A time-triggered implementation model for real-time distributed systems

Virginia Papailiopoulos
Dumitru Potop-Butucaru
Yves Sorel
INRIA Paris-Rocquencourt

SYNCHRON 2011
Dammarie-les-Lys, France
Outline

• Motivation
  - Avionics embedded computing systems
  - Integrated Modular Avionics (IMA)

• ARINC 653 overview
  - Focus on temporal aspects

• Time-triggered implementation model
  - Reservation/Scheduling tables

• Proposition: time-triggered IMA
  - Time-triggered IMA implementation
Outline

• Motivation
  - Avionics embedded computing systems
  - Integrated Modular Avionics (IMA)

• ARINC 653 overview
  - Focus on temporal aspects

• Time-triggered implementation model
  - Reservation/ Scheduling tables

• Proposition: time-triggered IMA
  - Time-triggered IMA implementation
Integrated Modular Avionics (IMA)

- Better use of hardware resources
- Lower design/maintenance costs

ARINC 653
robust partitioning
2-level scheduling

time partitioning
static TDM scheduling

space partitioning
no unspecified communication
no side-effects

different criticality levels
different safety/reliability requirements

integration of multiple applications in the same computer
data communications with multiplexed network

Level A
Level B
Level C

processor

P_2 P_1 P_2 P_3
0 W_1 W_2 W_3 W_4

SYNCHRON 2011
November 28th-December 2nd - Dammarie-les-Lys, France
Integrated Modular Avionics (IMA)

- Inside each partition:
  - Partition-level scheduler (L1) within TDM slots allocated by the static scheduler (L0)
  - Any scheduling policy can be used (RR, EDF, …)
  - ARINC 653
    - priority-preemptive L1 scheduler
      - easy porting of existing software
Motivation

• Inside each partition:
  - Partition-level scheduler (L1) within TDM slots allocated by the static scheduler (L0)
  - Any scheduling policy can be used (RR, RM, EDF, ...)
  - ARINC 653
    • priority-preemptive L1 scheduler
      - easy porting of existing software

• Dynamic scheduling + static TDM
  - many TDM slots of short duration \(\Rightarrow\) increased cost
  - interruption at the end of TDM slots \(\Rightarrow\) worse deadline guarantees
Proposition

- Fully static scheduling (L0+L1)
  - Time-triggered process scheduling within partition allocated TDM slots
- Conditional scheduling tables
  - Precise start dates
  - Execution condition

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>if ( c₁ ) then f</td>
<td>if ( \neg c₁ )</td>
</tr>
<tr>
<td>t₂</td>
<td></td>
<td>then g</td>
</tr>
<tr>
<td>t₃</td>
<td></td>
<td>if ( c₃ ) then h</td>
</tr>
<tr>
<td>t₄</td>
<td>if ( c₄ ) then f</td>
<td></td>
</tr>
</tbody>
</table>

easy and predictable implementations
simple model for complex systems
better use of resources
automatic generation from data-flow formalisms, e.g. SCADE or Simulink
Outline

• Motivation
  - Avionics embedded computing systems
  - Integrated Modular Avionics (IMA)

• ARINC 653 overview
  - Focus on temporal aspects

• Time-triggered implementation model
  - Reservation/Scheduling tables

• Proposition: time-triggered IMA
  - Time-triggered IMA implementation
ARINC 653

Partition for Application Software 1
Partition for Application Software 2
Partition for Application Software 3

core module

A P E X

O/S

module scheduler (L0)

HW
ARINC 653 - Partitions

- Static allocation of resources
  - One partition ↔ one application

- Static scheduling → TDM
  - Fixed time windows → exclusive access to resources
  - One partition → several windows

\[ MTF = k \times LCM(T_i, ..., T_n) \]
ARINC 653 - Processes

• Application functional behavior
• Priority preemptive scheduling
ARINC 653 - MTF configuration

- According to partition and process requirements

![Diagram showing MTF configuration with partition and process details]
ARINC 653 - MTF configuration

- Not always unique

![Diagram of ARINC 653 MTF configuration]

- Parameters:
  - $T_F = 20$, $d_F = 3$
  - $T_H = 20$, $d_H = 5$
  - $T_{P1} = 20$, $D_{P1} = 8$
  - $T_{P2} = 40$, $D_{P2} = 10$
  - $T_G = 40$, $d_G = 10$

- Time points:
  - $0$, $3$, $15$, $20$, $23$, $25$, $35$, $40$ (MTF)
ARINC 653 - Structure of an implementation

- Configuration file (for the O/S)
  - Module configuration
    - window allocation to partitions
    - memory management
    - module scheduling (window start dates + durations)
- Main programs
  - One for each partition
    - processes creation
    - communication ports creation
    - partition scheduling
Outline

• Motivation
  - Avionics embedded computing systems
  - Integrated Modular Avionics (IMA)

• ARINC 653 overview
  - Focus on temporal aspects

• Time-triggered implementation model
  - Reservation/Scheduling tables

• Proposition: time-triggered IMA
  - Time-triggered IMA implementation
### Time-triggered implementation model

<table>
<thead>
<tr>
<th>Time</th>
<th>P₁</th>
<th>P₂</th>
<th>P₃</th>
<th>Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$F₁ @ true$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>send($P₁, iₐA) @ true$</td>
</tr>
<tr>
<td>2</td>
<td>$F₂ @ iₐA = true$</td>
<td></td>
<td></td>
<td>send($P₃, iₐB) @ true$</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>$F₃ @ iₐB = false$</td>
<td>send($P₃, iₐA) @ true$</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>$N @ iₐA = false$</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$M @ true$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Periodic **non-preemptive** execution model
- Table size = execution cycle duration
- Operations with disjoint conditions can run concurrently
- Data dependencies respected
- No data race
Outline

- Motivation
  - Avionics embedded computing systems
  - Integrated Modular Avionics (IMA)

- ARINC 653 overview
  - Focus on temporal aspects

- Time-triggered implementation model
  - Reservation/Scheduling tables

- Proposition: time-triggered IMA
  - Time-triggered IMA implementation
Time-triggered IMA

- Table size = MTF
- Each operation $o_i$ is associated to a partition $P_i$
- Slot reservation for window changes
- Precomputed preemption
  - Allow for operations spanning over several windows
  - Multiple reservations per operation
Time-triggered implementation

- For each partition
  - One aperiodic process/scheduled operation
  - One periodic process/slot change
  - Fixed priorities
    - higher priority given to periodic processes
  - Start dates fixed w.r.t. the partition period

+ APEX
Time-triggered implementation

#include "local_definitions.c"
const int OpNum;  // # of operations
const int DNum;   // # of start/end dates

processes associated to operations

processes associated to slot changes and the start dates

PROCESS_ATTRIBUTE_TYPE* dates(int Di) {
    return {date_name[Di], slot_change,
            date_stack, HI_PRIO, part_period, 
            date_duration, HARD};
}

const SYSTEM_TIME_TYPE slot_offset[DNum];

void init_inter_partition_ports();

int main() {
    RETURN_CODE_TYPE ret;
    PROCESS_ID_TYPE d_pid;
    init_inter_partition_ports();
    for(int i=0; i<OpNum; i++) {
        CREATE_PROCESS(op(i), OP_PID+i, &ret);
    }
    for(int i=0; i<DNum; i++) {
        CREATE_PROCESS(dates, &d_pid, &ret);
        DELAYED_START(d_pid,
                      slot_offset[i], &ret);
    }
    SET_PARTITION_MODE(NORMAL, &ret);
    return 0;
}

int d_i = DNum-1;
void slot_change() {
    scheduler function
    RETURN_CODE_TYPE ret;
    d_i = (d_i+1)%DNum;
    for(int i=0; i<OpNum; i++) {
        if((op_start[d_i][i]() )
            START(OP_PID[i], &ret);
        else if ((op_resume[d_i][i]() ))
            RESUME(OP_PID[i], &ret);
        else if ((op_suspend[d_i][i]() ))
            SUSPEND(OP_PID[i], &ret);
    }
}
Time-triggered implementation

- No preemption → no aperiodic process

```c
#include "local_definitions.c"
const int opNum = 2;
PROCESS_ATTRIBUTE_TYPE op[opNum] = {
    {"f", f, 1000, LO_PRIO, 0.020, 0.005, HARD},
    {"h", h, 1000, LO_PRIO, 0.020, 0.005, HARD}};
SYSTEM_TIME_TYPE slot_offset[opNum] = {0.005, 0.020} ;
PROCESS_ID_TYPE OP_PID[opNum];

int main () {
    RETURN_CODE_TYPE ret;
    init_inter_partition_ports();
    for(int i=0;i<opNum;i++) {
        CREATE_PROCESS(op[i],OP_PID+i,&ret);
        DELAYED_START(OP_PID[i], slot_offset[i], &ret);
    }

    SET_PARTITION_MODE(NORMAL, &ret);
    return (0);
}
```
Conclusion

ARINC 653 time partitioning constraints
+
conditional scheduling tables
⇓
fully RT static schedule
⇓
Automatic synthesis of ARINC-based implementations from data-flow specifications

• Future work
  - L1 scheduler implementation
  - Evaluation