled to avoid queuing delay
- Frames are not queued at the output port
- Frame transmission on each link according to static schedule

![Diagram of Time-triggered Ethernet Communication](image)
Time-triggered Ethernet Communication

- **Time-triggered Ethernet communication**
  - Frames are scheduled to avoid queuing delay
  - Frames are not queued at the output port
  - Frame transmission on each link according to static schedule

- **Ethernet-based protocols with time-triggered traffic**
  - Profinet IRT [1]
  - Time-triggered traffic in TT Ethernet [3]
  - IEEE802.1Qbv (not yet released) [5]
Problem Formulation

- **Topology** \( G(\mathcal{V}, \mathcal{E}) \)
  - \( v_i \in \mathcal{V} \) → processing units or switches
  - \( l_{m,n} \in \mathcal{E} \) → Ethernet links
Problem Formulation

- **Topology** $G(V, E)$
  
  $v_i \in V \quad \longrightarrow \quad \text{processing units or switches}$

  $l_{m,n} \in E \quad \longrightarrow \quad \text{Ethernet links}$

- **Application task** $\tau$

  $\tau_i = \{\tau_i.p, \tau_i.o, \tau_i.e\}$

  $\downarrow \quad \downarrow \quad \downarrow$

  period, offset, WCET
Problem Formulation

- **Topology** \( G(\mathcal{V}, \mathcal{E}) \)
  
  \( v_i \in \mathcal{V} \) → *processing units or switches*

  \( l_{m,n} \in \mathcal{E} \) → *Ethernet links*

- **Application task** \( \mathcal{T} \)

  \[ \tau_i = \{ \tau_i.p, \tau_i.o, \tau_i.e \} \]

  *period, offset, WCET*

- **Communication task** \( \mathcal{C} \)

  \[ c_i = \{ f_i, c_i.tr, c_i.o, c_i.p \} \]

  *frame, path tree, offsets, period*
Problem Formulation

- **Topology** $G(V, E)$
  
  $v_i \in V$ → processing units or switches
  
  $l_{m,n} \in E$ → Ethernet links

- **Application task** $\mathcal{T}$
  
  $\tau_i = \{\tau_i.p, \tau_i.o, \tau_i.e\}$
  
  period, offset, WCET

- **Communication task** $C$
  
  $c_i = \{f_i, c_i.tr, c_i.o, c_i.p\}$
  
  frame, path tree, offsets, period

  **path** ← $c_i.ph_j$
  
  from sender to one receiver

  **path tree** ← $c_i.tr = \{c_i.ph_1, c_i.ph_2, \ldots\}$
  
  all paths in a communication task
**Problem Formulation**

- **Application** $\alpha$
  
  $$a_i = \{a_i.tc, a_i.p, a_i.rt, a_i.lz\}$$
  
  *task chain, period*
  
  *response time, latency*
Problem Formulation

- **Application** $\alpha$
  
  $a_i = \{a_i.tc, a_i.p, a_i.rt, a_i.lz\}$
  
  task chain, period
  
  response time, latency

- **task chain** $\leftarrow a_i.tc$
  
  all application and communication tasks in temporal order

- **response time** $\leftarrow a_i.rt$
  
  time from period begin to the end of last task in task chain

- **end-to-end latency** $\leftarrow a_i.lz$
  
  time from begin of first task to the end of last task in task chain
Problem Formulation

- **Application** $\mathcal{A}$
  
  $a_i = \{a_i.tc, a_i.p, a_i.rt, a_i.lz\}$

  *task chain, period*

  *response time, latency*

- **Schedule co-synthesis problem**
  
  - To co-synthesize
    
    - task schedules $\{\tau_i.o\}$
    
    - communication schedules $\{c_i.o\}$

  according to **application-level objectives**

  (e.g. end-to-end latency, response time)
Mixed Integer Programming (MIP)

- Mixed Integer (Linear) Programming:

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax \leq b \\
& \quad lb \leq x \leq ub \\
& \quad \text{some variables in } x \text{ must take integer values}
\end{align*}
\]

- Model formulation
  - Formulate system constraints of the co-synthesis problem into a MIP problem
Constraints

- (C1) Collision-free application tasks
  
  no overlap between execution of two instances of tasks

\[
\begin{align*}
\tau_i.p \times k_i + \tau_i.o + \tau_i.e & < \tau_j.p \times k_j + \tau_j.o \\
\text{or} \\
\tau_j.p \times k_j + \tau_j.o + \tau_j.e & < \tau_i.p \times k_i + \tau_i.o
\end{align*}
\]

\[\text{enumerate instances if periods are not equal}\]

\[\begin{array}{c}
\text{end of } \tau_j \\
\text{begin of } \tau_i
\end{array}\]

\[k_i, k_j\]
Constraints

- **(C1) Collision-free application tasks**
  - no overlap between execution of two instances of tasks

  \[
  \tau_i.p \times k_i + \tau_i.o + \tau_i.e \leq \tau_j.p \times k_j + \tau_j.o \text{ or } \\
  \tau_j.p \times k_j + \tau_j.o + \tau_j.e \leq \tau_i.p \times k_i + \tau_i.o
  \]

  - enumerate instances if periods are not equal

- **(C2) Collision-free communication tasks**
  - no overlap between transmission of two frames

  \[
  c_i.p \times k_i + c_i.o^{l,m,n} + f_i.fl/bw + i.fg \leq c_j.p \times k_j + c_j.o^{l,m,n} \text{ or } \\
  c_j.p \times k_j + c_j.o^{l,m,n} + f_j.fl/bw + i.fg \leq c_i.p \times k_i + c_i.o^{l,m,n}
  \]

  - Inter-frame gap
Constraints

- **(C3) Path dependency**
  - Communication schedules
    - Correct temporal order in the path

\[
c_i.o[ph_i, q - 1] + f_i.fl/bw + pd + sync < c_i.o[ph_j, q]
\]

schedule on one link    schedule on the following link

- Communication schedules on one link followed by communication schedules on the following link.
## Constraints

- **(C3) Path dependency**
  - Communication schedules
    -> correct temporal order in the path

\[
c_{i}.o[ph_{i}, q-1] + f_{i}.fl/bw + pd + sync < c_{i}.o[ph_{j}, q]
\]

- **(C4) Data dependency**
  - task and communication schedules
    -> correct temporal order in task chain

\[
\text{if } \tau_{i} \text{ followed by } \tau_{j} \\
\tau_{i}.o + \tau_{i}.e < \tau_{j}.o
\]

\[
\text{if } \tau_{i} \text{ followed by } c_{j} \\
\tau_{i}.o + \tau_{i}.e + sd < c_{j}.o[first]
\]

\[
\text{if } c_{i} \text{ followed by } \tau_{j} \\
c_{i}.o[last] + f_{i}.fl/bw + sync + rd < \tau_{j}.o
\]
(C5) Application response time
- Response time < upper bound

\[ a_i \cdot rt < a_i \cdot rt_{max} \]
Constraints

- **(C5) Application response time**
  - Response time < upper bound

  \[ a_i.r_t < a_i.r_{t_{\text{max}}} \]

- **(C6) Application end-to-end latency**
  - End-to-end latency < upper bound

  \[ a_i.l_z < a_i.l_{z_{\text{max}}} \]
Multi-Objective Optimization

- **Application-level objectives**
  - **Response time**
    - Applications that need to be finished as soon as possible in a period
    - E.g. platform/system states, data/state integrity checks

For a set of applications \( \mathcal{A}(\text{obj}) \) \( \forall i, a_i \in \mathcal{A}(\text{obj}) \)

- **Max. response time:** \( \text{obj} = \max(a_i.rt) \)
- **AVG. response time:** \( \text{obj} = \sum a_i.rt/N \)

![Diagram showing response time, latency, and period in an application](image-url)
Multi-Objective Optimization

- **Application-level objectives**
  - **Response time**
    -> Applications that need to be finished as soon as possible in a period
    -> E.g. platform/system states, data/state integrity checks
  ```
  For a set of applications \( \mathcal{A}(\text{obj}) \) \( \forall i, a_i \in \mathcal{A}(\text{obj}) \)
  \[
  \begin{align*}
    \text{Max. response time:} & \quad \text{obj} = \max(a_i.r_t) \\
    \text{AVG. response time:} & \quad \text{obj} = \sum a_i.r_t/N
  \end{align*}
  \]
  ```
  - **End-to-end latency**
    -> Applications that need to have a low latency
    -> E.g. feedback control loops
  ```
  \[
  \begin{align*}
    \text{Max. latency:} & \quad \text{obj} = \max(a_i.l_z) \\
    \text{AVG. latency:} & \quad \text{obj} = \sum a_i.l_z/N
  \end{align*}
  \]
  ```
Multi-Objective Optimization

- **Application-level objectives**
  - **Response time**
    -> Applications that need to be finished as soon as possible in a period
    -> E.g. platform/system states, data/state integrity checks
    
    \[ \text{Max. response time: } \forall i, a_i \in A(\text{obj}) \]
    \[ \text{obj} = \max(a_i.rt) \]
    \[ \text{AVG. response time: } \text{obj} = \frac{\sum a_i.rt}{N} \]

  - **End-to-end latency**
    -> Applications that need to have a low latency
    -> E.g. feedback control loops
    
    \[ \text{Max. latency: } \text{obj} = \max(a_i.lz) \]
    \[ \text{AVG. latency: } \text{obj} = \frac{\sum a_i.lz}{N} \]

- **Multi-objective optimization**
  - Optimize according to several objectives
    
    \[ \text{For all objectives } \{\text{obj}_i\} \]
    \[ \text{obj}_M = \sum \text{obj}_i \times \omega_i \]
MIP Model Formulation/Solving

- **Constraints and objective formulation MIP**
  - Simple inequity constraints:
    -> straight forward constraint formulation
  - Either-or constraints (e.g. collision free constraints) :
    -> introduce a binary decision variable and formulate the constraint with two inequities [15]
  - Mini-max objective (e.g. max. latency of N applications):
    -> introduce a continuous variable in the objective function and N inequities in the constraints [15]

- **Solving the MIP models**
  - Commercial or non-commercial solvers (e.g. Gurobi, Cplex)
Case Study

- **System description**
  - 30 applications: $a_1$ to $a_{30}$, 53 application tasks, 23 communication tasks (frames)
  - Harmonic periods – {4,5,10,20} ms, various WCETs and frame lengths

- **Network topologies**
  - 12 processing units
  - 4 topologies
Case Study

- **System description**
  - 30 applications: $a_1$ to $a_{30}$, 53 application tasks, 23 communication tasks (frames)
  - Harmonic periods – \{4,5,10,20\} ms, various WCETs and frame lengths

- **Network topologies**
  - 12 processing units
  - 4 topologies

- **Optimization Objectives**
  - $obj_1$: max. response time of $a_1$ to $a_{30}$
  - $obj_2$: max. response time of $a_1$ to $a_5$
  - $obj_3$: max. response time of $a_1$ to $a_{10}$
  - $obj_4$: avg. response time of $a_1$ to $a_{30}$
  - $obj_5$: max. end-to-end latency of $a_1$ to $a_{30}$
Experimental Results

- Experimental Results
  - Comparison of different single-objective optimizations in tree topology

![Graph comparing different optimization objectives](image-url)
Experimental Results

- Comparison of different single-objective optimizations in tree topology
Experimental Results

- Experimental Results
  - Comparison of different multi-objective optimizations in tree topology

\[ \text{multi-objective case} \]
\[ \text{obj}_1, \text{obj}_2, \text{obj}_3 \]
Experimental Results

- Comparison of different multi-objective optimizations in tree topology

**multi-objective case**

\[ obj_1, obj_2, obj_3 \]

**multi-objective case**

\[ obj_1, obj_4, obj_5 \]
Experimental Results

- Influence of weight in multi-objective optimization

multi-objective case with different weight ratio for $\text{obj}_1, \text{obj}_4$

$$\text{obj}_M = \text{obj}_1 \times \omega_1 + \text{obj}_4 \times \omega_4$$
Computational Cost/Scalability

- **Scalability analysis**
  - Synthetic test configurations from size of 9 applications to 90 applications
  - Setup: 1.87GHz dual core CPU, 4 GB memory, MATLAB 2010 with Gurobi 5.10
Concluding Remarks

- **Approach**
  - Schedule co-synthesis problem for Ethernet-based time-triggered system
  - Formulation of constraints in such a system
  - Multi-objective optimization
Concluding Remarks

- **Approach**
  - Schedule co-synthesis problem for Ethernet-based time-triggered system
  - Formulation of constraints in such a system
  - Multi-objective optimization

  - Co-synthesis of task and communication schedules according to application-level objectives
  - Independent of task and communication configuration, network topologies and device performance
Concluding Remarks

- **Approach**
  - Schedule co-synthesis problem for Ethernet-based time-triggered system
  - Formulation of constraints in such a system
  - Multi-objective optimization
  - Co-synthesis of task and communication schedules according to application-level objectives
  - Independent of task and communication configuration, network topologies and device performance

- **Outlook**
  - Extensibility and sustainability of synthesized schedules
  - Local sub-optimal searches for plug-in schedules
  - Schedule synthesis according to function-level properties
References

[16] “www.gurobi.com”
The End

Many thanks

Q/A