Schedule Management Framework for Cloud-based Future Automotive Software Systems

Licong Zhang\textsuperscript{1}, Debayan Roy\textsuperscript{1}, Philipp Mundhenk\textsuperscript{2}, Samarjit Chakraborty\textsuperscript{1}

\textsuperscript{1}TU Munich, Germany
\textsuperscript{2}TUM CREATE, Singapore

RTCSA 2016
Aug. 17, 2016, Daegu
Overview

- **Problem**
  - Schedule synthesis for Ethernet-based time-triggered system
  - Online schedule generation and management for Plug-and-Play scenario
Overview

- **Problem**
  - Schedule synthesis for Ethernet-based time-triggered system
  - Online schedule generation and management for Plug-and-Play scenario

- **Approach**
  - Software framework for schedule management
  - Utilization of both local computation and cloud-computing
  - Four-stage scheduling strategy, online schedule synthesis, configuration pool
Outline

- Motivation
- Background
- Problem Formulation
- Proposed Framework
- Experimental Results
- Concluding Remarks
Motivation

- **Software update and installation after sales**
  - Shift of innovation in automotive domain to Electrical/Electronics systems and software
  - Development cycle of electronic system and software is much shorter than vehicle life cycle
  - Increasingly more new software functions, e.g., in driver assistance, autonomous driving and infotainment domain
Motivation

- Software update and installation after sales
  - Shift of innovation in automotive domain to Electrical/Electronics systems and software
  - Development cycle of electronic system and software is much shorter than vehicle life cycle
  - Increasingly more new software functions, e.g., in driver assistance, autonomous driving and infotainment domain

- Functionality of a vehicle becomes 'outdated' easily
Motivation

- **Software update and installation after sales**
  - Shift of innovation in automotive domain to Electrical/Electronics systems and software
  - Development cycle of electronic system and software is much shorter than vehicle life cycle
  - Increasingly more new software functions, e.g., in driver assistance, autonomous driving and infotainment domain

- Functionality of a vehicle becomes 'outdated' easily

- In the future it would advantageous for both car manufacturers and customers that
  - Software functions can be constantly updated
  - Newly developed software can be deployed after sales
Motivation

- **Software update and installation after sales**
  - Shift of innovation in automotive domain to Electrical/Electronics systems and software
  - Development cycle of electronic system and software is much shorter than vehicle life cycle
  - Increasingly more new software functions, e.g., in driver assistance, autonomous driving and infotainment domain

  - Functionality of a vehicle becomes ‘outdated’ easily

- In the future it would advantageous for both car manufacturers and customers that
  - Software functions can be constantly updated
  - Newly developed software can be deployed after sales

- **Cloud-based future automotive software systems**
  - Internet connection for cars
  - Vehicle is becoming increasingly autonomous

  - Autonomous detection of driving condition and download software applications on demand
Motivation

- **Plug-and-play of software applications**
  - Software applications can be installed or updated through storage devices (e.g., USB stick) or through cloud service
Motivation

- **Plug-and-play of software applications**
  - Software applications can be installed or updated through storage devices (e.g., USB stick) or through cloud service

- **Challenges and issues to be addressed**
  - Supply chain – ECU with single independent function
  - ECU consolidation – ECUs are becoming computing platforms and multiple applications can be mapped on a single ECU
Motivation

- **Plug-and-play of software applications**
  - Software applications can be installed or updated through storage devices (e.g., USB stick) or through cloud service

- **Challenges and issues to be addressed**
  - Supply chain – ECU with single independent function
    - ECU consolidation – ECUs are becoming computing platforms and multiple applications can be mapped on a single ECU
  - Deployment of software components – ECU software into single binary file
    - Related research works in developing operating systems and runtime environment to accommodate new software without flashing the whole ECU
Motivation

- **Plug-and-play of software applications**
  - Software applications can be installed or updated through storage devices (e.g., USB stick) or through cloud service

- **Challenges and issues to be addressed**
  - Supply chain – ECU with single independent function
    - ECU consolidation – ECUs are becoming computing platforms and multiple applications can be mapped on a single ECU
  - Deployment of software components – ECU software into single binary file
    - Related research works in developing operating systems and runtime environment to accommodate new software without flashing the whole ECU
  - Reallocation of communication and computation resources
    - The problem to be addressed in this work
Motivation

- **Resource reallocation**
  - Change of configuration for processor scheduling and network scheduling
  - Issues like schedule generation, schedule deployment and safe reconfiguration process need to be addressed
Motivation

- **Resource reallocation**
  - Change of configuration for processor scheduling and network scheduling
  - Issues like schedule generation, schedule deployment and safe reconfiguration process need to be addressed

  ➢ This work focuses on the schedule generation and management problem
Motivation

- **Resource reallocation**
  - Change of configuration for processor scheduling and network scheduling
  - Issues like schedule generation, schedule deployment and safe reconfiguration process need to be addressed

- **This work focuses on the schedule generation and management problem**

- **Time-triggered System**
  - Task and network transmission are triggered according to pre-calculated schedules
  - In the case of adding new applications or the update changes requirements, new schedule set needs to be calculated.
Motivation

- **Resource reallocation**
  - Change of configuration for processor scheduling and network scheduling
  - Issues like schedule generation, schedule deployment and safe reconfiguration process need to be addressed

  ➢ This work focuses on the schedule generation and management problem

- **Time-triggered System**
  - Task and network transmission are triggered according to pre-calculated schedules
  - In the case of adding new applications or the update changes requirements, new schedule set needs to be calculated.

- **Requirements**
  - Obtain schedules online in relatively short time
  - As many as possible new applications can be accommodated
  - Facilitation of schedule reuse and minimization of disturbance to existing schedules
Motivation

- Related Works
  - Schedule synthesis problem for Ethernet-based time-triggered systems [10,11,13,15,16]
  - Incremental scheduling [11,12]
  - Configuration and reconfiguration of time-triggered Ethernet networks [18,19]
  - Plug-and-Play in the automotive setting [2,3,5,6]
Motivation

- **Related Works**
  - Schedule synthesis problem for Ethernet-based time-triggered systems [10,11,13,15,16]
  - Incremental scheduling [11,12]
  - Configuration and reconfiguration of time-triggered Ethernet networks [18,19]
  - Plug-and-Play in the automotive setting [2,3,5,6]

- **Contributions**
  - Software framework for schedule generation and management for Plug-and-Play
  - Online schedule synthesis based on a mixture of embedded and cloud computing
  - Configuration pool for schedule reuse
  - A four-stage scheduling strategy offering trade-off between chance of accommodating new applications and synthesis time and disturbance to existing schedules
Background

- Distributed embedded systems
  - Hardware architecture
Background

- Distributed embedded systems
  - Hardware architecture
  - Distributed applications

![Diagram of a communication network with ECU nodes and arrows indicating communication paths.]

Licong Zhang, RTCSA 16
Background

- **Distributed embedded systems**
  - Hardware architecture
  - Distributed applications
  - Task mapping / network communication
Background

- **Distributed embedded systems**
  - Hardware architecture
  - Distributed applications
  - Task mapping / network communication

- **Switched Ethernet**
  - Processing units connected through switches
  - Commonly with full-duplex links
Background

- **Distributed embedded systems**
  - Hardware architecture
  - Distributed applications
  - Task mapping / network communication

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

- **Distributed embedded systems**
  - Hardware architecture
  - Distributed applications
  - Task mapping / network communication

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

  - Queueing delay at each switch
    - Not deterministic
    - Can be relatively large
Background

- **Time-triggered Ethernet communication**
  - Frames are scheduled to avoid queueing delay
  - Frames transmission on each link according to static schedule

![Diagram of a network with end stations and switches](Diagram.png)

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

- **Transmission Time Schedule**

---

Licong Zhang, RTCSA 16
Background

- **Time-triggered Ethernet communication**
  - Frames are scheduled to avoid queueing delay
  - Frames transmission on each link according to static schedule

![Diagram of network and frames transmission](image)

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

<table>
<thead>
<tr>
<th>Link</th>
<th>Transmission Time Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link 1 (1-&gt;5)</td>
<td></td>
</tr>
<tr>
<td>Link 2 (2-&gt;5)</td>
<td></td>
</tr>
<tr>
<td>Link 2 (5-&gt;2)</td>
<td></td>
</tr>
<tr>
<td>Link 3 (5-&gt;6)</td>
<td></td>
</tr>
<tr>
<td>Link 4 (6-&gt;3)</td>
<td></td>
</tr>
<tr>
<td>Link 5 (6-&gt;4)</td>
<td></td>
</tr>
</tbody>
</table>
Background

- **Time-triggered Ethernet communication**
  - Frames are scheduled to avoid queueing delay
  - Frames transmission on each link according to static schedule

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

```
<table>
<thead>
<tr>
<th>Link 1</th>
<th>Link 2</th>
<th>Link 3</th>
<th>Link 4</th>
<th>Link 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame 1</td>
<td>Frame 2</td>
<td>Frame 3</td>
<td>Frame 4</td>
<td></td>
</tr>
</tbody>
</table>

Transmission 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)
Background

- Time-triggered Ethernet communication
  - Frames are scheduled to avoid queueing delay
  - Frames transmission on each link according to static schedule

- Ethernet-based time-triggered systems
  - Processor: time-triggered non-preemptive task scheduling
  - Network: time-triggered Ethernet communication
Problem Formulation

- The scheduling problem
  - Application task
    \[ \tau_i = \{ \tau_i.p, \, \tau_i.o, \, \tau_i.e \} \]
    \[ \text{period} \quad \text{offset} \quad \text{WCET} \]
Problem Formulation

- The scheduling problem
  - Application task
    \[ \tau_i = \{ \tau_i.p, \tau_i.o, \tau_i.e \} \]
    - period
    - offset
    - WCET
  - Communication task
    \[ c_i = \{ f_i, c_i.tr, c_i.o, c_i.p \} \]
    - frame length
    - path tree
    - offsets
    - period

Licong Zhang, RTCSA 16
Problem Formulation

- The scheduling problem
  - Application task
    \[ \tau_i = \{ \tau_i.p, \tau_i.o, \tau_i.e \} \]
    period offset WCET
  - Communication task
    \[ c_i = \{ f_i, c_i.tr, c_i.o, c_i.p \} \]
    frame length path tree offsets period
  - Application
    \[ a_i = \{ a_i.tc, a_i.p, a_i.rt, a_i.lz \} \]
    period task chain response time latency

Licong Zhang, RTCSA 16
Problem Formulation

- **The scheduling problem**
  - Hardware specific parameters $h_\omega$
    - System topology
    - Timing parameters including network bandwidth, synchronization precision, etc.

- Application parameters $A_\omega$
  - Task mapping, period, WCET
  - Communication frame length, path tree, latency and response time constraints

- Application schedules $A_o$
  - Task schedules and frame transmission schedules on each link
Problem Formulation

- **The scheduling problem**
  - Hardware specific parameters $h\omega$
    - System topology
    - Timing parameters including network bandwidth, synchronization precision, etc.

- Application parameters $A_\omega$
  - Task mapping, period, WCET
  - Communication frame length, path tree, latency and response time constraints

- Application schedules $A_\omega$
  - Task schedules and frame transmission schedules on each link

- **Approach**
  - Formulation of the problem in SMT or MIP problem and use solvers to obtain the schedules, as in [10,11,13,15,16]
Problem Formulation

- **The schedule management problem**
  - Consider a system with $\mathcal{A}_w$, $\mathcal{A}_o$, and existing application set $\mathcal{A}_o$, $\mathcal{A}_o$.
  - Obtain $\mathcal{A}_n$ for the new application set $\mathcal{A}_n$, while addressing the requirements:
    - Obtain schedules in relatively short time
    - As many as possible new applications can be accommodated
    - Facilitation of schedule reuse and minimization of disturbance to existing schedules
Problem Formulation

- **The schedule management problem**
  - Consider a system with $h \omega$ and existing application set $A_o \cdot \omega$, $A_o \cdot o$
  - Obtain $A_n \cdot o$ for the new application set $A_n$ with $A_n \cdot \omega$, while addressing the requirements:
    - Obtain schedules in relatively short time
    - As many as possible new applications can be accommodated
    - Facilitation of schedule reuse and minimization of disturbance to existing schedules

- **Alternatives and Challenges**
  - Synthesize schedules offline for a specific application -> conflicts with existing schedules
  - Synthesize all possible schedule sets offline -> possibly a huge number combinations
  - Online schedule synthesis on-board -> long synthesis time due to limited computing power
Problem Formulation

- **The schedule management problem**
  - Consider a system with \( h\omega \) and existing application set \( A_0 \omega \), \( A_o \cdot O \)
  - Obtain \( A_n \cdot O \) for the new application set \( A_n \) with \( A_n \cdot \omega \), while addressing the requirements:
    - Obtain schedules in relatively short time
    - As many as possible new applications can be accommodated
    - Facilitation of schedule reuse and minimization of disturbance to existing schedules

- **Alternatives and challenges**
  - Synthesize schedules offline for a specific application -> conflicts with existing schedules
  - Synthesize all possible schedule sets offline -> possibly a huge number combinations
  - Online schedule synthesis on-board -> long synthesis time due to limited computing power

- Need of an online schedule management framework

Licong Zhang, RTCSA 16
Schedule Management Framework

- **Overview**
  - Client-server architecture
  - Utilization of both onboard processor and cloud-computing
  - Components: Synthesis Module, Web Module, Configuration Pool, Management Module

![Diagram of Schedule Management Framework]

- **Server**
  - Synthesis Module
    - Formulation & Interpretation
    - Schedule Synthesis
  - Web Module
    - Websocket Secure
  - Configuration Pool
  - Management Module

- **Client**
  - Synthesis Module
    - Formulation & Interpretation
    - Schedule Synthesis
  - Web Module
    - Websocket Secure
  - Configuration Pool
  - Management Module

- **Internet through WLAN or 3G/LTE/4G**

- **Offline**
  - Schedule Synthesis

- **Request**
- **Deployment**
Schedule Management Framework

- Configuration and request
  - TTCN
    - An XML format containing $h w \omega$, $A_\omega$ and $A_o$
    - Configuration: all application schedules have valid values
    - Request: some application schedules are empty
Schedule Management Framework

- **Configuration and request**
  - TTNCON
    - An XML format containing \( h_w, A_w \) and \( A_o \)
    - Configuration: all application schedules have valid values
    - Request: some application schedules are empty

- **Configuration Pool**
  - Can be managed through different metrics like frequency of reuse
  - Retrieve a configuration
    - If the application set of request matches exactly or is a subset of a configuration in pool, the configuration can be retrieved
    - In the case of a subset, schedules of other applications are removed
  - Update the configuration pool
    - Add a configuration if it is not in pool
    - If the new configuration is a superset of an existing one, it replaces the existing one
  - It facilitates schedule reuse for a single vehicle or between vehicles of the same variant and request based configuration management
Schedule Management Framework

- Synthesis module

```
next stage

Pre-processing

Model Formulation

Solver

is Solved?

Configuration Generation

no

yes
```
Schedule Management Framework

- Synthesis module

next stage

Pre-processing

Model Formulation

Solver

is Solved?

yes

Configuration Generation

determine which applications are reschedulable

\[ C_o \quad C_r \quad stage \]

\[ A_k \quad A_u \]
Schedule Management Framework

- Synthesis module

- Pre-processing
- Model Formulation
- Solver
- is Solved?
  - yes: Configuration Generation
  - no: next stage

\[ C_o \quad C_r \quad stage \]

- determine which applications are reschedulable
- formulate constraint programming problem
- \( A_k \quad A_u \)
- \( M \)
Schedule Management Framework

- Synthesis module

- Pre-processing
- Model Formulation
- Solver
- is Solved?
- Configuration Generation

- Next stage
- No

- Determine which applications are reschedulable
- Formulate constraint programming problem
- Solve model

- Configuration Generation

Licong Zhang, RTCSA 16
Schedule Management Framework

- Synthesis module

next stage

- Pre-processing
- Model Formulation
- Solver
- is Solved?
  - yes: Configuration Generation
  - no: Pre-processing

![Flowchart](image)

- Determine which applications are reschedulable
- Formulate constraint programming problem
- Solve model
- Generate configuration

License: cc-by-sa
Schedule Management Framework

- Four-stage scheduling strategy
  - Stage 1 – Incremental scheduling
    - None of the existing applications are rescheduled
    \[ A_k = A_o \cap A_n, A_u = A_n \setminus A_k \]
Schedule Management Framework

- **Four-stage scheduling strategy**
  - Stage 1 – Incremental scheduling
    - None of the existing applications are rescheduled
    \[ A_k = A_o \cap A_n, A_u = A_n \setminus A_k \]
  - Stage 2 – Rescheduling based on task conflict
    - Reschedule existing applications with common tasks with new ones
    \[ A_k = (A_o \cap A_n) \setminus A_t, A_u = A_n \setminus A_k \]
    \[ A_t = \{ a_i | a_i \in A_o^a \cap A_n^a \land \exists a_j \in A_n^a \setminus A_o^a \land \tau_k \in a_i.tc \land \tau_l \in a_j.tc \} \]
Schedule Management Framework

- **Four-stage scheduling strategy**
  - Stage 1 – Incremental scheduling
    - None of the existing applications are rescheduled
    \[ \mathcal{A}_k = \mathcal{A}_o \cap \mathcal{A}_n, \mathcal{A}_u = \mathcal{A}_n \setminus \mathcal{A}_k \]
  - Stage 2 – Rescheduling based on task conflict
    - Reschedule existing applications with common tasks with new ones
    \[ \mathcal{A}_k = (\mathcal{A}_o \cap \mathcal{A}_n) \setminus \mathcal{A}_\tau, \mathcal{A}_u = \mathcal{A}_n \setminus \mathcal{A}_k \]
    \[ \mathcal{A}_\tau = \{ a_i | a_i \in \mathcal{A}_o^a \cap \mathcal{A}_n^a \land \exists a_j \in \mathcal{A}_n^a \setminus \mathcal{A}_o^a \forall \tau_k \in a_i.tc \forall \tau_l \in a_j.tc \} \]
  - Stage 3 – Rescheduling based on computation resource conflict
    - Reschedule existing applications with tasks mapped on common ECU with new ones
    \[ \mathcal{A}_k = (\mathcal{A}_o \cap \mathcal{A}_n) \setminus \mathcal{A}_E, \mathcal{A}_u = \mathcal{A}_n \setminus \mathcal{A}_k \]
    \[ \mathcal{A}_E = \{ a_i | a_i \in \mathcal{A}_o^a \cap \mathcal{A}_n^a \land \exists a_j \in \mathcal{A}_n^a \setminus \mathcal{A}_o^a \forall \tau_k \in a_i.tc \forall \tau_l \in a_j.tc \} \]
Schedule Management Framework

- **Four-stage scheduling strategy**
  - **Stage 1 – Incremental scheduling**
    - None of the existing applications are rescheduled
    \[ A_k = A_o \cap A_n, A_u = A_n \setminus A_k \]
  - **Stage 2 – Rescheduling based on task conflict**
    - Reschedule existing applications with common tasks with new ones
    \[ A_k = (A_o \cap A_n) \setminus A_T, A_u = A_n \setminus A_k \]
    \[ A_T = \{ a_i | a_i \in A_o^a \cap A_n^a \land \exists a_j \in A_n^a \setminus A_o^a, \tau_k \in a_i.tc, \tau_l \in a_j.tc \} \]
  - **Stage 3 – Rescheduling based on computation resource conflict**
    - Reschedule existing applications with tasks mapped on common ECU with new ones
    \[ A_k = (A_o \cap A_n) \setminus A_E, A_u = A_n \setminus A_k \]
    \[ A_E = \{ a_i | a_i \in A_o^a \cap A_n^a \land \exists a_j \in A_n^a \setminus A_o^a, \tau_k \in a_i.tc, \tau_l \in a_j.tc \} \]
  - **Stage 4 – Complete rescheduling**
    - All existing applications are considered reschedulable
    \[ A_k = A^b, A_n = A^a_n \]
Schedule Management Framework

- Four-stage scheduling strategy
  - Stage 1 – Incremental scheduling
    - None of the existing applications are rescheduled
      \[ A_k = A_o \cap A_n, A_u = A_n \setminus A_k \]
  - Stage 2 – Rescheduling based on task conflict
    - Reschedule existing applications with common tasks with new ones
      \[ A_k = (A_o \cap A_n) \setminus A_T, A_u = A_n \setminus A_k \]
      \[ A_T = \{ a_i | a_i \in A_o \cap A_n \land \exists \tau_i \in a_i.tc, \exists \tau_l \in a_j.tc \} \]
  - Stage 3 – Rescheduling based on computation resource conflict
    - Reschedule existing applications with tasks mapped on common ECU with new ones
      \[ A_k = (A_o \cap A_n) \setminus A_E, A_u = A_n \setminus A_k \]
      \[ A_E = \{ a_i | a_i \in A_o \cap A_n \land \exists \exists \tau_k.E = \exists \tau_l.E \} \]
  - Stage 4 – Complete rescheduling
    - All existing applications are considered reschedulable
      \[ A_k = A^b, A_n = A_n^a \]

Licong Zhang, RTCSA 16
Schedule Management Framework

- **Web module**
  - Utilizes the Websocket Secure
    - Full-duplex communication between client and server
    - SSL/TLS layer for secure communication
Schedule Management Framework

- **Web module**
  - Utilizes the Websocket Secure
    - Full-duplex communication between client and server
    - SSL/TLS layer for secure communication

- Methods for client-server communication
  - Request
    - Client sends request file to server
  - Response
    - Server sends response to client: either a valid configuration or a request denial
  - Abort
    - Client informs the server to abort operation, when a local result is obtained first
  - Update
    - Client sends the new configuration to the server to update the configuration pool
Schedule Management Framework

- Management module – client side

![Diagram showing the schedule management framework with modules and arrows indicating the flow of information between them. The diagram includes labels for Web Module, Management Module, Synthesis Module, and Configuration Pool. The flow starts with generating a request, followed by handling current configuration and requested application set.]
Schedule Management Framework

- Management module – client side

  - Management Module
  - Web Module
  - Synthesis Module

  - Configuration Pool

  - generate request
    - check for valid configuration
      - yes
        - requested application set
          - current configuration
  - results received
Schedule Management Framework

- Management module – client side

![Diagram of Schedule Management Framework]

- Web Module
  - Management Module
    - Generate request
      - Check for valid configuration
        - Yes: Wait for results
        - No: Request denied
      - Results received
    - Configuration Pool
      - Pre-processing & model formulation
        - Next stage
      - Schedule synthesis
        - No for all stages
        - Yes: Generate new configuration
  - Synthesis Module
    - Yes for all stages
Schedule Management Framework

- Management module – client side

```
<table>
<thead>
<tr>
<th>Web Module</th>
<th>Management Module</th>
<th>Synthesis Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>send request to server</td>
<td>generate request</td>
<td>schedule synthesis</td>
</tr>
<tr>
<td>receive response from server</td>
<td>check for valid configuration</td>
<td>results received</td>
</tr>
<tr>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td>no for all stages</td>
</tr>
<tr>
<td>generate new configuration</td>
<td>request denied</td>
<td>no for all stages</td>
</tr>
<tr>
<td>yes</td>
<td>results received</td>
<td>yes</td>
</tr>
<tr>
<td>configuration</td>
<td>wait for results</td>
<td>next stage</td>
</tr>
<tr>
<td>requested application set</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Licong Zhang, RTCSA 16
Schedule Management Framework

- Management module – client side

```plaintext
Management Module

1. Current configuration
2. Requested application set
3. Generate request
4. Check for valid configuration
   - Yes: Add to configuration pool
   - No: Request denied

5. Pre-processing & model formulation
6. Schedule synthesis
   - No for all stages
5. Next stage

Web Module

1. Send request to server
2. Receive response from server
   - No: Request denied
   - Yes: Results received
3. Send abort/result to server

Synthesis Module

1. Generate new configuration
2. Deploy configuration
```

Licong Zhang, RTCSA 16
Schedule Management Framework

- Management module – server side

Web Module

Management Module

Configuration Pool

Synthesis Module
Schedule Management Framework

- Management module – server side

1. Receive request from client
2. Check for valid configuration
3. Yes
4. Valid configuration
5. Send response to client

Web Module

Management Module

Configuration Pool

Synthesis Module
Schedule Management Framework

- Management module – server side

- Web Module
  - receive request from client
  - valid configuration
  - request denial
  - send response to client

- Synthesis Module
  - check for valid configuration
  - no
  - yes
    - pre-processing & model formulation
    - schedule synthesis
      - yes
      - no for all stages
        - request denial
        - send response to client
      - next stage
        - pre-processing & model formulation
        - schedule synthesis
          - yes
          - no for all stages
            - request denial
            - send response to client
          - next stage
            - pre-processing & model formulation
            - schedule synthesis
              - yes
              - no for all stages
                - request denial
                - send response to client
              - next stage
                - pre-processing & model formulation
                - schedule synthesis
                  - yes
                  - no for all stages
                    - request denial
                    - send response to client
Schedule Management Framework

- Management module – server side

**Web Module**

- receive request from client
- valid configuration
- request denial
- abort/update from client

**Management Module**

- check for valid configuration
- pre-processing & model formulation
- schedule synthesis
- generate new configuration

**Synthesis Module**

- valid configuration
- request denial
- abort/update from client

**Configuration Pool**

- add to configuration pool

Flowchart:
- yes for valid configuration
- no for all stages
- yes for next stage
- no for all stages
- yes for abort/update

Licong Zhang, RTCSA 16
Results

- **Implementation**
  - Client on a Raspberry PI 2 Model B
  - Server on a PC
  - Connection through WLAN

- **Case study**
  - Hardware architecture: 10 ECUs connected by 4 switches
  - 100 applications are randomly generated (10 basic applications, 90 plug-in applications)
  - 20 request series of incrementally adding applications
  - Different overhead provision for possible authentication and security process on server
Results

- **Synthesis time**
  - Case 0 s overhead provision for server
Results

- **Synthesis time**
  - Case 3 s overhead provision for server
Results

- Scheduling stages

![Diagram showing scheduling stages and time taken for client and server results]
Results

- **Comparison of synthesis time for client, server and proposed framework**
  - Case 1.5 s overhead provision for server
Concluding Remarks

- Problem
  - Ethernet-based time-triggered automotive system
  - Resource reallocation for accommodating new software applications in a Plug-and-Play manner
Concluding Remarks

- **Problem**
  - Ethernet-based time-triggered automotive system
  - Resource reallocation for accommodating new software applications in a Plug-and-Play manner

- **Approach**
  - Client-server based software framework for schedule management
  - Use of local computation and cloud-computing for online schedule synthesis and management
  - Four-stage scheduling strategy for trade-off between synthesis time, disturbance to existing applications and the chances of accommodating new ones
Concluding Remarks

- Problem
  - Ethernet-based time-triggered automotive system
  - Resource reallocation for accommodating new software applications in a Plug-and-Play manner

- Approach
  - Client-server based software framework for schedule management
  - Use of local computation and cloud-computing for online schedule synthesis and management
  - Four-stage scheduling strategy for trade-off between synthesis time, disturbance to existing applications and the chances of accommodating new ones

- Future work
  - Utilize the multi-core architecture to parallelize synthesis methods to reduce synthesis time
  - Explore extensibility-aware scheduling to provision resources for future applications so more applications can be accommodated using incremental design
References

References

Thanks for your attention!

Q/A