CSE 480/CIS 700: Distributed Real-Time and Embedded Systems

Insup Lee
Department of Computer and Information Science
University of Pennsylvania

September 7, 2006

CSE 480/CIS 700, Fall 2006

- Class: Towne 313, 1:30-3 pm, TTh
- Instructors:
  - Insup Lee
    - Email: lee@cis.upenn.edu
    - Office: 602 Levine
    - Office Hours: 4-5 pm, TTh
  - Sebastian Fischmeister
    - Email: sfischme@seas.upenn.edu
    - Office: 264 Levine North
    - Office Hours: TBD
- Lab: 269 GRW (Levine North)
- Home page: www.cis.upenn.edu/~lee/06cse480/

Topics for the course

- How to program an embedded processor
- Temporal Constraints in Programming Languages
- Programming paradigms
  - Execution/Data-flow/Resource Semantics (Esterel, Giotto, RTSJ)
  - Quantum Programming
- RTOS/Middleware
- Real-Time Scheduling
- Embedded Networks, Network Code Machine
- Requirements, Design Specifications, Software Architecture
- Verification, Validation, and Testing

Assignments & Grading

- Expected to do:
  - Lab assignments
  - Homework assignments (e.g., short summaries)
  - Term project
- Possible Projects
  - Programming in ACSR support (a la QP)
  - GBP support on Motes (extend TinyGBP in nesC)
  - Modular robots
  - Real-time software analysis
  - …

Challenges and Opportunities for Embedded Systems

Insup Lee
Department of Computer and Information Science
University of Pennsylvania

September 7, 2005

Example Embedded Systems

Profusion of Embedded Systems

- Two types of computing systems
  - Desktops, servers, clusters, grids
  - Embedded
- The next frontier
  - Mainframe computing (60's-70's)
    - Large computers to execute big data processing applications
  - Desktop computing & Internet (80's-90's)
    - One computer at every desk to do business/personal activities
  - Ubiquitous/physical computing (00's-?)
    - Numerous computing devices in every place/person
    - "Invisible" part of the environment
    - Automobiles, entertainment, communication, avionics, mobile devices, medical devices, sensor nets
- Number of units per year
  - Millions for desktops
  - Billions for embedded processors
- Number of motors in a household (10's) vs. number of embedded systems (100's)
- Applications:
  - Hybrid and embedded systems
    - Aerospace, automobiles, robotics, process control, sensor networks, smart spaces
  - Multimedia
    - Virtual reality, immersive environment, tele-presence
  - Consumer electronics
    - Mobile phones, office electronics, digital appliances
  - Network components
    - Bridges, routers, switches, hubs
  - Medical devices and instruments
    - Patient monitoring, MRI, infusion pumps, artificial organs
  - E-business
    - ATM, vending machines
  - Distributed and grid computing
    - Critical infrastructure defense system, air traffic control, intelligent highway systems, emergence response system
  - Wireless sensor nets
    - Environmental monitoring, agriculture, transportation
  - Etc.

Key Trends and Economic Impact

- Increasing complexity
- Increasing integration and networking interoperability
- Growing importance and reliance on software
- Increasing number of non-functional constraints
- Shortening time to market
- Reuse of existing hardware and software components
- Great variety of component types
- Open standards
- Cost-effective production of dependable, secure, and innovative systems

[EU Study]

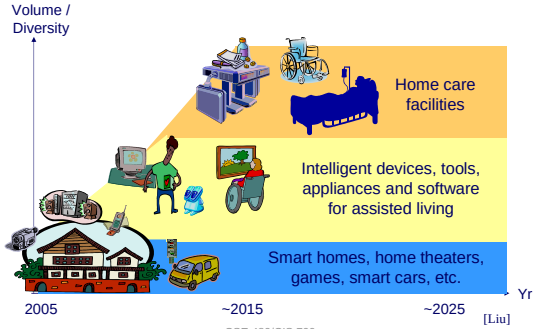
Embedded Software

- Greater impact of information technology on embedded systems.
- Software development accounts for 70-80 % of the overall development cost
- Characteristics
  - Tightly-coupled to the physical world, i.e., interacts with (or reacts to) its environment
  - Correct operation is subject to
    - Physical constraints imposed by the environment
      - Need to operate/adapt/communicate in real-time
    - Resource constraints of the device
      - Space, size, power, memory, weight
  - Heterogeneity, networked at extreme scale
    - Interoperability, stability, fault-tolerance, QoS guarantees
  - Sociological and ethical requirements
    - Usability - users are not system experts
    - Dependability, robustness, and safety
    - Security and privacy.

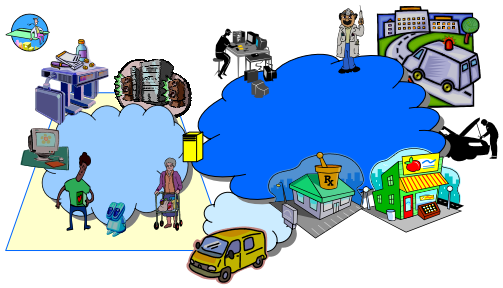
Example: Automotive Telematics

- In 2005, 30-90 processors per car
  - Engine control, Break system, Airbag deployment system
  - Windshield wiper, door locks, entertainment systems
  - Example: BMW 745i
    - 2,000,000 LOC
    - Window CE OS
    - Over 60 microprocessors
      - 53 8-bit, 11 32-bit, 7 16-bit
    - Multiple networks
    - Buggy?
- Problems
  - Disparity between the design cycle of a car and the design cycle of embedded components
  - Difficult to upgrade
  - Not possible to integrate the user's own devices into a car

Example: Home and Personal Appliances



Embedded Home Environment



Embedded Software - Goals

- Trustworthy: should not fail (or at least gracefully degrade), and safe to use. The existence of embedded software becomes apparent only when an embedded system fails.
- Context- and Situation-Aware: should be able to sense people, environment, and threats and to plan/notify/actuate responses to provide real-time interaction with the dynamically changing physical environment with limited resources.
- Seamless Integration: should be invisible at multiple levels of a hierarchy: home systems, metropolitan systems, regional systems, and national systems.
- Validation and Certification: should be able to assure that embedded systems work correctly with respect to functional and nonfunctional requirements with high degree of certainty.

Challenges

- Assuring high confidence
  - Safety, reliability, correctness …
  - QoS such as real-time, performance, security
- Safety-critical computing systems
  - Design errors can be catastrophic
- Design, testing, certification are very expensive
  - Money and time as well as innovation
- Large scale, heterogeneous systems
  - Verified composition for embedded systems
  - Interoperability
- Heterogeneous (or hybrid) systems
  - Integrate computing with physics
- Education
  - Inadequate education in embedded systems
  - inadequate education in system architecture/integration

- Partial List of Embedded System failures
  - Denver baggage handling system ($300M)
  - Power blackout in NY (2003)
  - Ariane 5 (1996)
  - Mars Pathfinder (1997)
  - Mars Climate Orbiter ($125M,1999)
  - The Patriot Missile (1991)
  - USS Yorktown (1998)
  - Therac-25 (1985-1988)
  - London Ambulance System (£9M, 1992)
  - Pacemakers (500K recalls during 1990-2000)
  - Numerous computer-related Incidents wth commer aircraft (http://www.rvs.uni-bielefeld.de/publications/compendium/incidents_and_accidents/index.html)

R&D Needs

- Development of high-confidence software requires
  - Engineering design techniques and tools
    - Modeling and analysis, requirements capture, hybrid systems, testing …
    - Capture and optimization of inter-dependencies of different requirements
    - Domain-specific model-based tools
  - Systems Software and Network Supports
    - Virtualization, RTOS, Middleware, …
    - Predictable (not best-effort) communication with QoS, predictable delay & jitter bounds, …
    - Trusted embedded software components
      - To help structured system design and system development
      - To reduce the cost of overall system development and maintenance efforts
      - To support the reuse of components within product families
  - Validation and Certification
    - Metrics for certification/validation
    - Evidence-based certification, Incremental certification
- Fundamental understanding of
  - Added complexity from interaction with the physical world in real-time under (severe) computing, memory, power and cost constraints
  - Composition and Integration
    - Interoperation
    - Compositional frameworks for both functional, temporal, and non-functional properties
  - Abstraction
    - Virtualization
    - Measures on verification, validation, testing
  - Distributed Sensing & Control
    - Cognition of environment and system state, and closing the loop
    - Graceful adaptation to applications, environments, and resource availability
    - Scalability, reliability, robustness, stability of system of systems