Decomposing the Dashboard Example for a Distributed Implementation

Jason Shamberger
EE249 Fall 1999
Mentor: Dr. Alberto Ferrari
Outline:

- Motivation for a distributed system
- CAN specification
- Behavioral modeling
- Mapping to architecture
- Simulation and results
- Synthesis
- Conclusion
Goal of Project

- Evaluate POLIS for designing a distributed real-time system
  - By using POLIS, we can explore different network architectures and then move directly to synthesis
- Motivation: As more electronics are in modern cars, need to use network to share communications
Controller Area Network (CAN)

- Used for communication with real-time constraints
- Small amounts of data - up to 8 bytes of data per packet
- Total packet size up to 120 bits
- Higher priority device always transmits first
Original design of dashboard

Sensor1

Sensor2

Sensor3

Display

All one CPU
New dashboard design

- Sensors
- Communication
- Display

First CPU

Second CPU
Architecture Proposed

CPU1

CAN Cont.

CPU2

CAN Cont.
Communication Refinement
Architecture Mapping
Modeling in POLIS

- CAN controller only stores one packet from each sender
- Give different priorities to different types of data
  - High priority: belt alarm, fuel alarm, water alarm
  - Low priority: speedometer, tachometer, odometer
Modeling in POLIS

- Assume the CAN bus already has traffic on it

- Traffic distribution:
  - High frequency small data (1 packet) at regular intervals
  - High frequency small data (1 packet) with Poisson distribution
  - Low frequency, larger data (multiple packets)
Modeling in POLIS

- Ensure that traffic + dashboard messages do not exceed total bandwidth
- Place the dashboard system on the same CAN bus and see what priority is necessary for dashboard to function satisfactorily
Results of Traffic Modeling

- Assume small delay in data is ok, goal is to avoid losing packets
- Measured number of overwrites in network controller - lost data

<table>
<thead>
<tr>
<th></th>
<th>75% Bandwidth</th>
<th>90% Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Priorities</td>
<td>38 overwrites</td>
<td>341 overwrites</td>
</tr>
<tr>
<td>Engine Speed</td>
<td>0 overwrites</td>
<td>16 overwrites</td>
</tr>
<tr>
<td>Highest Priority</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
For simulation, network interface is modeled by a CFSM which muxes inputs onto network channel

Replace mux with code to call APIs of CAN controller
Conclusion

- POLIS can be used to model a distributed system
- The network can be simulated and design decisions made
- Move directly on to synthesis (hopefully)