CAN in Automotive Applications: a Look Forward

Nicolas NAVET (INRIA/RTaW), Hervé PERRAULT (PSA)

13th International CAN Conference
Hambach Castle, March 5-6 2012.

Automotive CAN: the early days (1/2)

<table>
<thead>
<tr>
<th>Priority</th>
<th>Sender node</th>
<th>DLC</th>
<th>Period (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Engine Controller</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Wheel angle sensor</td>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>Engine Controller</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>AGB</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>ABS</td>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>ABS</td>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>ABS</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>Body gateway</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>9</td>
<td>undisclosed</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>10</td>
<td>Engine Controller</td>
<td>7</td>
<td>100</td>
</tr>
<tr>
<td>11</td>
<td>AGB</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>12</td>
<td>ABS</td>
<td>1</td>
<td>100</td>
</tr>
</tbody>
</table>

Early CAN project at PSA (1996, see [1])
250kbit/s

Worst-case latencies (=response times) are less than 5.5 ms

NETCAR-Analyzer screenshot

Proliferation of ECUs and buses

Up to 5 CAN interconnected by gateways

# ECUs and buses in some PSA projects between 2000 and 2010 [2]
Today’s set of messages

- **Size**: Up to 20 nodes and 100 frames
- **Bus-rate**: 250 or 500 kbits
- **Load**: > 50%, sometimes 60% or more …
- **Max latencies**: 5ms or less
- **Gateways**: CAN/CAN or CAN/FLEXRAY induce delays and bursty traffic.
- **Aperiodic traffic** (eg, Autosar mixed transmission mode)

*NETCARBENCH is a GPL licensed software to generate “realistic” and non confidential CAN message sets according to a set of user-defined parameters. Available at www.netcarbench.org*

---

**RTaW : help designers build truly safe and optimized systems**

- **Services and Software for**: architecture design, ECU and network configuration, formal and temporal verification (simulation, analysis, trace-inspection)
- **Communications systems**: CAN, FlexRay, AFDX, industrial Ethernet, TTP, etc …
- **CAN customers**: PSA and Renault

*Most software tools are downloadable at www.realtimeatwork.com / we provide R&D, support and custom extensions*
*No black box software: we publish all algorithms that are implemented (ongoing)*

---

**Automotive CAN communication stack**

A simplified view

- **ECU**
  - **Middleware**
    - **Frame-packing task**
    - **Waiting queue**:
      - FIFO
      - Highest Priority First
      - OEM specific

- **CAN Controller**
  - **CAN Bus**
    - **buffer Tx**
    - **9 6 8**
    - **2000**
Optimizing CAN: meeting performance and robustness constraints at higher load

An industrial requirement
- Reduce architecture complexity, HW costs & weight, consumption and emission
- Avoid industrial risks and costs of new technologies
- Incremental design / better performances

How?
1. Keep amount of data transmitted minimum! → better identification and traceability of timing constraints
2. Synchronize producing tasks with communication tasks
3. Desynchronize frames by using offsets [3,4]
4. Assign priorities according to deadlines
5. Re-consider frame packing [12]
6. Optimize communication stacks so as to remove all "distortions" to the ideal CAN behavior

Scheduling frames with offsets ?!

Principle: desynchronize transmissions to avoid load peaks

Offsets algorithm applied on a typical body network

Worst-case latencies on a 125 kbit/s body network

Let's assume frame waiting queue is FIFO on ECU1, the OEM does not know it or software cannot handle it ...

Many high-priority frames are delayed here because a single ECU (out of 15) has a FIFO queue ... could propagate through gateways

Up to recently [5,6], no response analysis on CAN was published ...
Our work: bridging the gap between (analytic) models and reality

Higher load → less margin → more accurate models

1. Hardware models
2. Software models (producer, sender, receiver, device drivers, etc)
3. Error models (reboot, errors)
4. Traffic models incl. aperiodic

Departure from the ideal CAN behavior
Some reasons

Departure from ideal CAN (1/2)

1. Non-HPF waiting queues [5,6]
2. Frame queuing not done in priority order by communication task

Analyzing communication traces: priority inversion

Priority inversion here (probably) because frames are not queued in the order of priority
Higher load level calls for

1. More constraining specifications / or conservative assumptions → a single node can jeopardize the system

2. Thorough use of Validation & Verification techniques:
   - simulation, analysis and trace inspection
   - none of them alone is sufficient!

Know-how, embedded software, verification techniques, and tool support have progressed to a point where highly loaded CAN networks - yet safe are possible.

References

Software used in this study

- **RTaW-Sim**, Fine-grained simulation of CAN based communication systems with fault injection capabilities, downloadable at http://www.realtimeatwork.com/software/rtaw-sim/
- **RTaW-TraceInspector**, Analyze communication traces and check communication stack implementation and specification compliance, see http://www.realtimeatwork.com/software/rtaw-traceinspector/