An experience report on the integration of ECU software using an HSF-enabled real-time kernel


System Architecture and Networking (SAN)
Department of Mathematics and Computer Science
Eindhoven University of Technology
The Netherlands

7th July 2015

An automotive example

Why this growth of electronic parts?

An automotive example: electronic stability program (ESP)

Increasing number of applications, for example: ABS, TCS, ESP;
Extensive networking between them.

Chip Design Magazine (Jan. 2005)
An automotive example: IVDC advances beyond ESP...

Adding suspension control and software-based vehicle state estimation:


Demo: IVDC with active suspension

Experimental setup:
- 4X Local controllers for active suspension;
- 1X Global IVDC and supervisory control.

Our contribution:
- Virtualization techniques applied to a local control node:
  - Communication between independently developed components.

Demo: active suspension on
Resource sharing across components: a closer look

Tasks may request:
- Access to shared memory:
  - shared buffers;
- Operating-system services:
  - processor scheduling;
  - device drivers.
- Network services:
  - send/receive messages;
  - abstraction of fieldbus technology:
    FlexRay vs. CAN

Challenges for resource sharing across components

- Independent development of components:
  - RAP, Suspension and Supervisor;
- Integration of components:
  - Communication abstraction;
  - Servers as a virtual processor;
- Scheduling components and containment of temporal faults.
Challenges for resource sharing across components

- **Independent development of components:**
  - RAP, Suspension and Supervisor;
- **Integration of components:**
  - Communication abstraction;
  - Servers as a virtual processor;
- **Scheduling components and containment of temporal faults.**

Independent development: Run-Away Process

**Greedy for processor cycles:**

1. `repeat`
2. `; {Do nothing}`
3. `until` received-message "stop"

- Can be started/stopped via network messages;
- Highest priority application;

Purpose: Demonstrate temporal isolation

Independent development: Suspension control

**Hardware for 1 axle:**

- Pump
- Damper valves (CDC valves)
- Accumulator to compensate for volume changes
- Hydraulic actuator (instead of dampers)
- Valve for leveling
- Valves for force control
- Oil tank

**Software per wheel (2 tasks for force control):**

- Current control of valves (400 Hz);
- Pressure control of valves (100 Hz);

Independent development: Supervisory control

**Handle state changes:**

- Fault model:
  - message loss;
  - range check sensor values.
- Formal verification:
  - deadlock avoidance;
  - completeness of actions.

- Code generation:
  - MISRA C compliant;

Purpose: Demonstrate temporal isolation
Challenges for resource sharing across components

- Independent development of components:
  - RAP, Suspension and Supervisor;

- Integration of components:
  - Communication abstraction;
  - Servers as a virtual processor;

- Scheduling components and containment of temporal faults.

Integration of components: overview

Independently developed components, their modules and interfaces:

Dependencies between software modules and components:

Virtualization of:
- the processor (HSF);
- the network (communication stub).

Integration of components on HSF-enabled kernel

Hierarchy of processor schedulers:

- component: server, set of tasks and local (task) scheduler
- server or virtual processor: a CPU budget allocated each period

Tasks, located in arbitrary components, may communicate
Components, servers and tasks

4 servers:
- In descending priority: RAP, suspension, supervisor, idle.

System tasks:
- Initialization task;
- Idle task.

Application tasks:
- Run-Away Process;
- 2x suspension control (400 Hz and 100 Hz);
- Supervisor (100 Hz).

Integration of components: communication abstraction

Time- vs. event-triggered communication:
- Suspension control loops assume timed activation;
- Supervisory control assumes event triggers.

Network technology:
- CAN: event-triggered.
- FlexRay (static segment): time triggered.

Communication abstraction:
- Make all events timed.
- Assumption: application states depend on last event only.

Challenges for resource sharing across components

Independent development of components:
- RAP, Suspension and Supervisor;

Integration of components:
- Communication abstraction;
- Servers as a virtual processor;

Scheduling components and containment of temporal faults.

Current and future challenges:
- Mixed criticality:
  - deal with uncertain timing specs of tasks;
  - graceful degrade functions by enabling/disabling optional ones.

Questions?

Let’s pass the remote . . .