Contiki IPSN 2009 Tutorial Handouts

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

April 2009

1 Introduction

Welcome to this Contiki tutorial! The purpose of the tutorial is twofold: to understand Contiki, and to set up an IPv6-based sensor network with 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) [10], and software-based power profiling [4]. Recent features include a networked shell and the memory efficient flash-based Coffee file system. In this tutorial, we use the uIPv6 stack to communicate with Tmote Sky motes and the network shell to interact with the network.

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

We use the Instant Contiki development environment. 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 tutorial, we use Instant Contiki 2.2.1 and an updated version of Contiki 2.2.3 that has a number of bugfixes from the current CVS version, as well as a set of programs we use for this tutorial.

Figure 1: A Tmote Sky mote.

2 System Requirements

For this tutorial, you need a PC that can run Instant Contiki (VMWare) and access to a Tmote Sky mote [11] (Figure 1. We have brought a set of Tmote Sky and Sentilla JCreate motes to participants to lend during the tutorial.

2.1 Download, Install Tools

Prior to starting the tutorial, download and install the following tools (all available from the Contiki website: http://www.sics.se/contiki/instant-contiki.html as well as from USB sticks we bring to the tutorial):

- VMWare Player
- FTDI Driver
- Instant Contiki

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

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**

3 Getting Started

Before starting, 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-2.x
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

Connect a Tmote Sky to 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 Mote

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 2.2.3 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.

3.8 Contiki Mote Shell

The Contiki shell is an interactive on-mote shell that provides a set of commands for interacting with the system. The shell can be accessed either over a serial USB connection or over a network using Telnet. In this tutorial, we first run the shell over a USB serial connection.

First, compile and upload the shell:

```
cd
cd contiki-2.x
cd examples
cd sky-shell
make sky-shell.upload
```

Wait for the compilation and uploading to finish.

To connect to the shell over the USB port, run:

```
make login
```

Press the return key to get a prompt. *(We have found that with Instant Contikik running in VMWare, garbage characters are sometimes printed out when connecting. They can be safely ignored.)*

Next, we try a few shell commands. To get a list of available commands, run:

```
help
```

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
```

The `power` command prints out the current power profile from Contiki's software-based power profiler. To print out the power profile in decimal digits, run:

```
power | binprint
```

The output will look something like:

```
12 236 0 37421 0 4 0 380 0 0 0 380 0
```

The numbers that the `power` command outputs are:

1. The number of data items that follow, to make automatic parsing of the output easier
2. CPU low-power mode time, least significant 16 bits
3. CPU low-power mode time, most significant 16 bits
4. CPU active mode time, least significant 16 bits
5. CPU active mode time, most significant 16 bits
6. Radio transmission time, least significant 16 bits
7. Radio transmission time, most significant 16 bits
8. Radio listen time, least significant 16 bits
9. Radio listen time, most significant 16 bits
10. Radio transmission time in idle listening, least significant 16 bits
11. Radio transmission time in idle listening, most significant 16 bits
12. Radio listen time in idle listening, least significant 16 bits
13. Radio listen time in idle listening, most significant 16 bits

The output of the `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 [11]. Because the `power` command only measures the time during which the peripherals are switched on, the power consumption estimate is not affected by environmental noise such as temperature differences nor by noise due to subtle differences between different versions of hardware.

4 IPv6 Networking

Contiki has supported IP networking, through the uIP TCP/IP stack [1], since its first release in 2003. In 2008, Cisco released the world's smallest IPv6 stack, called uIPv6, for Contiki [8].

In this tutorial, we setup a simple IP network and use it to login to a Contiki node over Telnet.

4.1 Upload a Telnet Binary

Before setting up the IPv6 network, we'll upload a Telnet server onto one mote, so that we have something to communicate with once we've got bridging up.

To upload a Telnet server onto a Tmote Sky, go to the telnet6-server directory in the ipsn-2009-tutorial directory:

```
cd
cd ipsn-2009-tutorial
cd telnet6-server
```

Next, we compile and upload the software to the mote:

```
make telnet-server.upload
```

Finally, we need to get the IPv6 address of the mote, to be able to ping and contact the node later. To get the IPv6 address of the mote, do

```
make login
```

Press the return key to reboot the mote. The mote prints out its IPv6 addresses when it boots. Copy the address starting with `aaaa` into a text editor window for later.

4.2 Setting up IPv6 Bridging with a Bridge Mote

One of the benefits of IP is that routing packets in and out from an IP network is easy. We have previously used a USB stick as a network interface that seamlessly connects an IP-based sensor network with a general-purpose IP network [9].

A USB bridge is, however, not yet widespread and in this tutorial we therefor focus on set up IP networking using only Tmote Sky/JCreate motes.

To get IPv6 packets routed into the sensor network, we use a Tmote Sky or JCreate as a bridge. The bridge mote acts as a link layer interface for the Linux host, as shown in Figure 2. The Linux host sends its IPv6 packets over the USB connection to the Tmote Sky/JCreate mote. Contiki forwards the IPv6 packets over the 802.15.4 radio, with SICSlowpan header compression.

Because the mote is the MAC/link layer interface of the Linux host, Linux needs to be know the MAC address of the Tmote Sky/JCreate that we use as a bridge.

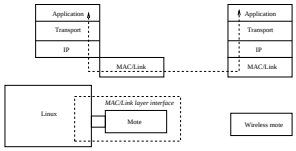

Figure 2: The mote acts as a link layer interface for the Linux host.

Contiki prints out its MAC address as part of its boot-up process, and we will use this to get the MAC address of the mote.

4.3 Obtain the MAC Address of the Bridge Mote

After connecting the mote to the USB connector, we download the Contiki shell to the mote:

```
cd contiki-2.x
cd examples/sky-shell
make sky-shell.upload
```

Wait a while for the compilation and uploading procedure to complete.

Next, log in to the shell:

```
make login
```

Wait a while for the serial connection to go up. Then press the return key to get the Contiki prompt.

At the prompt, type "reboot" and press return:

```
reboot
```

Contiki will reply with the text "Rebooting the node in 4 seconds...". After four seconds, Contiki reboots and prints out a message similar to the following:

```
Contiki 2.2.3 started. Node id is not set.
Rime started with address 19.38
MAC 00:12:74:00:11:e4:26:13 X-MAC channel 26
Starting 'Sky Contiki shell'
19.38: Contiki>
```

The 00:12:74:00:11:e4:26:13 in above is the 802.15.4 MAC address of the mote. The MAC address consists of eight bytes, displayed in hexadecimal format, with colons between.

Write down the MAC address, as we'll need it when setting up the bridging.

4.4 Upload Bridge Mote Software

We then upload the Contiki image that does the bridging to the Tmote Sky/JCreate:

```
cd
cd ipsn-2009-tutorial/uip6-bridge
make uip6-bridge-tap.upload
```

Wait for the compilation and upload to complete.

4.5 Set Up Bridging

To set up bridging, we first connect to the USB port of the Tmote Sky/JCreate, then we setup Linux to route packets to the sensor network. We use two separate terminal windows, one to connect, and one to set up bridging.

Open two new terminal windows. In both windows, become root:

```
sudo -s
```

When asked for a password, type "user" (without quotes).

In the first window, connect to the Tmote Sky/JCreate over USB:

```
cd ipsn-2009-tutorial
make connect
```

An LED on the Tmote Sky / JCreate will light up for a few seconds. The Tmote Sky / JCreate is now connected as a network interface in Linux. Next, we configure Linux to send and receive packets over the Tmote Sky / JCreate.

In the second window, we configure Linux to route packets over the USB. To do this, we need the MAC address that we previously obtained. Take the last six bytes of the MAC address (in the above example, this was 74:00:11:e4:26:13, but you should use the MAC address you obtained, not the one in this example) and use it as follows:

```
cd ipsn-2009-tutorial
make bridge MAC=74:00:11:e4:26:13
```

The bridge mote should light up its LED for a few seconds.

The IP network is now connected over the Tmote Sky / JCreate.

4.6 Use Ping and Telnet

With the bridge mote acting as a network interface, we can now connect to our Telnet mote. First, we use the Linux command ping6 to ping the mote with IPv6. Use the IPv6 address that we copied in Section 4.1:

```
ping6 aaaa::0212:7400:11e4:2613
```

Note: use the IPv6 address obtained from your own mote, not the exact address above. The address above is used only as an example.

Connect to the shell on the mote:

```
telnet aaaa::0212:7400:11e4:2613
```

Note: use the IPv6 address obtained from your own mote, not the exact address above. The address above is used only as an example.

4.7 Debugging the IPv6 Network

Any IP sniffer software, such as tcpdump and Ethereal, can be used to debug the connection to the IP network. To use tcpdump to sniff the packets going over the bridge mote, run:

```
sudo tcpdump -lni tap0
```

This shows all IP packets going over the mote bridge to and from the IPv6 sensor network.

5 Conclusions

This tutorial is an introduction to IP-based sensor networks with Contiki – 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 the shell 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.

Visit the Contiki web site for documentation, tutorials, and the mailing list: 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] M. Durvy, J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels. Making Sensor Networks IPv6 Ready. In *Proceedings of the Sixth ACM Conference on Networked Embedded Sensor Systems (ACM SenSys 2008)*, Raleigh, North Carolina, USA, November 2008.

[9] G. Mulligan, C. O'Flynn, M. Durvy, J. Abeillé, P. Wetterwald, B. Leverett, E. Gnoske, M. Vidales, N. Tsiftes, N. Finne, and A. Dunkels. Seamless sensor network ip connectivity. In *Proceedings of the 6th European Conference on Wireless Sensor Networks, EWSN 2009*, Cork, Ireland, February 2009.

[10] 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.

[11] 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.