Contiki Programming Course: Hands-On Session Notes

Fredrik Österlind, Adam Dunkels
fros@sics.se, adam@sics.se
Swedish Institute of Computer Science

Siena, July 2009

1 Introduction

Welcome to this Contiki programming course! Contiki is a state-of-the-art, open source operating system for sensor networks and other networked embedded devices [3]. Contiki was the first operating system for sensor networks to provide TCP/IP communication (with the uIP stack) [1, 6], loadable modules [2], threading on top of its event-driven kernel [3], protothreads [5], protocol-independent radio networking (with the Rime stack) [7], cross-layer network simulation (with COOJA) [8], and software-based power profiling [4]. Recent features include a networked shell and the memory efficient flash-based Coffee file system [10]. In this course, we use the Rime stack to communicate in a network of Tmote Sky boards and the network shell to interact with the network. The course ends with a challenging exercise: we will collaboratively build a neighbor discovery application based on RSSI measurements.

These notes describe the practical, hands-on session with Tmote Sky boards and the COOJA simulator. The purpose of this session is to get experience with how to use Contiki with actual hardware and in simulation. We hope that this experience will be of help when later working with Contiki.

We use the Instant Contiki development environment in the exercises. Instant Contiki is a single-file download that contains the Contiki source code and all necessary compilers and tools required for developing software for Contiki. The Instant Contiki environment is a Ubuntu Linux installation that runs within the VMware Player virtual machine execution environment. VMware Player is available for free at the VMware website.

In this course we use Instant Contiki 2.3, with the addition of a few Contiki programs used in the final exercise. The initial exercises, however, are based on standard Contiki example programs already included in Contiki 2.3. These illustrate how to communicate between Contiki nodes and how to use the Contiki shell.

Figure 1: A Tmote Sky board.

These notes are structured as follows. Section 2 describe the pre-course requirements and Section 3 how to get started at the course. Section 4 contains step-by-step instructions for the Tmote Sky programming exercises. Section 5 contains instructions for the final neighbor exercise. Section 6 contains information on how to simulate Contiki programs using MSPSim and COOJA, which can be useful for debugging or understanding network behavior throughout the course. Finally, Section 7 concludes this course.

2 System Requirements

For this course, you need a PC that can run Instant Contiki (VMWare) and access to a Tmote Sky board [9] (Figure 1). We have confirmed that Instant Contiki runs in the VMWare Player, which is available free of charge from the VMWare website, under Windows and Linux, and with VMWare Fusion, which costs money, under Mac OS X. We have brought a few Tmote Sky boards to the course.

2.1 Download, Install Tools

We will bring copies of Instant Contiki with the additional Contiki exercise programs to the course. Downloading Instant Contiki prior to the course is hence optional. Instant Contiki and other required tools can be downloaded from the Contiki website: http://www.sics.se/contiki/instant-contiki.html:

- VMWare Player
- FTDI Driver
- Instant Contiki

Note that the FTDI driver must be installed before plugging in any Tmote Sky boards.

2.2 Exercise Software

The exercise program used in the course are already installed in the provided Instant Contiki 2.3. The Contiki exercise programs are also available for download:

http://www.sics.se/~fros/siena09.zip

The zip-file contains a single directory called siena09. The contents should be extracted into the Contiki examples directory, thus creating the directory:

```
contiki-2.x/examples/siena09
```

3 Getting Started

Before starting with the actual exercises, make sure that your development setup works by conducting the steps below.

3.1 Start Instant Contiki

Open the file

```
instant-contiki.vmx
```

to start VMware and Instant Contiki.

3.2 Log In

When the login screen appears, log in to Instant Contiki:

- Username: **user**
- Password: **user**

3.3 Open a Terminal Window

After logging in, click on the terminal icon to start a terminal window.

3.4 Compile, Run Hello World

In the terminal window, go to the hello world example directory, and compile for the native platform:

```
cd contiki-2.x_siena09
cd examples/hello-world
make TARGET=native
```

Wait for the compilation to finish. Run the Hello World program in Contiki:

```
./hello-world.native
```

The program should print the words “Hello, world” on the screen and then appear to hang. In reality, Contiki is still running correctly, but will not produce any output because the Hello World program has finished. Press Ctrl+C on the keyboard to quit.

3.5 Connect the Tmote Sky

Put a Tmote Sky in the computer’s USB port. The Tmote Sky will appear in the top of the Instant Contiki (VMware Player) window with the name “Future Technologies Device”. Click on the name to connect the Tmote Sky to Instant Contiki.

3.6 Run Hello World on the Tmote Sky

Compile and upload the Hello World program on the Tmote Sky:

```
make hello-world.upload TARGET=sky
```

Wait for the compilation and uploading procedure to finish. During the uploading the Tmote Sky should quickly flash the red LEDs next to the USB connector.

Connect to the USB port to view the program output:

```
make login TARGET=sky
```

Press the reset button on the Tmote Sky and a message similar to the following should appear:

```
Contiki 2.3 started. Node id is set to 1.
Rime started with address 1.0
MAC 00:12:75:00:11:6e:cd:fb X-MAC channel 26
Starting 'Hello world process'
Hello, world
```

The Contiki boot-up code prints the first four lines, and the Hello World program prints out the last line. Press Ctrl+C to quit.

4 Tmote Sky Exercises

The exercises consist of running and altering a set of programs that use the Rime stack to communicate with other Contiki nodes over the radio. All programs run on the Tmote Sky boards. The purpose of the exercises is to play around with Contiki network programming.

For information on how to simulate these programs, see Section 6.

4.1 Broadcast

The first exercise consists of sending your name to a base station with broadcast packets. This exercise shows how to use the broadcast communication primitive in the Rime stack and how to put data in packets.

The program to use reside in the directory examples/rime. Go there directly by typing

```
cd ..
cd rime
```

The example program sends broadcast packets with a random interval between 2 and 4 seconds. All nearby Tmote Sky boards will receive the packet.

Open the file example-abc.c by typing:

```
gedit example-abc.c &
```

Go down to the line that contains

```
packetbuf_copyfrom("Hello", 6);
```

Change this line so that the string contains your name or email address. The second argument is the number of characters in the name string, including the terminator character.

Compile and upload the broadcast program to your Tmote Sky: in the terminal window, type:

```
make example-abc.upload TARGET=sky
```

This will compile and upload the program to the Tmote Sky connected to your PC. This takes some time the first time because the entire Contiki operating system is compiled.

When the compilation and uploading has finished, watch your name appear on the projector screen.

To see what broadcast messages your Tmote Sky see, type the following in the terminal window:

```
make login TARGET=sky
```

This shows all serial output that the Tmote Sky is sending over the USB port. To stop, press Ctrl+C.

Go through the code in the example-abc.c file to see how it works. If you have any questions about how it works, don't be afraid to ask!

4.2 Unicast

The unicast program looks similar to the broadcast program, however, a destination address is needed.

Open the file example-unicast.c. Locate the lines that contain the following:

```
    packetbuf_copyfrom("Hello", 5);
    addr.u8[0] = 41;
    addr.u8[1] = 41;
```

As in the first exercise, you should change "Hello" to your name or email address. Next, make sure that the unicast packets go the to right address. The program is configured to send data to node 41.41. You need to change this to the address of the projector screen node. You may also send the packets to neighboring course participants.

Compile and upload the program:

```
make example-unicast.upload TARGET=sky
```

See your name appearing on the projector screen when the compilation and uploading has finished.

Go through the example-unicast.c file to see how it works. Note the differences from example-abc.c. Feel free to ask questions!

To see what your Tmote Sky sees, run:

```
make login TARGET=sky
```

Press Ctrl+C to exit.

4.3 Data dissemination

In this exercise, we will disseminate data from a single node to all other nodes.

Open the file example-trickle.c. As in the previous exercises, you may change the data payload to your name. No destination address is needed here as the trickle dissemination protocol will send your data to all other nodes.

Compile, upload and login to the node:

```
make example-trickle.upload TARGET=sky
make login TARGET=sky
```

Your message is transmitted when the on-board user button is pressed.

Press Ctrl+C to exit.

4.4 Shell

The purpose of the shell exercise is to see how the Contiki shell works. The Contiki shell is a powerful application that makes it easy to interact with both individual nodes and a network of nodes.

Compile and upload the shell program:

```
cd ..
cd sky-shell
make sky-shell.upload
```

When compilation and uploading finishes, login to your Tmote Sky:

```
make login
```

You can now enter commands to the shell via the keyboard. help lists all installed commands, try them and play around:

```
help
sense | senseconv
ls
format
echo test | write file
ls
read file
nodeid
blink 10
reboot
repeat 2 2 { echo again } &
ps
```

Exit with Ctrl+C.

5 Neighbor Discovery Exercise

The neighbor discovery exercise consists of constructing a Contiki application that samples the on-board light sensor, and broadcasts the light measuments. In addition, it should also listen for light measurements from neighboring nodes, and sort all neighbors by their associated Received Signal Strength Indicator (RSSI). Finally, the three best neighbors (those with the highest RSSI values) should be reported to a sink node. The sink node is connected to a projector, and displays the overall progress on the projector screen.

This exercise involves understanding existing examples, and combining them to build a complex Contiki application. In addition, it also illustrates how well radio RSSI measurements are for differentiating between good and bad neighbors.

Go to the siena09 directory with the exercise skeleton code and examples:

```
cd contiki-2.x_siena09
cd examples/siena09
```

Below follows short descriptions about the provided Contiki code.

5.1 Skeleton: report-top3.c

A Contiki program with two processes: a light broadcasting process, and a neighbor reporting process. You may use this program as skeleton code when building the full application.

5.2 RSSI sampling: ex-broadcast-rssi.c

Example code that broadcasts packets using Rime, and prints the RSSI values of received radio packets.

5.3 Light sampling: ex-light.c

Example code reading from the Tmote Sky on-board light sensor.

5.4 Neighbor library: myneighbors.c

Simple neighbor library implementation that sorts neighbors according to their RSSI values. If you finish the exercise early, you may try to improve the implementation of this library, for example by keeping a moving average of RSSI values.

5.5 Neighbor library: ex-neighbors.c

Example code using the above neighbor library.

5.6 Packet structures: packets.h

Contains the packet structures to use, and the Rime channel info.

Note the simulation files (.csc) in the directory. These may be helpful for understanding and testing your programs. For example, to simulate the light sampling Contiki application:

```
make ex-light.csc TARGET=cooja
```

To solve the task you may need to help each other. Also, feel free to ask for help.

Good luck!

6 Simulating Contiki

This section explains how to simulate Contiki programs in the MSPSim emulator and in the COOJA simulator. MSPSim is an MSP430 microcontroller emulator. COOJA is a network simulator, and uses MSPSim to simulate networks of Tmote Sky nodes.

6.1 Start COOJA

Go to the COOJA directory, compile and start COOJA:

```
cd contiki-2.x_siena09
cd tools/cooja
ant run
```

COOJA compiles, and after a few seconds the simulator appears. All COOJA simulations are controlled using plugins: small Java programs that interact with simulations and simulated nodes. When COOJA is started, no simulation is loaded and no plugins are started.

6.2 Create a Simulation

A new simulation is created via the menu.

- Click menu item: **File**, **New Simulation**.

A number of configuration options are presented. Feel free to ask if you have any questions.

- Enter a **Simulation title**, and click **Create**.

We have now created our first simulation in COOJA. However, the simulation does not contain any nodes yet. To add nodes we need to first create a node type, and then add nodes to the simulation.

6.3 Create a Node Type

Any simulated node in COOJA belongs to a node type. The node type determines, among others, which Contiki applications to simulate. The node type also determines whether nodes are simulated or emulated.

- Click menu item: **Mote Types**, **Create mote type**, **Sky Mote Type**.

You have selected to emulate Tmote Sky nodes, and now need to select what Contiki program to simulate.

- Enter a **Description**
- Click **Browse**, and navigate to **examples/rime/example-abc.c**
- Click **Compile** to start compiling the Contiki program

When the compilation finishes:

- Click **Create**.

We have now created a simulation with a single node type. Before finally starting to simulate, we need to add nodes belonging to this node type.

6.4 Add Simulated Nodes

A dialog allowing you to add nodes has appeared. This dialog can later be accessed via:

- Menu item: **Motes**, **Add motes of type**, [your type description].

Add 5 nodes:

- Enter **5**, and press **Create and Add**.

Five nodes are added to the simulation, randomly located in the XY-plane.

6.5 Start Simulating

A number of plugins are automatically started. These, and more plugins, can be accessed via the plugins menu:

- Menu items: **Plugins, Log Listener** and **Plugins, Simulation visualizer**.

The Log Listener plugin listens to the serial ports of all nodes. Select the Log Listener and press the F1 key for help.

The Simulation Visualizer shows the positions of the simulated nodes as viewed from above. Enable the different visualizer skins by mouse clicking the upper part of the plugin. Mouse drag and drop nodes to change their positions.

In the Control Panel:

- Click **Start** to start the simulation.

Note the node serial data appearing in the Log Listener.

6.6 Save, Load and Reload

COOJA allows for saving and loading simulation configurations. When a simulation is saved, any active plugins are also stored with the configuration. The state of a current simulation is however not saved; all nodes are reset when the simulation is loaded again. To save your current simulation:

- Click menu item: **File, Save simulation**.

Simulations are stored with the file extensions ".csc". To later load a simulation:

- Click menu item: **File, Open simulation, Browse...**. Select a simulation configuration.

When a simulation is loaded, all simulated Contiki applications are recompiled.

A functionality similar to saving and loading simulations, is *reloading a simulation*. Reloading can be used to reset the simulation – to restart all nodes. More importantly, reloading a simulation will recompile all Contiki code, useful while developing Contiki programs. To reload your current simulation:

- Click menu item: **File, Reload simulation**, or press **Ctrl+R**

6.7 Quick-start Simulations

To quickly start simulating Contiki programs, both MSPSim and COOJA can be quickstarted from a Contiki working directory.

Go to the Rime examples directory:

```
cd contiki-2.x_siena09
cd examples/rime
```

To simulate example-abc.c in standalone MSPSim:

```
make example-abc.mspsim TARGET=sky
```

To create a COOJA simulation with example-abc.c:

```
make example-abc TARGET=cooja
```

Note that with this command, the nodes will be simulated at the operating system level, they will not be emulated using MSPSim.

To load a COOJA simulation configuration residing in the same directory:

```
make mysimulation.csc TARGET=cooja
```

7 Conclusions

This course is an introduction to Contiki programming – use it as a starting point for further exploration. Copy the examples and the exercise code in your own projects: it is a very good way to learn. Play around with COOJA to get a feeling for how it works.

Always feel free to ask us questions! Use the mailing list for best responses as more people will see the questions and have a chance to answer.

For more information, visit the Contiki web site: http://www.sics.se/contiki/

References

[1] A. Dunkels. Full TCP/IP for 8-bit architectures. In *Proceedings of The First International Conference on Mobile Systems, Applications, and Services (MOBISYS '03)*, May 2003.

[2] A. Dunkels, N. Finne, J. Eriksson, and T. Voigt. Run-time dynamic linking for reprogramming wireless sensor networks. In *ACM Conference on Networked Embedded Sensor Systems (SenSys 2006)*, Boulder, USA, November 2006.

[3] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Workshop on Embedded Networked Sensors*, Tampa, Florida, USA, November 2004.

[4] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the Fourth Workshop on Embedded Networked Sensors (Emnets IV)*, Cork, Ireland, June 2007.

[5] A. Dunkels, O. Schmidt, T. Voigt, and M. Ali. Protothreads: Simplifying event-driven programming of memory-constrained embedded systems. In *Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems (SenSys 2006)*, Boulder, Colorado, USA, November 2006.

[6] A. Dunkels, T. Voigt, and J. Alonso. Making TCP/IP Viable for Wireless Sensor Networks. In *Proceedings of the First European Workshop on Wireless Sensor Networks (EWSN 2004), work-in-progress session*, Berlin, Germany, January 2004.

[7] A. Dunkels, F. Österlind, and Z. He. An adaptive communication architecture for wireless sensor networks. In *Proceedings of the Fifth ACM Conference on Networked Embedded Sensor Systems (SenSys 2007)*, Sydney, Australia, November 2007.

[8] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*, Tampa, Florida, USA, November 2006.

[9] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proc. IPSN/SPOTS'05*, Los Angeles, CA, USA, April 2005.

[10] N. Tsiftes, A. Dunkels, Z. He, and T. Voigt. Enabling Large-Scale Storage in Sensor Networks with the Coffee File System. In *Proceedings of the 8th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN 2009)*, San Francisco, USA, April 2009.