OS / Middleware for Cyber-Physical Systems

Richard West

Boston University
Boston, MA
richwest@cs.bu.edu

Computer Science

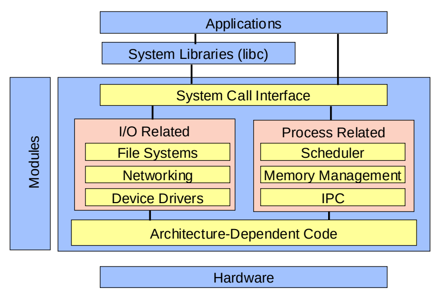
Cyber-Physical Systems (CPS)

- New innovations needed for software infrastructures
  - Communication, computation & physical system aspects to be considered
- Example applications:
  - Coordinated vehicle/traffic management systems
  - Tele-medicine
  - Intelligent homes for appliance management and energy-efficiency (electricity, gas, heating etc)

OS / Middleware Support

- COTS Systems
  - There has been a push for their use for the past 5-10 years to support specialist apps
  - Problems?
    - Semantic gap between app needs and service provisions of system
  - Benefits?
    - Cost savings, code reuse, reduced development time, well-tested basis for new applications/services
  - BUT…
    - Should we continue this path of enhancing COTS systems or are the CPS goals too challenging for existing technologies?

Example System Structure



Current System Problems

- Problems with current systems?
  - Inadequate APIs – application mismatch
  - Agnostic services – e.g., no real-time guarantees when needed, scheduling policies for fairness rather than predictability
  - Inadequate extensibility – geared towards drivers rather than app-specific services
- Need new interfaces to underlying system services to match application demands
  - Possibly retro-fit existing systems with APIs / mechanisms to support **extension technologies**

Challenges

- Cyber-Physical Systems pose challenges in:
  - Design of composable application-specific services that behave safely, securely, efficiently, predictably
  - Design of underlying system / infrastructure to support such services
  - Hardware and software issues affect both the above
    - More on this later…

What about Virtualization?

- Stephen Hand et al (Xen, Cambridge U.) – HotOS paper:
  - Are VMs micro-kernels done right?
  - Right now, virtualization is a means to provide isolation amongst other VMs/apps
    - Useful for legacy systems/apps to co-exist on same physical platform BUT…
      - No significant communication between VMs unlike client/server communication in micro-kernels
      - Coarse-grained solution to safety / security
      - No resource / service guarantees

Basic Goals

- Basic goals:
  - Service composition / customization
  - Safety / security
    - Access rights, capabilities
    - Who should be allowed to deploy services and where?
  - Predictability / efficiency
    - real-time, latency, throughput guarantees etc
    - Resource monitoring, management, QoS
  - Communication protocols
  - System structure
    - API between underlying system and application
    - Interactions between hardware and software
    - Hardware abstraction / heterogeneity

Interactions Between Hardware & Software

- Leveraging architectural features in "best" way, e.g.:
  - L2 shared caches
  - Hyper-threading
  - Multi-core architectures
  - Tagged TLBs for protection
  - Interrupt-vectoring to app-specific trusted services

Heterogeneity

- Physical systems may have diverse computational and resource characteristics
  - Different processor architectures, memory capacities, cache configurations, I/O devices, interconnects
- One vision:
  - Build a base software system deployed across hardware platforms that offers resource multiplexing and communication between higher-level applications/services
  - Have hardware or a software compiler take a common-language (or byte-code) base software and target it for given platform

A Common Platform Alliance

- OS developers provide base code and services in a hardware-independent manner
  - A target compiler for a given platform produces hardware-enhanced binary image of base OS (like a very small microkernel)
  - Additional services are isolated and communicate using "best" approach according to compiler for target platform, the features of that platform and the requirements of services/applications
    - e.g., services may be isolated using hardware segmentation/paging if available, or even compiler generated run-time software checks to enforce memory safety if hardware protection is unavailable

Example: Intelligent Home Network

- www.epa.gov/ne/pr/2004/jan/040110.html
  - Study suggested that by replacing 5 most used light-bulbs w/ energy efficient bulbs in every US household could reduce electricity usage by 800 billion KWh per year
    - Equivalent to $60/yr per homeowner or output from 21 power plants per year
    - Would reduce one trillion pounds of greenhouse gases that cause global warming

Example (continued)

- Intelligent home network could support services to monitor electricity (and other resources e.g., gas) throughout the day
  - Services could suggest ways to more efficiently spread energy usage over 24 hours, rather than at set hours when demand is excessive
    - Over-riding control of appliance usage
    - Possibly enforce resource quota or re-channeling of resource (here, electricity) distribution amongst homes according to a shared service policy
    - GOAL: lowering overall resource consumption while meeting individual objectives

Example (continued)

- Who should be allowed to deploy specific services and where?
  - Perhaps not homeowners except to configure basic parameters of existing services or to upgrade services
  - Service providers could be 3rd parties relative to system developers
- To what extent can users control / influence service provisions to other customers?
  - Perhaps they shouldn't be allowed to do this at all
  - Perhaps they should be allowed to do this to some degree if it is for the global good
    - The socialist view – if I share my resources will you repay the favor when needed?

Vehicle Control / Traffic Management Example

- Coordinated in-vehicle traffic management system
  - Allow in-car services to communicate congestion hot-spots to other vehicles, or even to over-ride user-responsiveness when emergency braking is required etc…

Questions?

- What limitations does the existing (architectural, intellectual etc) separation between X and Y place on our ability to develop CPS? How could we redesign X and Y to remove those limitations…?
  - Mismatch between app-needs and agnostic service provisions
    - TCP, IP networks not real-time, have bandwidth/latency mismatches with certain apps
    - OS services: scheduling, paging misaligned with demands of apps
    - Again, need extensibility here…a breakdown of the barriers between coarse-grained services and components
    - Possibly user-configurable and implementable protocols and services
      - Methods to activate those services in keeping with QoS (real-time, latency etc) requirements
      - Methods to safely and securely isolate X and Y
      - Leverage of hardware features in meeting these goals

Questions? (continued)

- Are there opportunities to co-design, hybridize, or otherwise combine parts of the current state of the art in ways that overcome existing limitations, without requiring us to re-start from too primitive a basis?
  - Could build new base software architecture for safe, predictable and efficient resource multiplexing to higher-level services and VMs
    - Could allow for existing software to run above this base layer
  - Could retro-fit existing systems to support better extensibility for user-configurable services, isolation and invocation
    - Provide improved APIs