Contiki Programming Course: Hands-On Session Notes

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

October 2008

1 Introduction

Welcome to this Contiki course and the hands-on session! The purpose of the course is threefold: to understand Contiki, to see how it can be used to write systems papers, and to meet others who are using Contiki. These notes and the slides presented during the course should be saved for future reference.

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. 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.

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.

Figure 1: A Tmote Sky board.

In this course, we use Instant Contiki 2.2.1 and an updated version of Contiki 2.2.1 that has a number of bugfixes from the current CVS version, as well as a set of Contiki tutorial programs that we run on the Tmote Sky boards during the course. The programs illustrate how to communicate between Contiki nodes and how to use the Contiki shell.

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 and Section 5 the instructions for the Cooja exercises. Section 6 concludes.

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

Prior to the course, download and install the following tools (all available 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 Start Instant Contiki

Open the Instant Contiki folder, and open the file

```
instant-contiki.vmx
```

to start VMware and Instant Contiki.

2.3 Log In

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

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

2.4 Download Exercise Software

Open the Firefox web browser by clicking the Firefox icon on the taskbar. Enter the following URL in the location bar:

http://www.sics.se/contiki/ckth08.tar.gz

Open the file with Archive Manager and extract contiki-kth08 in the directory /home/user/.

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

If you have not started Instant Contiki, do so by opening 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 loggin 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, compile for the native platform, and run:

```
cd contiki-kth08
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 Upload Blink

To check that the Tmote Sky is correctly connected to the computer and Instant Contiki, compile and upload the blink program to the Tmote Sky:

```
make TARGET=sky blink.upload
```

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. After uploading finished, the blink program will start to run and flash the three blue-red-free LEDs.

3.7 Run Hello World on the Tmote Sky

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

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

After the compilation and uploading has finished, connect to the USB port to view its output:

```
make TARGET=sky login
```

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

```
Contiki 2.2.1 started. Node id is set to 76.
Rime started with address 76.0
MAC 00:12:75:00:11:6e:cd:fb
Starting 'Hello world process'
Hello, world
```

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

4 Tmote Sky Exercises

The exercises consist of 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.

The exercise programs reside in a directory called examples/tutorials. Go to the directly by typing

```
cd ..
cd tutorials
```

4.1 Broadcast

The first exercise is about 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 exercise program sends broadcast packets with a random interval between 20 and 40 seconds. All nearby Tmote Sky boards will receive the packet.

Open the file tutorial-broadcast.c by typing:

```
emacs tutorial-broadcast.c &
```

Go down to the line that contains

```
rimebuf_copyfrom("Change me", 10);
```

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.

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

```
make tutorial-broadcast.upload
```

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 in between 20 to 40 seconds.

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

```
make login
```

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 tutorial-broadcast.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 purpose of the unicast exercise is to see the unicast communication primitive, see how to get data from the Tmote Sky on-board sensors, and see how to convert the sensor data to form a string message.

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

```
len = snprintf((char *)rimebuf_dataptr(),
      RIMEBUF_SIZE,
      "Change me: %d.%dC %d%% %d.%dV",
```

Change the "Change me" to your name or email address. Next, make sure that the unicast packets go the to right address. Locate the following lines:

```
  addr.u8[0] = 76;
  addr.u8[1] = 0;
```

These lines are pre-configured to send data to node 76.0. If we have another node connected to the projector screen, you will need to change the address here.

Compile and upload the program:

```
make tutorial-unicast.upload
```

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

Go through the tutorial-unicast.c file to see how it works. Feel free to ask questions!

Finally, edit the receiver address so that all packets go to some one else's Tmote Sky board, recompile and upload the program. To see what your Tmote Sky sees, run:

```
make login
```

Type ctrl-C to exit.

4.3 Mesh

The purpose of the mesh exercise is to see the mesh communication primitive, to see how to transmit binary data in packets, how to obtain power profiling information, how to interact with the on-board button on the Tmote Sky, and how to use the time-synchronized timer.

The mesh primitive is a multi-hop unicast message service that automatically discovers a route to the destination address.

Open the tutorial-mesh.c file and locate the line that contains:

```
strncpy(m->name, "Change me", sizeof(m->name));
```

Change the "Change me" to your name or email address. Locate the lines:

```
addr.u8[0] = 76;
addr.u8[1] = 0;
```

As with the unicast program, these lines may need to be changed so that they contain the correct address of the Tmote Sky connected to the projector screen.

Compile and upload the mesh program to your Tmote Sky board:

```
make tutorial-mesh.upload
```

When compilation and uploading has finished, press the "User" button on your Tmote Sky. This will send a packet to the node connected to the projector screen. The first packet will most likely be lost because it takes a while to set up the route. Press the button again in a few seconds. Try it again if the packet does not reach the receiver.

In the simulation exercises, you will see exactly why it takes so long to get the first packet through.

Once you have gotten the first packet through, the following packets usually are quick to be received.

The receiver will print out the received data, as well as the RSSI and LQI of the received packet. Try to place your hand over the antenna on your Tmote Sky when you press the button. Does the RSSI change? Does the LQI change? Can you manage to get a multihop route? You may need to reset your Tmote Sky to clear its routing table.

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:

```
make tutorial-shell.upload
```

When compilation and uploading finished, log into your Tmote Sky:

```
make login
```

The tutorial-shell program installs two new commands in the shell: hello-world and tutorial-broadcast. Try them:

```
hello-world
```

Try to pipe the output of the command to the binprint command:

```
hello-world | binprint
```

Try the broadcast command:

```
tutorial-broadcast change me
```

Where "change me" should be your name or email address.

Try a bunch of other commands:

```
help
sense | senseconv
power | powerconv
ls
format
echo test | write file
ls
read file
```

```
nodeid
blink 10
reboot
repeat 2 2 { echo again } &
ps
```

Exit with ctrl-C.

Open the tutorial-shell.c file. Locate the lines

```
shell_output_str(&hello_world,
             "Hello, world!", "");
```

Change “Hello, world” to something else, compile, upload, and login. Try the new hello-world command.

Finally, the most difficult exercise so far: add a tutorial-unicast command by copying the code from the tutorial-broadcast command and from the a tutorial-unicast.c file. Make sure that you correctly open the unicast connection and that you register the command with the shell. Compile, debug, compile, debug, ..., compile, run, test.

5 Cooja Simulation Exercises

This part of the tutorial shows how to simulate Contiki applications in Cooja, the Contiki network simulator. Cooja is Java-based, but simulates deployable Contiki programs. The simulator has been actively developed since 2006, and is included in Instant Contiki. In Cooja, nodes can be both simulated and emulated. Simulated nodes are compiled and executed natively, similar to how the native platform works but with glue drivers towards the simulator. Emulated nodes use MSPSim, the MSP430 emulator, to directly load and execute firmware files. Emulated nodes offer high timing accuracy and source-code debugging, but requires more memory and processing power. In this tutorial, we will use both.

5.1 Build and Start Cooja

Go to the Cooja directory, and start Cooja:

```
cd contiki-kth08
cd tools/cooja
ant run
```

Cooja builds, and after a few seconds the simulator starts. 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.

5.2 Create a Simulation

A new simulation is created via the menu.

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

A number of configuration options are presented. During this tutorial, we only use of one of these: “Mote startup delay”, a random node startup delay with which we avoid perfectly synchronized nodes.

- 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.

5.3 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**, **Contiki Mote Type**.

Cooja scans for valid Contiki processes to simulate, and after a few seconds a dialog with configuration options appears.

- Enter a **Description**, and select the process **test_etimer_process**.

The process periodically prints messages using Contiki’s event timers.

- Click **Compile** to compile the current Contiki configuration.

When the compilation finishes, close the compilation output window, and:

- 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.

5.4 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].

To add nodes:

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

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

5.5 Start Simulating

Two plugins are automatically started: a log listener and a visualizer. These, and other plugins, can be accessed via the plugins menu:

- Menu items: **Plugins, Log Listener** and **Plugins, State Visualizer**.

The log listener plugin listens to the serial ports of all nodes. Visualizers display information about nodes and their surroundings. For example, the UDGM visualizer allows you to see ongoing radio transmissions, and to change radio transmission ranges. In the Control Panel:

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

Debug output from nodes appear in the log listener plugin. Note the unsynchronized event times of the different nodes. This is due to our initial simulation configuration option, "Mote startup delay", is set to 1 second.

5.6 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*. When a simulation is reloaded, the simulation configuration is both extracted and restored. Reloading is used to reset the simulation; to restart all nodes. More importantly, this can be used during rapid prototyping: by reloading a simulation with evolving Contiki code, Cooja both compiles and simulates the code. To reload your current simulation:

- Click menu item: **File, Reload simulation**

5.7 Tutorial Programs

To simulate the tutorial programs, we need to create node types referencing to the code in:

```
contiki-kth08/examples/tutorials
```

A few of these programs use TmoteSky-dependant code, such as references to the CC2420 radio, and must hence be emulated using MSPSim. Prepared simulation configurations exist in the tutorial directory. Use these to immediately start simulating the tutorial programs.

Optional: Instead of using the menu option **Open simulation**, you can also load a simulation with **Open & Reconfigure simulation**. This allows you to overview and change configuration options when loading a simulation.

To load a tutorial simulation:

- Click menu item: **File, Open simulation, Browse...** and navigate to:

  ```
  contiki-kth08/examples/tutorials
  ```

The following four simulations are prepared.

5.7.1 BROADCAST.CSC (tutorial-broadcast.c)

Emulates 4 TmoteSky nodes sending periodic broadcast messages. Note the different boot times (0-20 seconds). Note also the big number of radio messages transmitted to deliver a single broadcast message, due to low power listening in X-MAC.

5.7.2 UNICAST.CSC (tutorial-unicast.c)

Emulates 6 TmoteSky nodes sending periodic unicast messages. You need to change the unicast destination address in the source code to match a simulated node's address, for example:

```
addr.u8[0] = 1;
addr.u8[1] = 0;
```

5.7.3 MESH_OS.CSC (tutorial-mesh-cooja.c)

Simulates 30 Contiki nodes in a mesh network. The source code is a modified version of tutorial-mesh.c - all references to the CC2420 radio are removed. You need to change the mesh destination address in the source code to match a simulated node's address, for example:

```
addr.u8[0] = 1;
addr.u8[1] = 0;
```

When all nodes have booted (after 1 second), simulate a button press by right-clicking a node in the visualizer and selecting **Click button on [...]**. Note that creating a route may require several button clicks.

5.7.4 SHELL.CSC (tutorial-shell.c)

Emulates 3 Tmote Sky nodes with the Sky shell. Try writing any of the following commands to a node serial port:

```
help
netcmd { blink 5 }
nodes
netcmd { reboot }
```

6 Conclusions

This course is an introduction to Contiki communication programming – use it as a starting point for further exploration. Feel free to copy the tutorial 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 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. Runtime 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.