EE 579: Wireless and Mobile Networks
Design & Laboratory

Lecture 6

Amitabha Ghosh
Department of Electrical Engineering
USC, Spring 2014

Lecture notes and course design based upon prior semesters taught by Bhaskar Krishnamachari and Murali Annavaram.
Outline

- Administrative Stuff
- Summer Internship Announcement
- Wireless Sensor Networks Overview
- Contiki Operating System
- Hands-on with Tmote Sky (Hello World, Broadcast)
Summer Internship

- Startup looking for good students with expertise in application, middleware, and engineering solutions for mobile platforms

- Possibility of grant/funding to develop some engineering solutions, for example, connecting different health monitoring sensors to mobile platforms

- Contact: Professor Raghu Raghavendra (raghu@vsoe.usc.edu)
Summer Internship

- Profile I
  - Independently design and develop mobile applications for Android platforms
  - Integrate health monitoring sensors to mobile platforms
  - Implement complex and responsive user interfaces and utilizing native frameworks for animations
Summer Internship

- **Profile II**
  - Web-based software development experience
  - Experience with Java web framework such as Spring, Grails
  - Knowledge of front-end Web technologies
    - HTML, CSS, Javascript, AngularJS, Node.js, Backbone.js
  - Understanding of data modeling and database technologies (MySQL and Mongo DB)
  - Experience with REST design concepts
  - Familiarity with version control and configuration management
  - Ability to work independently with little guidance

- **Profile III**
  - Interest in doing engineering work to integrate various sensors for health monitoring to mobile platform
Sensor Networks: The Vision

- The “many - tiny” principle: wireless networks of thousands of inexpensive miniature devices capable of computation, communication and sensing

- For smart spaces, environmental monitoring, battlefield applications...

From Pister et al., Berkeley Smart Dust Project
Tomorrow’s Devices

From Pister et al., Berkeley Smart Dust Project
Tomorrow’s Devices

From Manges et al., Oak Ridge National Laboratory, Instrumentation and Controls Division
WSN Devices

WINS (Rockwell)

MICA 2 Mote (Berkeley)

GNOMES (Rice)

MANTIS Nymph (Colorado)
Basic WSN Hardware

Memory/Storage → Processor → Radio Transceiver → Sensors → GPS

Power Source: Battery/Harvest
Moteiv Tmote Sky
Tmote Sky Features

- 2.4 GHz, 250 Kbps IEEE 802.15.4 CC2420 radio, range: tens of meters
- MSP430 microcontroller, 10 KB RAM, 48 KB flash
- 1 MB external flash
- USB programming using NesC/TinyOS/Contiki
- On-board Humidity, Temperature, and Light Sensors
- Power consumption @ 3V: mcu + radio: ~20 mA, mcu alone: ~2 mA, standby: 20 µA
Applications of Interest

- Seismic Sensing and Actuation
- Structural Condition Monitoring

From CENS
Applications of Interest

- Monitoring ecosystems and species habitats
Applications of Interest

- Contaminant Flow
- Chemical Leaks
- Forest Fires
- Emergency Response

Images from Google
Applications of Interest

• Target Tracking

From 29 Palms Demo, UC Berkley and others
WSN Companies

Hit the Ground Running with a Complete Solution
The only complete standards-based solution for embedded wireless networking
Ember's standard development pathway makes it easy.

Millennial Net delivers the tools and technologies that help our customers
build self-organizing wireless sensor networks using our
i-Bean® Endpoints, Routers and Gateways.

LATEST NEWS
Millennial Net Wins Sensors Expo Award
Sensor Network Management Software Released
Millennial Net Closes $15 Million Series B Funding Round
Motorola Labs is developing Wireless Local Area Network (WLAN) technology to deliver untethered information access in the home, business, vehicle and for "hot spot" areas (relatively distinct, highly concentrated areas such as business and university campuses).
ENABLE AUTOMATED AWARENESS

And effortlessly obtain more comprehensive data than you ever thought possible while slashing the cost of industrial monitoring.


Learn More
Challenges

- Unattended, ad-hoc deployment
- Energy scarcity
  - Radio communication is 100 to 10000 times more expensive than computational processing
- Large Scale: thousands of nodes (millions?!) 
- Distributed data
- Heterogeneous capabilities
- Faulty/failed nodes, noisy measurements
- Dynamic, uncertain environment
- Potentially demanding real-time constraints
Contiki OS

- Open source - BSD license
  - Multitasking using C programming language
- Developed by Adam Dunkels at the Swedish Institute of Computer Science
  - Version 1.0 released in March 2003
  - Version 2.7 released November 15, 2013
- Highly portable
  - Tmote Sky, JCreate, TelosB, Atmel Raven, MicaZ, ...
  - Simulators: Cooja, MSPsim, AvoraZ, netsim
  - Native platform
- Actively developed
  - 17 developers from SICS, SAP, Cisco, NewAE, TU
Contiki as a Research Theme

- Exploring successful computer science abstractions and mechanisms for sensor networks
  - Dynamic module loading and linking [ACM SenSys 2006]
  - File System [IEEE/ACM IPSN 2009]
  - Multi-threaded programming [EmNets 2004]
  - Java, scripting, ... [ACM SenSys 2006, ...]
  - Interactive network shell

- Pursuing new abstractions
  - Protothreads [ACM SenSys 2006]
  - Low-power radio networking [ACM SenSys 2007]
  - Power profiling [EmNets 2007]
  - Novel communication primitives
Features

- Multitasking kernel
- Preemptive scheduling
- Managed memory allocator
- Protothreads
- TCP/IP networking, including IPv6
Coffee File System [IPSN 2009]

- Flash-based file system
- open(), read(), seek(), write(), close()
- Constant memory complexity
- Very lightweight
  - 5 Kb ROM
  - < 0.5 Kb RAM
- Very fast
  - More than 92% of raw flash throughput
Interactive Shell

- Network debugging, performance tuning
- Leverage UNIX-style pipelines
- Network commands
- Direct serial connection, or over Telnet/TCP
- A generic interface for higher level applications
  - Automated interaction, scripting
Power Profiling [EmNets 2007]

- Software-based
  - Zero-cost hardware
  - Zero-effort deployment

- Good accuracy, low overhead

- Enables network-scale energy profiling

- Enables energy-aware mechanisms
Linear Current Draw
Rime: Communication Primitives

- Makes implementation of sensor network mechanisms easier

- A set of protocols
  - Data collection
  - Data dissemination
  - Unicast multi-hop routing
  - Single-hop bulk transfer
  - ...

Contiki IP Architecture
Run Contiki on Hardware

- Contiki has a build system that is intended to make it easy to run Contiki directly on hardware

- Build system is same across different hardware platforms, so that the build commands are familiar when switching hardware

- Comprises a set of makefiles
  - Base makefile: contiki/Makefile.include
  - Platform makefiles in
    - contiki/platform/*/Makefile.platform
    - contiki/cpu/*/Makefile.cpu
Getting Started

- **Step 1: Download Instant Contiki**
  - Contiki development environment - single-file download
  - Ubuntu Linux virtual machine with all development tools, compilers, and simulators installed
  - [www.contiki-os.org/start.html](http://www.contiki-os.org/start.html)

- **Step 2: Download VMWarePlayer**
  - [www.vmware.com/go/downloadplayer](http://www.vmware.com/go/downloadplayer)

- **Step 3: Start Instant Contiki**
  - Open the Instant Contiki folder and execute
  - `instantContiki2.6.vmx`
  - Wait for the virtual Ubuntu Linux to boot up
Getting Started

- Step 4: Log in to Instant Contiki
  - Username: user
  - Password: user

- Step 5: Compile and run “Hello World” on native platform
  - `cd contiki-2.x`
  - `cd examples/hello-world`
  - `make TARGET=native`
  - `./hello-world.native`
  - Should print “Hello, world” on your screen and continue to hang - hit ctrl+c to quit
Run Contiki on Hardware

- **Step 1:** Open a terminal, go to the code directory
  - `cd contiki/examples/hello-world`

- **Step 2:** Compile Contiki and the application
  - Compile Hello World for our hardware platform
  - Also compiles the entire Contiki system (take some time)
  - `make TARGET=sky hello-world`

- **Step 3:** Upload Contiki to the hardware
  - Done with the special `%.upload` maketarget
  - `make hello-world.upload`

- **Step 4:** Check the serial port
  - `make login`
Contiki Mote Shell

- An interactive on-mote shell that provides a set of commands for interacting with the system

- Can be accessed over a serial USB connection, or over a network using Telnet

Run over a USB serial connection
- Compile and upload the shell
  - `cd contiki-2.x/examples/sky-shell`
  - `make sky-shell.upload`

To connect over the USB port
- `make login`
- Next, we will try a few shell commands
Contiki Mote Shell

- To get a list of available commands
  - `help`

- Try other commands
  - `sense | senseconv`
  - `power | powerconv`
  - `ls`
  - `format`
  - `echo test | write file`
  - `ls`
  - `read file`
  - `nodeid`
  - `blink 10`
  - `reboot`
The Power Command

- Prints the current power profile from Contiki’s software-based power profiler

- For decimal digit output
  - `power | binprint`
  - Example output
    
    12  236 0 37421 0 4 0 380 0 0 0 380 0

- Output of power command can be used to compute an estimate of the mote’s power consumption by multiplying the time with pre-measured current draw metrics
Run Contiki in Cooja Simulator

- Cooja - a network simulator

- Nodes can be either of three classes
  - Emulated nodes - entire hardware of each node is emulated (slower but allows precise inspection of the system behavior)
  - Cooja nodes - contiki code for the node is compiled and executed on the simulation host
  - Java nodes - behavior of the node must be re-implemented as a Java class

- A single Cooja simulation may contain a mixture of nodes from any class
Run Contiki in Cooja Simulator

- Open a terminal window and go to the Cooja directory
  - `cd contiki/tools/cooja`
  - `ant run`
  - Wait for Cooja to start

- When Cooja first starts, it will first compile itself (takes some time)
Comparison

Tiny OS
- Event-driven OS with multitasking
- Completely non-blocking
- Programs are built out of software components
- Tasks are non-preemptive and run in FIFO order
- Static linking
- Written in NesC (Networked Embedded Systems C)

Contiki OS
- Event-driven OS with multitasking
- Optional preemptive multitasking
- Dynamic linking
- Written in C