Communication Scheduling for Time-Triggered Systems

Paul Pop, Petru Eles and Zebo Peng

Dept. of Computer and Information Science
Linköping University
Sweden
Conditional Process Graph

Subgraph corresponding to $D \land C \land K$
Hardware Architecture

- Safety-critical distributed embedded systems.
- Nodes connected by a broadcast communication channel.
- Nodes consisting of: TTP controller, CPU, RAM, ROM, I/O interface, (maybe) ASIC.
- Communication between nodes is based on the time-triggered protocol.
- Bus access scheme: time-division multiple-access (TDMA).
- Schedule table located in each TTP controller: message descriptor list (MEDL).
Problem Formulation

Input

• Safety-critical application with several operating modes.
• Each operating mode is modelled by a conditional process graph.
• The system architecture and mapping of processes to nodes are given.
• The worst case delay of a process is known:

\[ T_{P_i} = (\delta_{PA} + t_{P_i} + \theta_{C_1} + \theta_{C_2}) \]

\[ \theta_{C_1} = \sum_{i=1}^{N_{local}^{out}(P_i)} \delta_{S_i} \]
\[ \theta_{C_2} = \sum_{i=1}^{N_{remote}^{out}(P_i)} \delta_{KS_i} + \sum_{i=1}^{N_{remote}^{in}(P_i)} \delta_{KR_i} \]

Output

• Local schedule tables for each node and the MEDL for the TTP controllers.
• Delay on the system execution time for each operating mode, so that this delay is as small as possible.
Scheduling Example

Round 1 | Round 2 | Round 3 | Round 4 | Round 5
---|---|---|---|---
$S_1$ | $S_0$ | $m_1$ | $m_2$ | $m_3$ | $m_4$

Round 1 | Round 2 | Round 3 | Round 4
---|---|---|---
$S_0$ | $S_1$ | $m_1$ | $m_2$ | $m_3$ | $m_4$

Round 1 | Round 2 | Round 3
---|---|---
$S_0$ | $S_1$ | $m_1$ | $m_2$ | $m_3$ | $m_4$

24 ms
22 ms
20 ms
Experimental Results

The Greedy Approach is producing accurate results in a very short time (few seconds for graphs with 400 processes).

Greedy 1 performs slightly better than Greedy 2, but it is a bit slower.

SA finds near-optimal results in a reasonable time (few minutes for graphs with 80 processes and 275 minutes for graphs with 400 processes).

A real-life example implementing a vehicle cruise controller validated our approach.