COSC402 Lab 5: Contiki Programming

1 Description

In this lab, you need to use MAC OS. You will:

- Learn how to upload a program to a sensor mote;
- Learn how to use the COOJA simulator;
- Learn how to use Contiki to develop a very simple program.

2 Setup virtual machine

1. Go to the course pickup folder (/home/cshome/coursework/402/pickup/), and copy the Contiki development tool (COSC402-lubuntu64) in the folder named 402Contiki to your local machine. It will take a couple of minutes to finish the copying as the development tool is very large.
2. Start VirtualBox, and add the machine by going to **Machine** → **Add** and select the machine files /COSC402-lubuntu64/COSC402-lubuntu64.vbox.
3. Start the virtual machine you just added.

3 Hands-on exercises

3.1 System verification

Open a terminal window, and go to the hello-world example directory (∼/contiki-2.6/examples/hello-world/), and compile it for the native platform:

```
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.

3.2 Connect the sensor mote

Connect the sensor mote to the computer via the USB port. Click the USB icon on the status bar at the bottom of the virtual machine, and the sensor mote will appear on the top with the name "FTDI MTM-CM5000MSP[0400]". Click on the name to connect the sensor mote to Contiki.

3.3 Run Hello-world on the sensor mote

Compile and upload the Hello World program on the Tmote Sky:
cd ~/contiki-2.6/examples/hello-world/

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

Wait for the compilation and uploading procedure to finish, and try to understand each step for uploading. During the uploading the sensor mote should quickly flash the red LEDs next to the USB connector.
Log into the USB port to view the program output:

```
make login TARGET=sky
```

Press the reset button on the sensor mote and a message similar to the following should appear:

```
Rime started with address 86.177
MAC 56:b1:00:00:00:00:00:00 Contiki 2.5 started. Node id is not set.
CSMA ContikiMAC, channel check rate 8 Hz, radio 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. Note that "Node id is not set" in the output. You can assign a node-id to the sensor mote using the following command:

```
make burn-nodeid.upload TARGET=sky  nodeid=33
```

Login again and press the reset button to see if the new node-id has been burned.

3.4 Contiki Shell

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 sky-shell
make sky-shell.upload TARGET=sky
```

When compilation and uploading finishes, login to your sensor mote:

```
make login
```

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

```
help
reboot
blink 10
nodeid 20
sense | senseconv
```

3.5 Simulating Contiki with COOJA

This section explains how to simulate Contiki programs in the COOJA simulator.

3.5.1 Start COOJA

```
cd contiki-2.6/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.

3.5.2 Create a simulation

A new simulation is created via the menu.

- Click menu item: File, New Simulation. A number of configuration options are presented. 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.

3.5.3 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: Motes, Add Motes, Create new mote type, Sky mote. 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
- Click Create when complication finishes.

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.

3.5.4 Add simulated nodes

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

- Menu item: Motes, Add motes, [your type description].
- Enter the number of nodes you want to simulate (e.g., 5), and press Create and Add.

Five nodes are added to the simulation, randomly located in the XY-plane. Instead of using the Random Position, you can also use Linear, Ellipse, or Manual Positioning. Note that when you add nodes later, the visualiser plugin is rescaled in order to distribute the nodes all over the XY- plane. Moving nodes is possible by left-clicking on a node and dragging it across the XY-plane. Go to the "Network" window, click the view menu, and choose the properties you want to show for the nodes.

3.5.5 Start simulation

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

3.5.6 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. 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, press Ctrl+R

3.6 Simulating communications

3.6.1 broadcast

Go to ∼/contiki-2.6/examples/rime/, open example-abc.c. This example program sends broadcast packets with a random interval between 2 and 4 seconds. All nearby sensor motes will receive the packet. Simulate this program in COOJA and see how does it work.
Open the file 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 sensor mote: in the terminal window, type:

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

This will compile and upload the program to the sensor mote connected to your PC. When the compilation and uploading has finished, watch your name appear on the projector screen. To see what broadcast messages your Tmote Sky sees, type the following in the terminal window:

```
make login TARGET=sky
```

This shows all serial output that the sensor mote 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, dont be afraid to ask!

3.6.2 unicast

The unicast program looks similar to the broadcast program, however, a destination address is needed. Now start COOJA and create two nodes that run the program example-unicast in examples/rime. If you look at the COOJA Mote output window, you will see that only one node receives packets (if the nodes are in transmission range, check with in Network window by selecting UDGM skin and left-clicking on one of the nodes). Your next task is to try to make both nodes send and receive messages. For this, you use the nodes ID that you can include with

```
#include "node-id.h"
```

and check the node ID of a node with, e.g.

```
if (node_id == 2)
```

Then update the receiver address in the lines

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

Can you get both two nodes to send messages to each other? Also make the receiver print out the received message (hint: check the abc example). You can play with the Mote output tool. Also choose the View menu in the Mote output window and tick Mote-specific coloring. Very useful!

4 Programming

Implementing another version of blink using etimer

Go to the directory ∼/contiki-2.6/TELE402/Flash/, open flash.c, and complete the code according to the instructions given the code.

- please refer to core/sys/etimer.h and the examples given in ∼/contiki-2.6/examples/ for details on how to use etimer.
- please refer to core/dev/leds.h for details on how to control the leds.