*-aware Software for Cyber Physical Systems

John A. Stankovic
BP America Professor
University of Virginia
Theme

• How can we build practical cyber physical systems of the future?

• 3 Critical (Foundational) Issues: must be addressed together
  - Robustness
  - Real-Time
  - Openness
Foundational Principle

- Scientific and systematic approach for the impact of the physical on the cyber

Propose:
- Physically-aware SW
- Validate-aware SW
- Privacy/security aware SW

Real-time aware
Openness

• Typical embedded systems closed systems design not applicable

• Added value
• Systems interact with other systems
• Evolve over long time
• Physical system itself changes

• High levels of uncertainty: Guarantees
Outline

- Physically-aware software
- Validate-aware software
- Real-Time-aware software
- Privacy-aware software
Physically Aware: Impact of the Physical

• For Wireless Communications (things we know)
  - Noise
  - Bursts
  - Fading
  - Multi-path
  - Location (on ground)
  - Interference
  - Orientation of Antennas
  - Weather
  - Obstacles
  - Energy
  - Node failures
B, C, and D are the same distance from A. Note that this pattern changes over time.
Routing

• DSR, LAR:
  - Path-Reversal technique

Impact on Path-Reversal Technique
Uncertainties - Voids

Destination

VOID

Left Hand Rule
Physically-aware SW

Source
Cyber-Physical Dependencies

- Sensing
  - Sensor properties
  - Target Properties
  - Environmental interference
1. An unmanned plane (UAV) deploys motes

2. Motes establish a sensor network with power management

3. Sensor network detects vehicles and wakes up the sensor nodes

Energy Efficient Surveillance System

Sentry
Tracking

- Magnetic sensor takes 35 ms to stabilize
  - affects real-time analysis
  - affects sleep/wakeup logic

- Target itself might block messages needed for fusion algorithms
  - Tank blocks messages
# Environmental Abstraction Layer (EAL)

<table>
<thead>
<tr>
<th>Wireless Communication</th>
<th>Sensing and Actuation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interference</td>
<td>Target Properties</td>
</tr>
<tr>
<td>Burst Losses</td>
<td>Weather</td>
</tr>
<tr>
<td>Weak Links</td>
<td>Obstacles</td>
</tr>
<tr>
<td>Fading</td>
<td>Wake Up Delays</td>
</tr>
</tbody>
</table>

Not HW-SW co-design, but rather Cyber-Physical co-design
Validate Aware: Run Time Assurance (RTA)

- Safety Critical
- Long Lived
- Validated
- Re-validated

- Dynamics of Environmental Changes Influence Correctness

See Run Time Assurance paper in IPSN 2010.
RTA Goals

• Validate and Re-validate that system is still operational (at semantics level)

• Anticipatory RTA
  - Before problems arise

• Robust to evolutionary changes

Validate-aware software
RTA Solution

- Emulate sensor readings
- Reduce tests to focus on key functionality
- Overlap tests and system operation
- Evolve required tests
Current Solutions

• Prior deployment analysis
  - Testing
  - Debugging
• Post mortem analysis
  - Debugging
• Monitoring low-level components of the system
  - System health monitoring

Necessary, but not sufficient
RTA Framework

Inputs

- Formal application model
- RTA test specifications
- Network database

RTA framework

- Code generation
- Test generation
- Test execution support
Model-based Specification

Sensor Network Event Description Language (SNEDL)

Smoke

Temperature

Smoke > x

Temperature > 30°C

Smoke > 80°C

Fire
//Declare the basic elements of the language
Time T1;
Region R1, R2;
Event FireEvent;

//Define the elements (time and place)
T1=07:00:00, */1/2010;  //first day of month
R1={Room1};
R2={Room2};

FireEvent = Fire @ T1;
Token Flow

Smoke

Temperature

> x

>30°C

>80°C

Smoke alarm

Temp. alarm

Fire
Code Generation

- Code is automatically generated from the formal model

- Advantages of the token-flow model:
  - efficiently supports self-testing at run time
  - it is easy to monitor execution states and collect running traces
  - we can easily distinguish between real and test events
Validate-aware SW

- High level spec on “function”

- Runtime SW that targets demonstrating “validation”

- SW design for ease of validation

- Framework – to load, run, display tests

- System: Be aware of validation mode
Real-Time Aware

- Hard deadlines
- Hard deadlines and safety critical
- Soft deadlines
- Time based QoS

- Dynamically changing platform (HW and SW)
Example: Group Management (Tracking)
Deadlines

• If we have enough late messages within groups we can lose the track
  - Not straightforward deadline
  - Tied to redundancy, speed of target

• If messages don’t make it to base station in hard deadline we miss activating “IR camera”

• If we don’t act by Deadline D truck carrying bomb explodes - safety critical
Real-Time Scheduling

Tasks
1
2
3

Deadlines

Algorithm
EDF

Schedulable
Yes

Order
1, 2, 3

How robust?
$CF = 1$

TIME
Robust RT Scheduling
For Real World CPS

Tasks
1
2
3

Deadlines
(1.8)

Algorithm
EDF

Schedulable
Yes

Order
1, 2, 3

How robust?
1.8 CF
Real-Time Technology

• Three possible approaches
  - Velocity Monotonic
  - Exact Characterization
  - SW-based Control Theory
Feedback Control

- Front-End
  - feedback loops based on real world control
  - generate timing requirements/rates
  - generally fixed
  - handed to scheduling algorithm

University of Virginia
FC-EDF Scheduling

PID Controller

Service Level Controller

Admission Controller

EDF Scheduler

MissRatio_s

MissRatio(t)

MissRatio

Completed Tasks

Accepted Tasks

Submitted Tasks

FC-EDF

Real-Time aware SW

Real-Time aware SW

University of Virginia
Privacy-aware: Fingerprint And Timing-based Snoop attack

Performance

• 8 homes - different floor plans
  - Each home had 12 to 22 sensors
• 1 week deployments
• 1, 2, 3 person homes
• Violate Privacy - Techniques Created
  - 80-95% accuracy of AR via 4 Tier Inference
• FATS solutions
  - Reduces accuracy of AR to 0-15%
• **ADLs inferred:**
  - Sleeping, Home Occupancy
  - Bathroom and Kitchen Visits
  - Bathroom Activities: Showering, Toileting, Washing
  - Kitchen Activities: Cooking

• **High level medical information inference possible**

• **HIPAA requires healthcare providers to protect this information**
Solutions

- Periodic
- Delay messages
- Add extra cloaking messages
- Eliminate electronic fingerprint
  - Potentiometer
- Etc.

Privacy-aware software
Summary

• Robustness - to deal with uncertainties: (major environment and system evolution)
• Real-Time - for dynamic and open systems
• Openness - great value, but difficult

• Physically-aware
• Validate-aware
• Real-Time-aware
• Privacy/security-aware

• Diversity - coverage of assumptions
• EAL

CPS-aware