CSE 520S
Real-Time Systems

Prof. Chenyang Lu
TAs: Chong Li, Chao Wang
Real-Time Systems

- Systems operating under timing constraints
  - Automobiles.
  - Airplanes.
  - Mars rovers.
  - Game console.
  - Factory automation.
  - Telecom and mobile network management.
  - Stock trading.
  - Air traffic control.

- >95% of microprocessors are used for embedded systems.
Embedding a Computer

CPU

mem

input

output

actuators

sensors

embedded computer

analog sensors

analog actuators
Anti-lock Brake System

- Pumps brakes to reduce skidding → **real-time** → **safety**
GM Super Cruise

Highway Autopilot

General Motors Super Cruise: John Capp, Jeremy Salinger, Eric Raphael, and team

The self-driving car started as a science-fiction fantasy—then DARPA and Google turned it into a real-world experiment. Now GM is working to make automotive autonomy part of an option package. The automaker has combined two existing technologies—adaptive cruise control and lane centering—into its Super Cruise system, which allows for hands-free driving at highway speeds. It could be available in a production vehicle as soon as 2018.

How It Works

Collision Avoidance

A long-distance radar system detects vehicles more than 300 feet ahead. The vehicle will automatically accelerate or apply the brakes to maintain a preset following distance.

Lane Centering

Using a combination of GPS and infrared optical cameras, the Super Cruise system watches the road ahead and adjusts steering to keep the car in the middle of its lane.
A Distributed Real-Time System

ECU A
- Microcontroller 1
  - Core 1
  - Core 2
- Microcontroller 2
  - Core 1
  - Core 2

ECU B
- Microcontroller 1
  - Core 1
  - Core 2
- Microcontroller 2
  - Core 1
  - Core 2

FlexRay Channel A
- Radar
- Radar
- Camera

FlexRay Channel B
- Radar
- Radar
- Camera

CAN Bus #1
- Brake Controller
- Steering Controller
- Engine Controller
- Transmission Controller

CAN Bus #2

Courtesy: GM
More on a Car

~100 microprocessors:
- 4-bit microcontroller checks seat belt;
- microcontrollers run dashboard devices;
- 16/32-bit microprocessor controls engine;
- Navigation;
- Entertainment: DVD, audio, satellite radio…
Real-Time Applications in a Car

- Soft real-time: Infotainment on Linux or Android
- Hard real-time: Safety-critical control on AUTOSAR

Internet of Things

- Convergence of
  - Miniaturized hardware: processor+sensors+wireless
  - Low-power wireless: connect millions of devices to the Internet.
  - Data analytics: make sense of sensor data.
  - Cloud computing: scalable data processing.

- Real-time monitoring and control of physical systems
  - Smart *: house, healthcare, manufacturing, transportation, grid…

- We are in the **Golden Age** of Internet of Things!
  - “A period in a field of endeavor when great tasks were accomplished.”
Real-Time Cloud

- IoT $\rightarrow$ large-scale sensing and control of physical world
  - Smart manufacturing, smart transportation, smart grid...
  - Feedback control demands real-time performance guarantees.

- Example: Intelligent Transportation
  - Cloud collects data from cameras and roadside detectors.
  - Control the traffic signals and message signs in real-time.
  - Transportation information feed to drivers.
  - SCATS @ Sydney: controlling 3,400 signals at 1s round-trip latency.

- Cloud needs to be much more real-time and predictable than they are today!
Towards Real-Time Clouds

- Support real-time applications in the cloud.
  - Latency guarantees to tasks in virtual machines (VMs).
  - Real-time performance isolation between VMs.
  - Resource sharing between real-time and non-real-time VMs.

- Multi-level real-time performance provisioning.
  - RT-Xen → Real-time VM scheduling within a virtualized host.
  - VATC → Real-time network I/O within a virtualized host.
  - RT-OpenStack → Real-time cloud resource management.

- Make cloud real-time by incorporating real-time scheduling into cloud infrastructure.
Example: RT-Xen

- Real-time schedulers in the Xen hypervisor.
- Provide real-time guarantees to tasks in VMs.
- Incorporated in **Xen 4.5** as the rtds scheduler.


https://sites.google.com/site/real7mexen/
Watch This for Inspiration!

Advancing Real-Time Virtualization, Cloud Computing (HEC-TV):

Challenges

Must meet non-functional constraints

- Real-time
- Memory
- Battery lifetime
- Reliability, safety and certification
- Cost

Correct output is NOT enough!
Real-time Requirements

- **Period**: release a job every $T$ sec
  - Playback 30 video frames per second

- **Deadline**: complete a job within $D$ sec
  - Anti-lock brake must start within 10 ms after skidding starts
Hard vs. Soft Real-Time

- **Hard**: violating timing constraints $\rightarrow$ failure
  - Automobile: active safety features, autonomous driving
  - Air traffic control

- **Soft**: violating timing constraints $\rightarrow$ inconvenience
  - Video
  - Audio ("harder" than video)
  - Stock trading
Topics

- Real-Time Scheduling
  - Single-core and multicore processors, parallel and distributed systems

- Real-Time Platforms
  - Real-time operating systems
  - Real-time middleware

- Real-Time Virtualization and Cloud
  - RT-Xen: real-time virtualization
  - RT-OpenStack: real-time cloud
  - VATC: real-time network I/O

- Adaptive Real-Time Systems

- Real-Time Wireless Sensor-Actuator Networks

- Projects: Cloud for Real-Time and IoT Systems
  - Amazon Web Services and Google Cloud Platform
Grading

- **Projects 60%**
  - Cloud warm-up homework: 1%
  - Proposal and presentation: 10%
  - Demo 1: 5%
  - Demo 2: 5%
  - Final demo & report: 39%

- **Critiques 35%**

- **Participation 5%**
Critiques

- 1/2 page critiques of research papers
- Submit by 10am before class
- Back-of-envelop comments - NOT whole essays
- See guidelines on class web site:
  - [http://www.cs.wustl.edu/~7Elu/cse521s/critique.html](http://www.cs.wustl.edu/~7Elu/cse521s/critique.html)
Project

- Three students per team

- System projects
  - Develop real-time software
  - Experiment, measure and analyze
  - Write a paper
  - Demo to the class
Project Topics

- **Theme:** *Cloud for real-time and IoT systems*

- **Target:** Platform as a Service (PAAS) of Amazon and Google
  - Streaming processing
  - Messaging service
  - IoT cloud services

- **Experimental methodology**
  - Develop applications and benchmark programs
  - Measure in experiments
  - Analyze and present results

- **Focus on latency and IoT-like workloads**

- **Attend Chong and Chao’s lectures on Wed and Mon!**
Steps

1. Choose your favorite topic
2. Form a team
3. Propose a plan
4. Implement
5. Measure and analyze
6. Demo: 1, 2, final
7. Write a technical report
Start Early and Work Often!

- Choose topics
- Put together a team
- Meet every week to coordinate
- Lots of development and experiments throughout the semester!
Coming Up

- This Wednesday: project topics discussion.

- Next Monday: no class (Labor Day).

- Next Wednesday: Amazon and Google cloud tutorials.
Pointers

- http://www.cse.wustl.edu/~lu/cse520s/

Email for appointment
- Chenyang (Jolley 213)
- Chong Li (Jolley 217): AWS
- Chao Wang (Jolley 507): Google Cloud