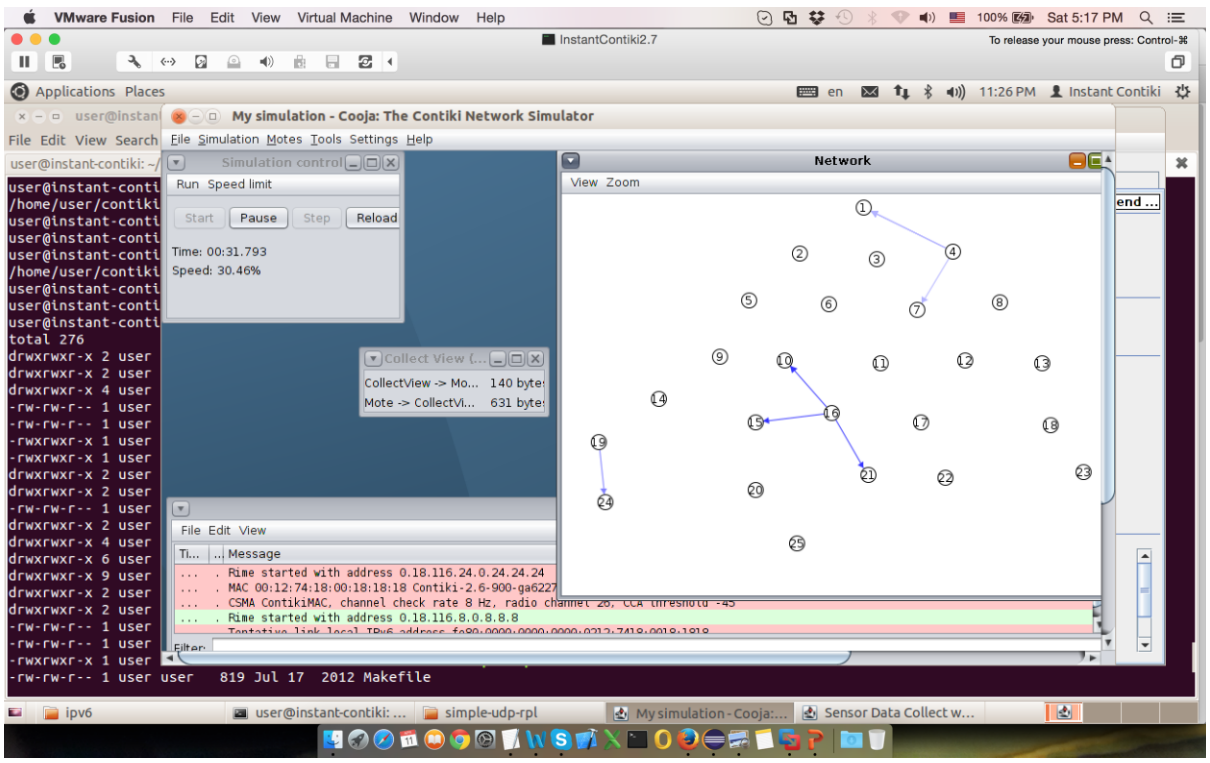
Edinburgh Napier UNIVERSITY

Cooja Simulator Manual

Version 1.0



My simulation - Cooja: The Contiki Network Simulator
File Simulation Motes Tools Settings Help
Simulation control
Run Speed limit
Start
Pause
Step
Reload
Time: 00:31.793
Speed: 30.46%
Collect View
CollectView -> Mo... 140 byte
Mote -> CollectVi... 631 byte
Network
View Zoom

```
user@instant-conti
/home/user/contiki
user@instant-conti
user@instant-conti
user@instant-conti
/home/user/contiki
user@instant-conti
user@instant-conti
user@instant-conti
total 276
drwxrwxr-x 2 user
drwxrwxr-x 2 user
drwxrwxr-x 4 user
-rw-rw-r-- 1 user
-rw-rw-r-- 1 user
-rwxrwxr-x 1 user
-rwxrwxr-x 1 user
drwxrwxr-x 2 user
drwxrwxr-x 2 user
-rw-rw-r-- 1 user
drwxrwxr-x 2 user
drwxrwxr-x 4 user
drwxrwxr-x 6 user
drwxrwxr-x 9 user
drwxrwxr-x 2 user
drwxrwxr-x 2 user
-rw-rw-r-- 1 user
-rw-rw-r-- 1 user
-rwxrwxr-x 1 user
-rw-rw-r-- 1 user user    819 Jul 17  2012 Makefile
```

```
Ti... | ... | Message
...     .   Rime started with address 0.18.116.24.0.24.24.24
...     .   MAC 00:12:74:18:00:18:18:18 Contiki-2.6-900-ga6227
...     .   CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26, CCA threshold -45
...     .   Rime started with address 0.18.116.8.0.8.8.8
```

Edited by: "IoT Networking Research Group"

Edinburgh Napier University (C) 2015-2016

List of Contributors

- Craig Thomson
- Dr Imed Romdhani
- Dr Ahmed Al-Dubai
- Mamoun Qasem
- Barraq Ghaleb
- Isam Wadhaj

Table of Contents

Table of Figures

1. Introduction

While Cooja has been proven to be an ideal tool for the simulation of RPL in WSNs, there are challenges involved in its use. This is particularly pertinent in regard to the lack of documentation available. The Contiki website [1] may be a first port of call in regard to Cooja, and provides an image of Instant Contiki which can then be used with the virtualisation tool VMware [2]. However, once Instant Contiki is successfully started, the Contiki website can then be referred to for nothing more than brief instructions regarding a simple network setup on Cooja. With this the sum total of any official documentation regarding Cooja, with the majority of support being provided within internet discussion boards.

The aim of this tutorial, therefore, is to provide an extensive instruction set in the use of Cooja. This is based on the personal experience of the contributors involved, enabling the reader to avoid some of the pitfalls inherent in attempting to use Cooja with no or limited experience. This will firstly enable someone with no previous exposure to Cooja to be able to create network layouts, compile motes, examine output using the Sensor Data Collect plugin and also utilise scripts to produce more fine-grained results. From this starting point we move onto more complex tasks including the manipulation of the Cooja code and the use of Cooja in physical nodes. This tutorial seeks to be both a reference point for a user seeking specific knowledge in a particular task, as well as a step by step instruction set for the new user.

2. The Cooja GUI – From the Beginning

1.1 Start-up and Initial Settings

If this tutorial is a complete starting point in the use of Cooja you will first need to visit the Contiki website[1] in order to download Instant Contiki. Once the Instant Contiki image has been downloaded and unzipped it can be opened using VMware. Instant Contiki is an Ubuntu based operating system with Cooja already built in and ready to use. To start the simulation software open a terminal window and enter the following commands:

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

In the event that large, memory heavy simulations are to be run, the command **'ant run_bigmem'** can alternatively be used. The Cooja software should now start, resulting in the screen as shown in Figure 1. The Contiki website now demonstrates how to set up a simple 'Hello World' simulation. However, this tutorial demonstrates how to set up a simple RPL network utilising a UDP Sink mote and several UDP Sender motes. Although Cooja can be extremely complex, by following a few simple instructions, very quickly data can be collected from motes as well as giving you the ability to observe radio communication, messages output from motes within the network and to alter the transmission and interference range of motes.

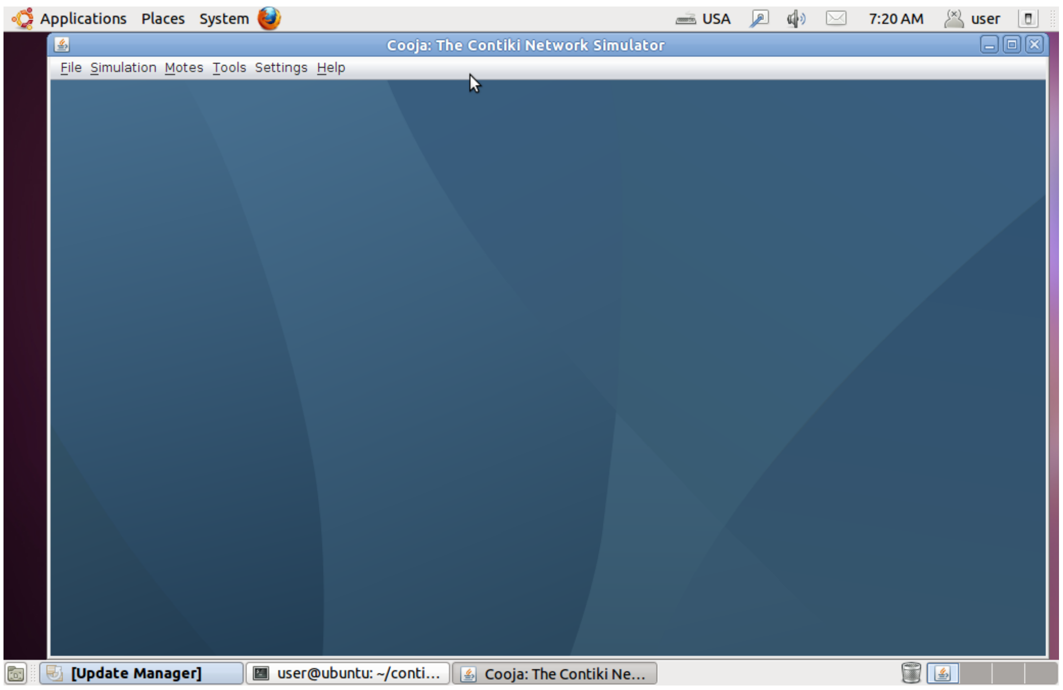
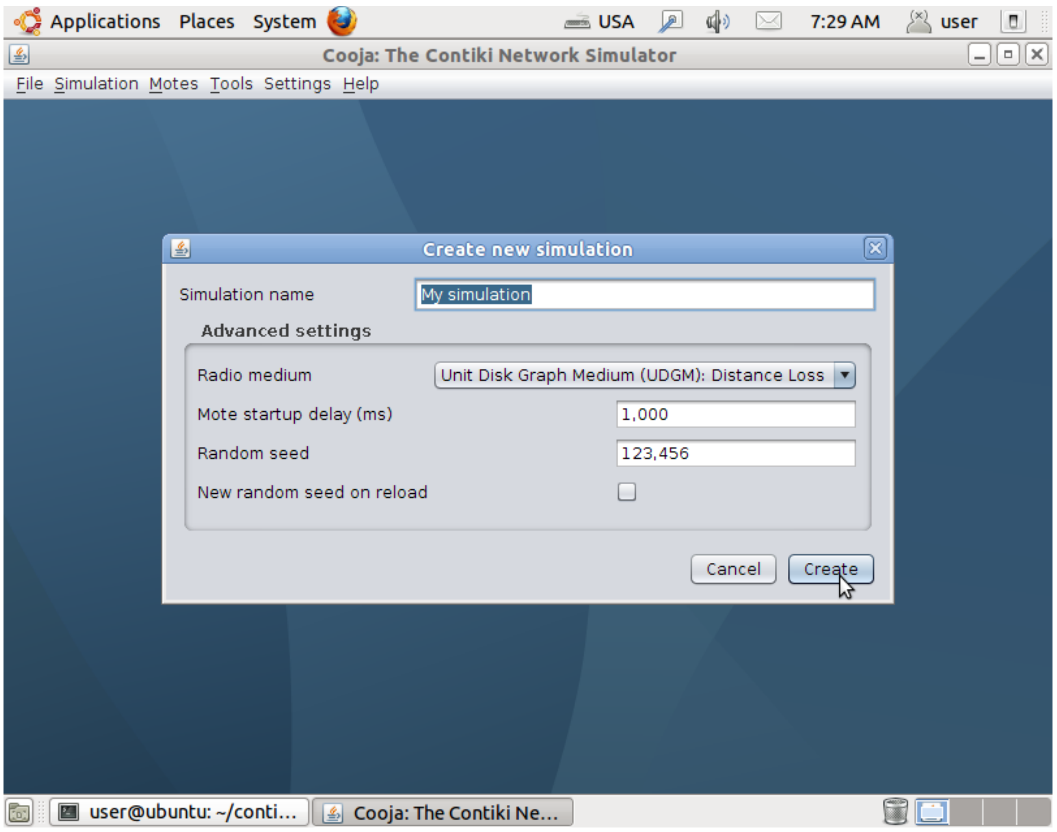


Figure 1. Initial Cooja Screen [1]

Click on **File** and then **New Simulation** and the screen shown in Figure 2 is returned. The settings on this screen do not need to be changed, with deployment of the Unit Disk Graph Model (UDGM) covered in greater detail later. The **Create** button can now be clicked. This results in the initial simulation screen as shown in Figure 3. At the moment there is nothing to run as there are no motes in the network as of yet. These are added by clicking on **Motes**, **Add motes**, **Create new mote type** and then **Sky mote** from the resulting drop-down menu as demonstrated in Figure 4. The Sky mote is the most simple of motes for use within a WSN and ideal for initial configurations within a Cooja simulation. The resultant screen is displayed in Figure 5. It is here where the firmware for the mote is selected and this is also where these instructions diverge from the instructions on the Contiki website.



Simulation name: My simulation
Radio medium: Unit Disk Graph Medium (UDGM): Distance Loss
Mote startup delay (ms): 1,000
Random seed: 123,456
New random seed on reload: ☐

Figure 2. Cooja Create new simulation [1]

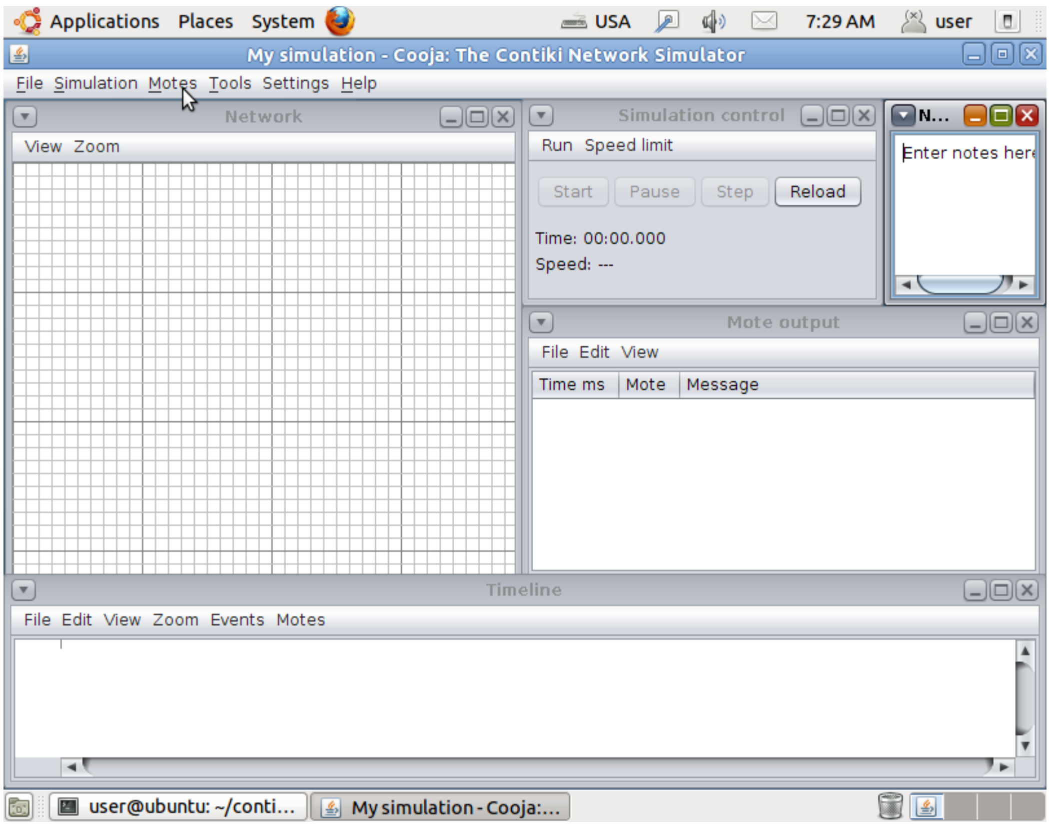


Figure 3. Initial Cooja Simulation Screen [1]

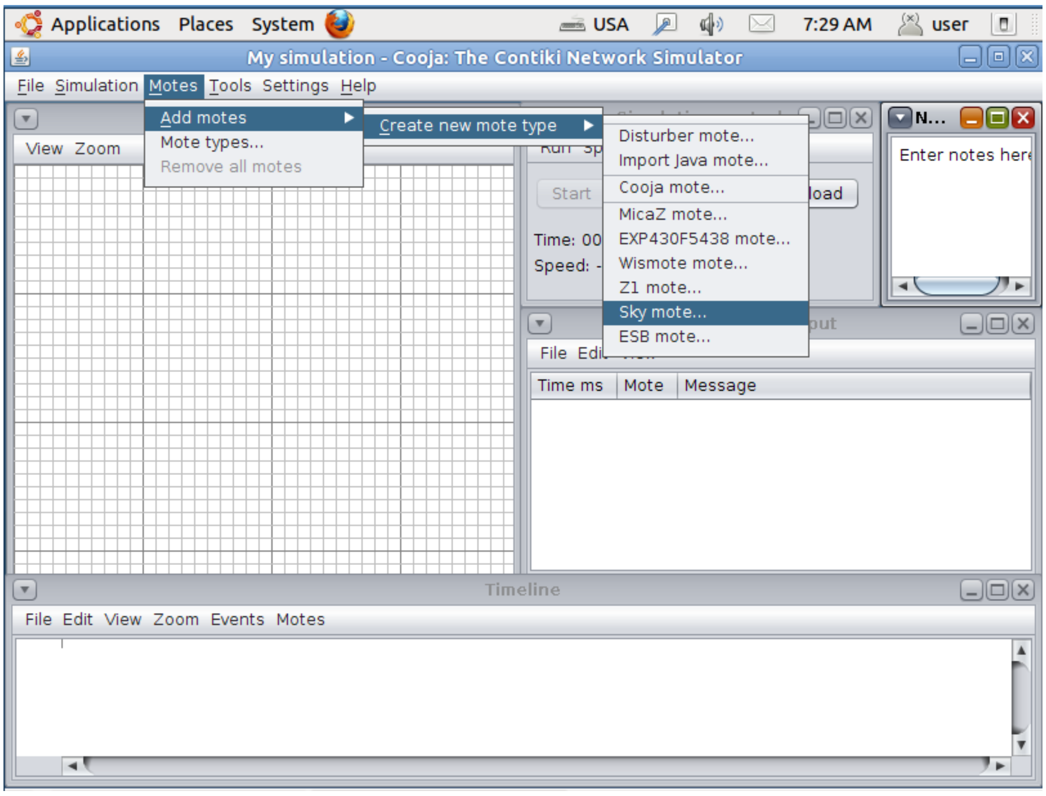
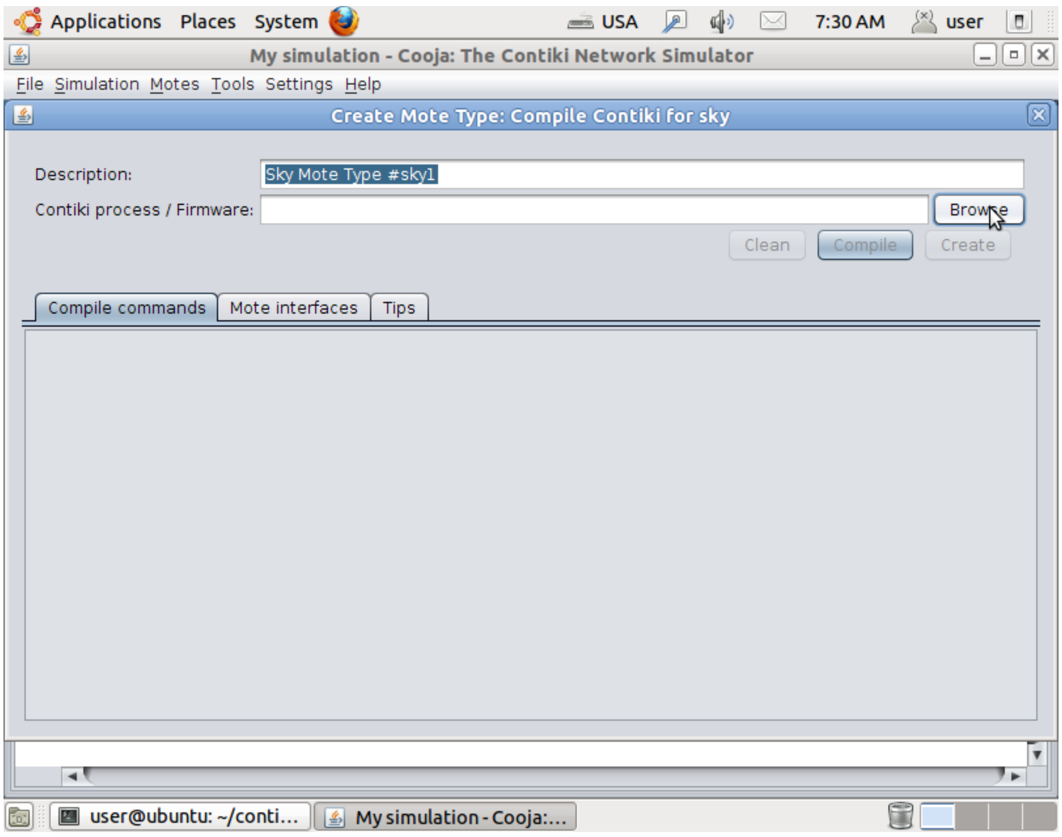


Figure 4. Cooja Add Mote Dropdown [1]

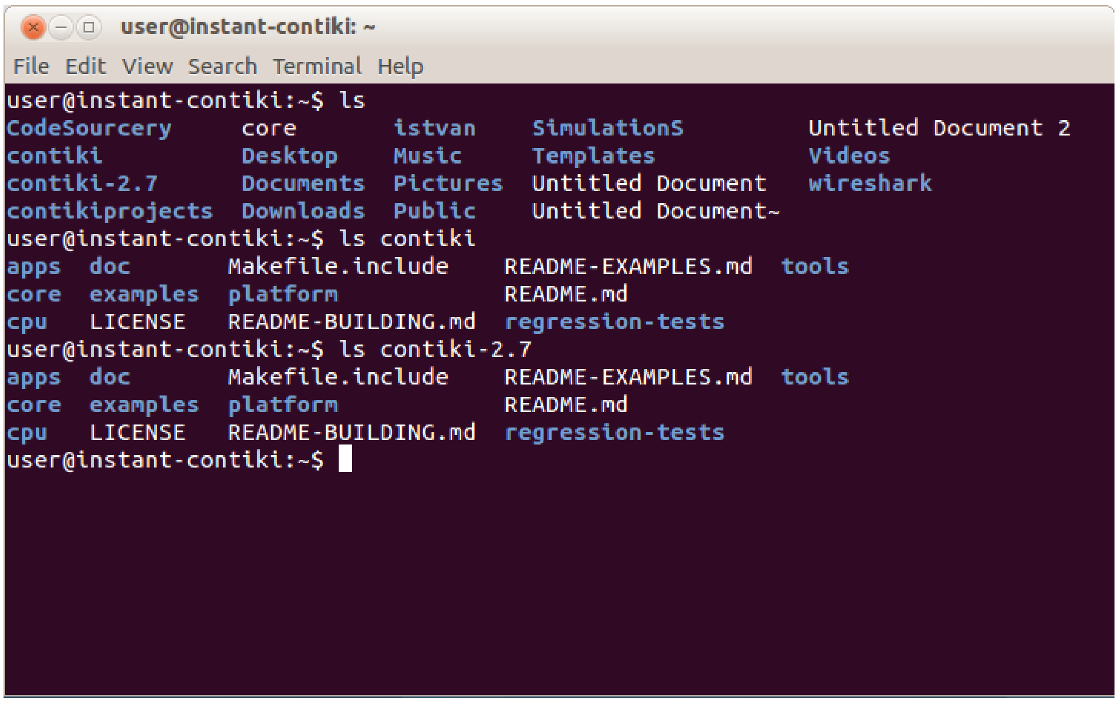

Applications Places System
USA 7:30 AM user
My simulation - Cooja: The Contiki Network Simulator
File Simulation Motes Tools Settings Help
Create Mote Type: Compile Contiki for sky
Browse
Clean
Compile
Create
Compile commands
Mote interfaces
Tips
user@ubuntu: ~/conti...
My simulation - Cooja:...

Description: Sky Mote Type #sky1

Contiki process / Firmware:

Figure 5. Cooja Locate Mote Firmware [1]

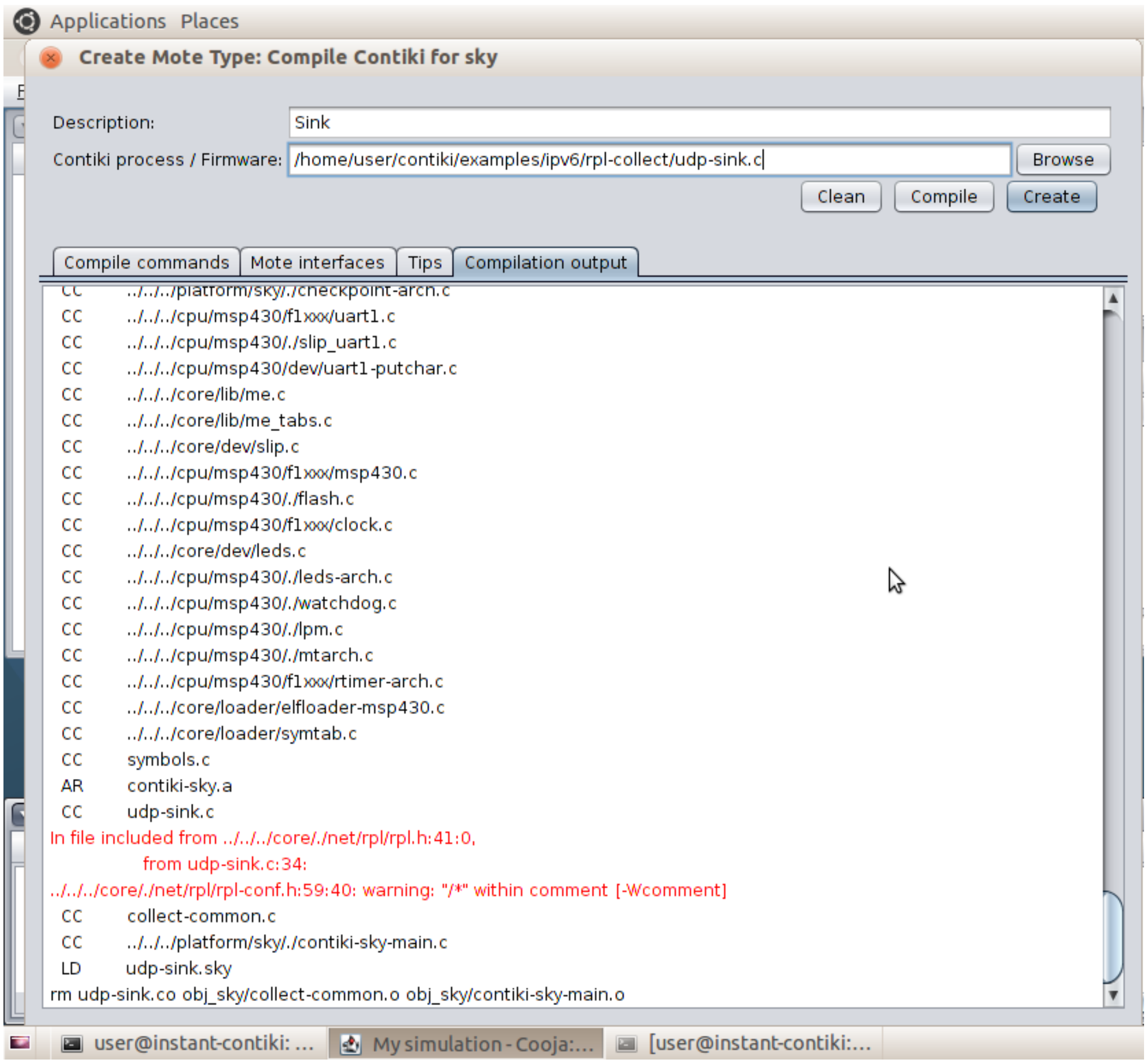
It is advisable to change the description to something more relevant to the test to be performed. In this case, the network topology shall involve a sink node and several sender nodes, therefore the description is changed to '**Sink'**. A great advantage of Cooja is that the motes used in a Cooja simulation use the same firmware as actual physical devices. At this point the firmware to be used to create the simulated mote must be located in order to create and compile it. It should be noted that the location of the firmware is extremely important. When Instant Contiki 2.7 is used there are two paths created within the file structure, **contiki** and **contiki-2.7**, as shown in Figure 6. As all tests created within this dissertation are run from the **contiki** path, this is where the firmware should be located. This becomes more important when adjustments are made to the firmware of motes in order to create customised versions. If these are located within the **contiki-2.7** path it will not cause an error. However, any further changes to source code would not be picked up as the motes are compiled every time a test is opened.



```
user@instant-contiki:~$ ls
CodeSourcery     core       istvan    SimulationS        Untitled Document 2
contiki          Desktop    Music     Templates          Videos
contiki-2.7      Documents  Pictures  Untitled Document  wireshark
contikiprojects  Downloads  Public    Untitled Document~
user@instant-contiki:~$ ls contiki
apps  doc       Makefile.include    README-EXAMPLES.md  tools
core  examples  platform            README.md
cpu   LICENSE   README-BUILDING.md  regression-tests
user@instant-contiki:~$ ls contiki-2.7
apps  doc       Makefile.include    README-EXAMPLES.md  tools
core  examples  platform            README.md
cpu   LICENSE   README-BUILDING.md  regression-tests
user@instant-contiki:~$ 
```

Figure 6. Contiki File Structure

As can be seen in Figure 6 there is an '**examples'** folder and this is where the firmware is located, with a great many options available. As this example involves the use of RPL the path selected is **/home/user/contiki/examples/ipv6/rpl-collect/udp-sink.c**. udp-sink.c is the actual C language firmware of the mote that will now be created. Click **Clean** to erase any previous compile of the mote and then **Compile**. This will result in output as in Figure 7 which shows the compilation output. There will always be some warning code in red, however, providing there are no red errors at the end of the output the mote has compiled successfully.

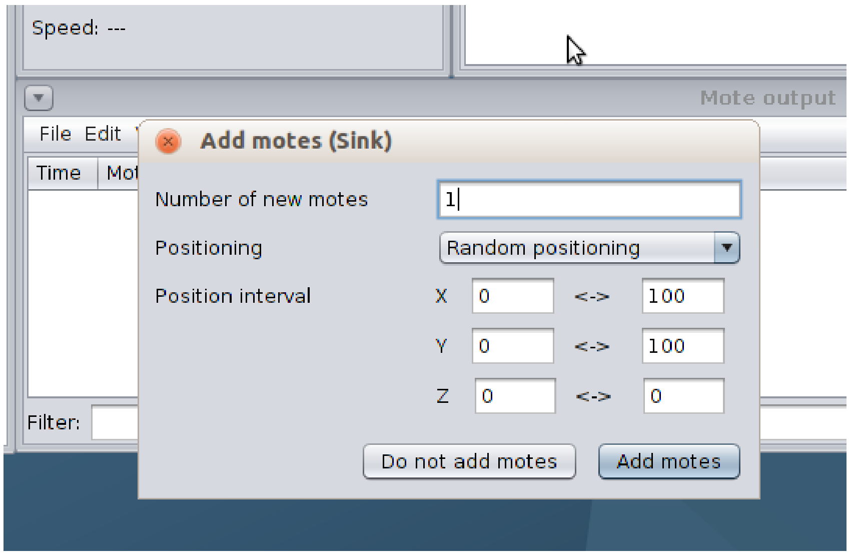

Applications Places
Create Mote Type: Compile Contiki for sky
Browse
Clean
Compile
Create
Compile commands
Mote interfaces
Tips
Compilation output
user@instant-contiki: ...
My simulation - Cooja:...
[user@instant-contiki:...

Description: Sink
Contiki process / Firmware: /home/user/contiki/examples/ipv6/rpl-collect/udp-sink.c

```
 CC       ../../../platform/sky/./checkpoint-arch.c
 CC       ../../../cpu/msp430/f1xxx/uart1.c
 CC       ../../../cpu/msp430/./slip_uart1.c
 CC       ../../../cpu/msp430/dev/uart1-putchar.c
 CC       ../../../core/lib/me.c
 CC       ../../../core/lib/me_tabs.c
 CC       ../../../core/dev/slip.c
 CC       ../../../cpu/msp430/f1xxx/msp430.c
 CC       ../../../cpu/msp430/./flash.c
 CC       ../../../cpu/msp430/f1xxx/clock.c
 CC       ../../../core/dev/leds.c
 CC       ../../../cpu/msp430/./leds-arch.c
 CC       ../../../cpu/msp430/./watchdog.c
 CC       ../../../cpu/msp430/./lpm.c
 CC       ../../../cpu/msp430/./mtarch.c
 CC       ../../../cpu/msp430/f1xxx/rtimer-arch.c
 CC       ../../../core/loader/elfloader-msp430.c
 CC       ../../../core/loader/symtab.c
 CC       symbols.c
 AR       contiki-sky.a
 CC       udp-sink.c
In file included from ../../../core/./net/rpl/rpl.h:41:0,
                 from udp-sink.c:34:
../../../core/./net/rpl/rpl-conf.h:59:40: warning: "/*" within comment [-Wcomment]
 CC       collect-common.c
 CC       ../../../platform/sky/./contiki-sky-main.c
 LD       udp-sink.sky
rm udp-sink.co obj_sky/collect-common.o obj_sky/contiki-sky-main.o
```

Figure 7. Cooja Mote Compilation

Now click on **Create** to bring up the option to create the number of motes required. A box shall appear as in Figure 8. As this is a Sink mote only one is required, click **Add motes** and the mote is added. This process should be repeated for the Sender motes with the path for the mote firmware now **/home/user/contiki/examples/ipv6/rpl-collect/udp-sender.c**. In this example we shall be adding 5 Sender motes.


Add motes (Sink)

Number of new motes: 1

Positioning: Random positioning

Position interval: X 0 <-> 100; Y 0 <-> 100; Z 0 <-> 0

Figure 8. Cooja Add Motes

Once the **'Add motes'** button has been clicked a screen similar to Figure 9 is returned, showing the motes in the network with number 1 being the Sink mote.

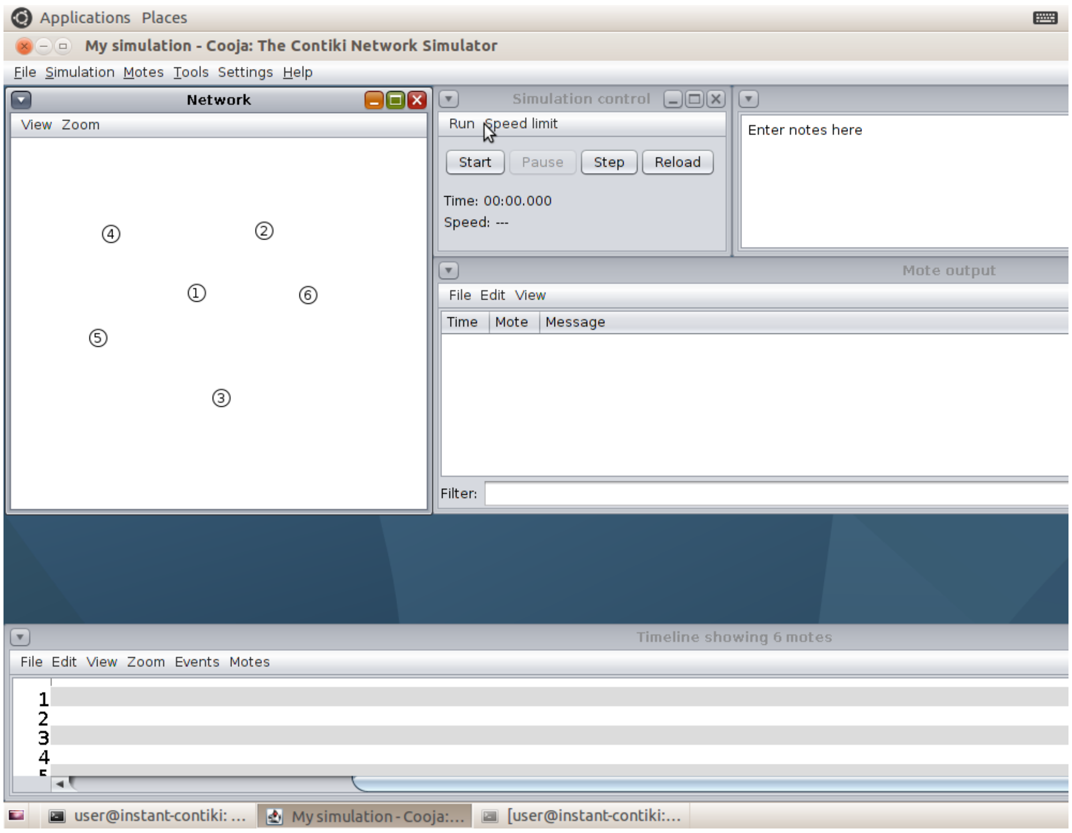


Figure 9. Cooja Initial Network

1.2 Network Options

This is a good point to describe the various windows and their purpose, as well as the many other display options available before actually running the network test:

- The **Timeline** window at the bottom of Figure 9 shows events over a period of time such as layer 2 communication to motes to waken them up [1]. This view can be filtered using the drop down menu and the output can be saved to a file if necessary.
- The **Mote Output** window shows the output window will display any printouts from the motes. This can be extremely useful in more complex networks which require more fine-grained analysis of results, as the actual source code of the motes can be altered at various levels, with messages produced which can be observed in this window. This is also where any print messages used to ascertain the flow of code will appear.
- The **Simulation Control** window is where the simulation is started, paused and stopped. The simulation can also be completely reloaded from this window although there are other options in this regard.
- The **Network** window displays the layout of the network motes and can be greatly modified in order to more easily display various factors of the network, as well as network traffic. The **View** dropdown menu shows the various options as shown in Figure 10.

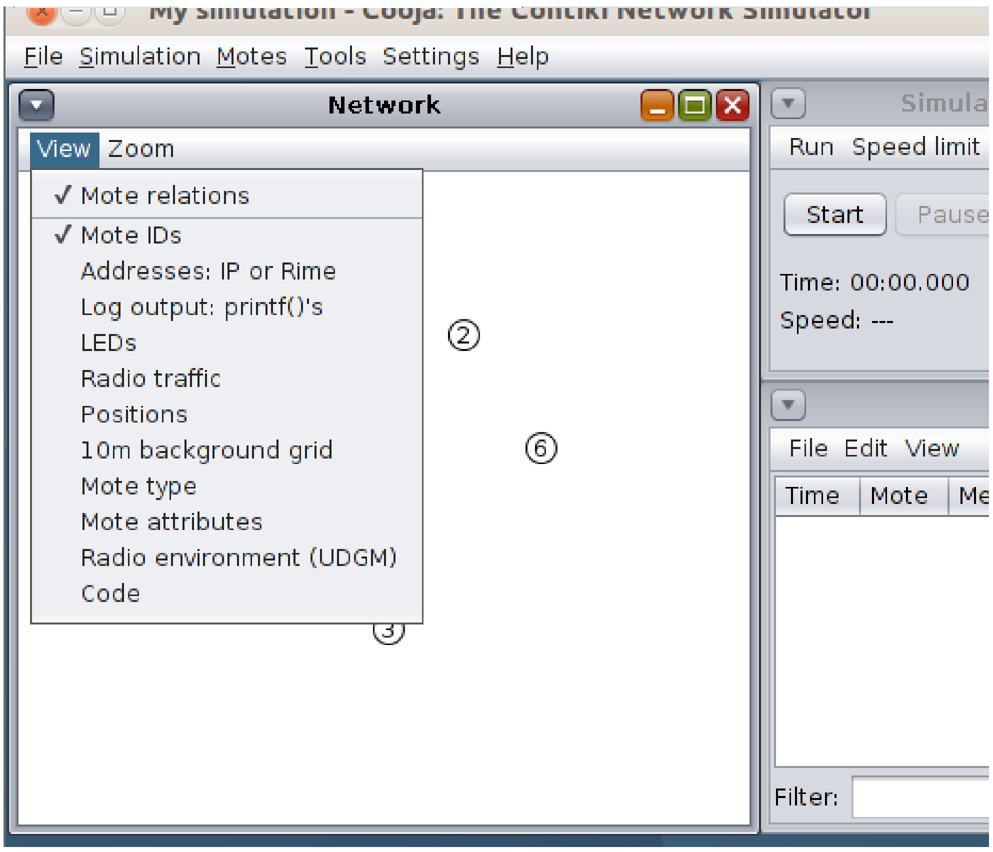
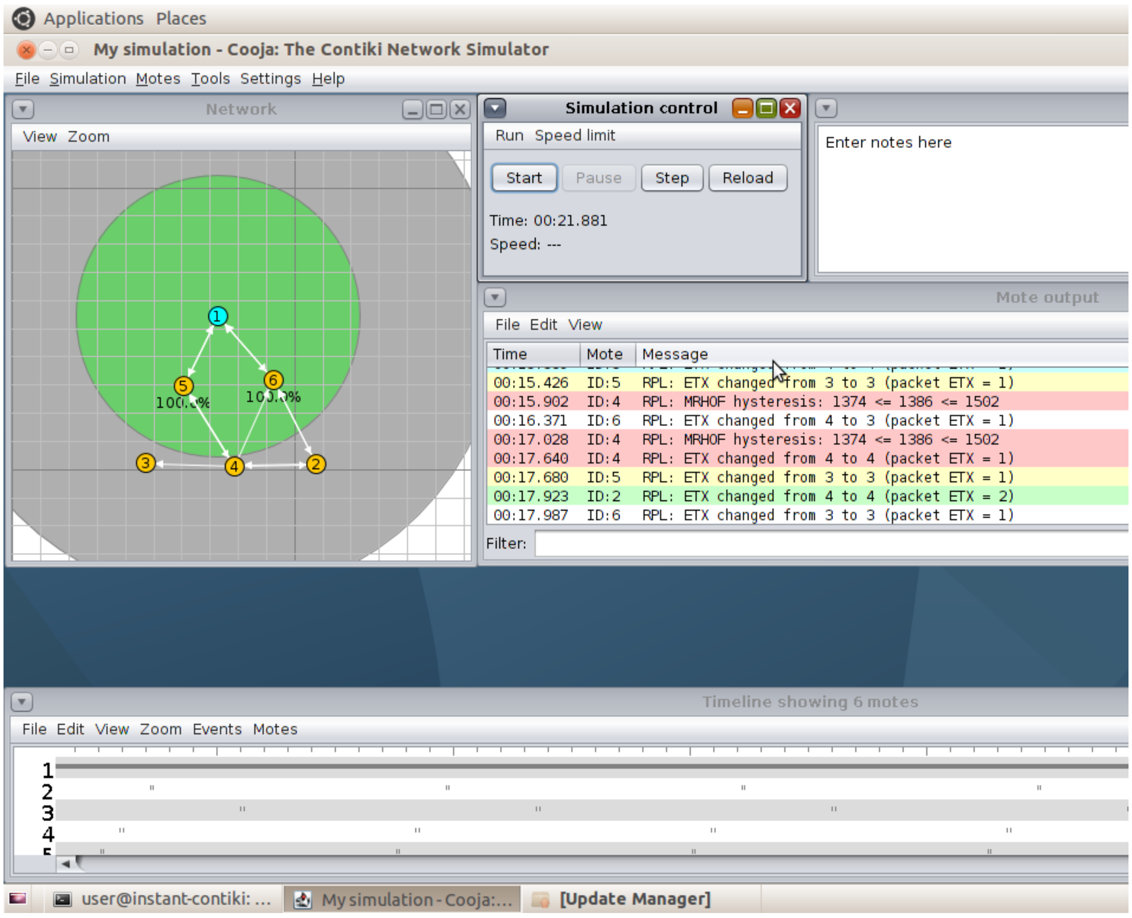


Figure 10. Cooja View Dropdown

It should be noted that these options are best utilised selectively, as the window can become cluttered with too many options selected. As can be observed, **'Mote relations'** and '**Mote IDs'** are already ticked, showing the numbering for each mote. The rest of the options are detailed as follows:

- The **'Addresses: IP or Rime'** option displays the addresses, in this case IPv6, of each mote. Given the length of an IPv6 address it can be seen how this option should be used sparingly as in a fairly dense network the addresses simply will not be viewable.
- **'Log output: printf()'s'** displays printf messages from the mote code inside the actual view window. These messages also appear within the **Mote Output** window which could be argued as a better way of analysing and filtering results.
- **'LEDs'** is useful in order to observe the LED lights on the simulated motes.
- The **'Radio Traffic'** option is extremely useful in animating the network once the simulation is running. Using this option will display the exchange of messages between nodes allowing the observation of the selection of parents and how the DODAG is built.
- **'Positions'** displays the relative location of each node.
- The **'10m background grid'** is extremely useful to give a sense of scale to the network layout. This displays a grid of 100m^2 squares rather than a blank background.
- **'Mote type'** employs a colour scheme to show the difference between motes of different types. In the case of this demonstration network two types of motes are employed – one sink and five senders.
- **'Mote attributes'** allows the use of code to alter the colourising of motes as well as other options and is not relevant to this discussion.
- **'Radio environment (UDGM)'** displays the transmission range of any particular mote and is extremely useful when deciding upon the optimal position of motes within a simulated network.
- **'Code'** merely displays code as it is being run and is not of relevance to this discussion.
- In the case of this demonstration the following options are used:
    - **Radio Traffic**
    - **10m background grid**
    - **Mote type**
    - **Radio environment (UDGM)**

The motes have been reorganised into a tree formation and the **Zoom** option utilised in order to place them at a more realistic distance. In order to change the transmission range and interference range of the motes simply right click any mote and select **'Change transmission ranges'**. Once the range has been changed the simulation must be reloaded. An example of this network, paused during the exchange of Hello messages, with a transmission range of 50m and an interference range of 50m can be seen in Figure 11. These instructions allow the setup and execution of a simple network simulation using Cooja, however, as can be clearly observed there are far more options than have been covered in this section. Two main areas of interest are in regard to utilising the script editor, primarily to limit the timing of a simulation, and the collection of sensor statistics. These shall be examined in greater detail in the next section.



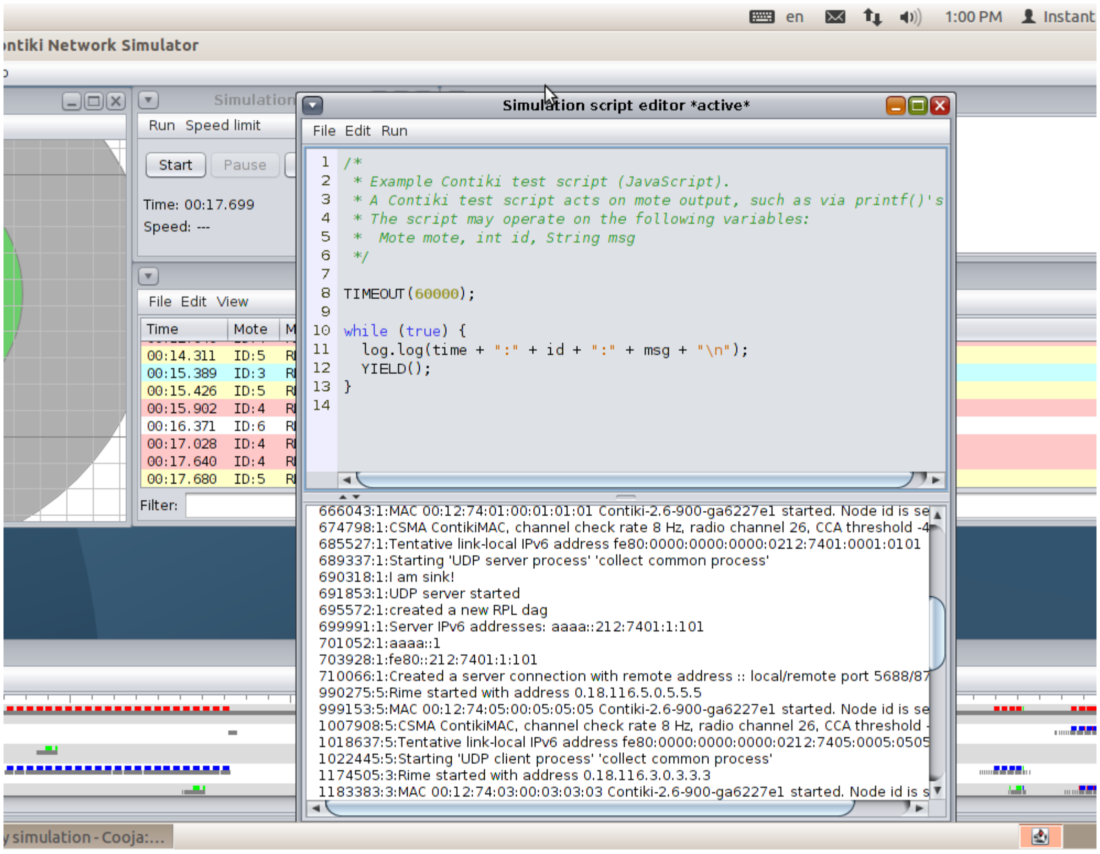
```
Time: 00:21.881
Speed: ---
```

```
Time       Mote   Message
00:15.426  ID:5   RPL: ETX changed from 3 to 3 (packet ETX = 1)
00:15.902  ID:4   RPL: MRHOF hysteresis: 1374 <= 1386 <= 1502
00:16.371  ID:6   RPL: ETX changed from 4 to 3 (packet ETX = 1)
00:17.028  ID:4   RPL: MRHOF hysteresis: 1374 <= 1386 <= 1502
00:17.640  ID:4   RPL: ETX changed from 4 to 4 (packet ETX = 1)
00:17.680  ID:5   RPL: ETX changed from 3 to 3 (packet ETX = 1)
00:17.923  ID:2   RPL: ETX changed from 4 to 4 (packet ETX = 2)
00:17.987  ID:6   RPL: ETX changed from 3 to 3 (packet ETX = 1)
```

Figure 11. Cooja View Options Utilised

1.3 Script Editor and Sensor Collect

In order to start the script editor within Cooja, select the **'Tools'** dropdown menu and the **'Simulation Script Editor'**. If familiar with Java Script you can use this facility to perform fine-grained analysis of the network output. However, for simplicity, the Script Editor can be used merely to display messages and to set a timer on the simulation. In order to do this, within the Script Editor menu select the **'File'** dropdown menu of which **'Load example script'** is the only option. Then select **'Just log all printf()'s and timeout'**. This results in a simple script as in Figure 12, with the greatest point of interest being the **TIMEOUT**. This number is displayed in milliseconds therefore the 60000 shown is 60 seconds, or 1 minute. This can be adjusted accordingly. However, note that the changes only take effect once the **'Run'** dropdown menu is clicked and **'Activate'** selected. The script can now no longer be edited until the same process is repeated. Figure 12 also shows the simulation paused and how messages can be displayed within the Script Editor output pane.



```
/*
 * Example Contiki test script (JavaScript).
 * A Contiki test script acts on mote output, such as via printf()'s
 * The script may operate on the following variables:
 *  Mote mote, int id, String msg
 */

TIMEOUT(60000);

while (true) {
  log.log(time + ":" + id + ":" + msg + "\n");
  YIELD();
}
```

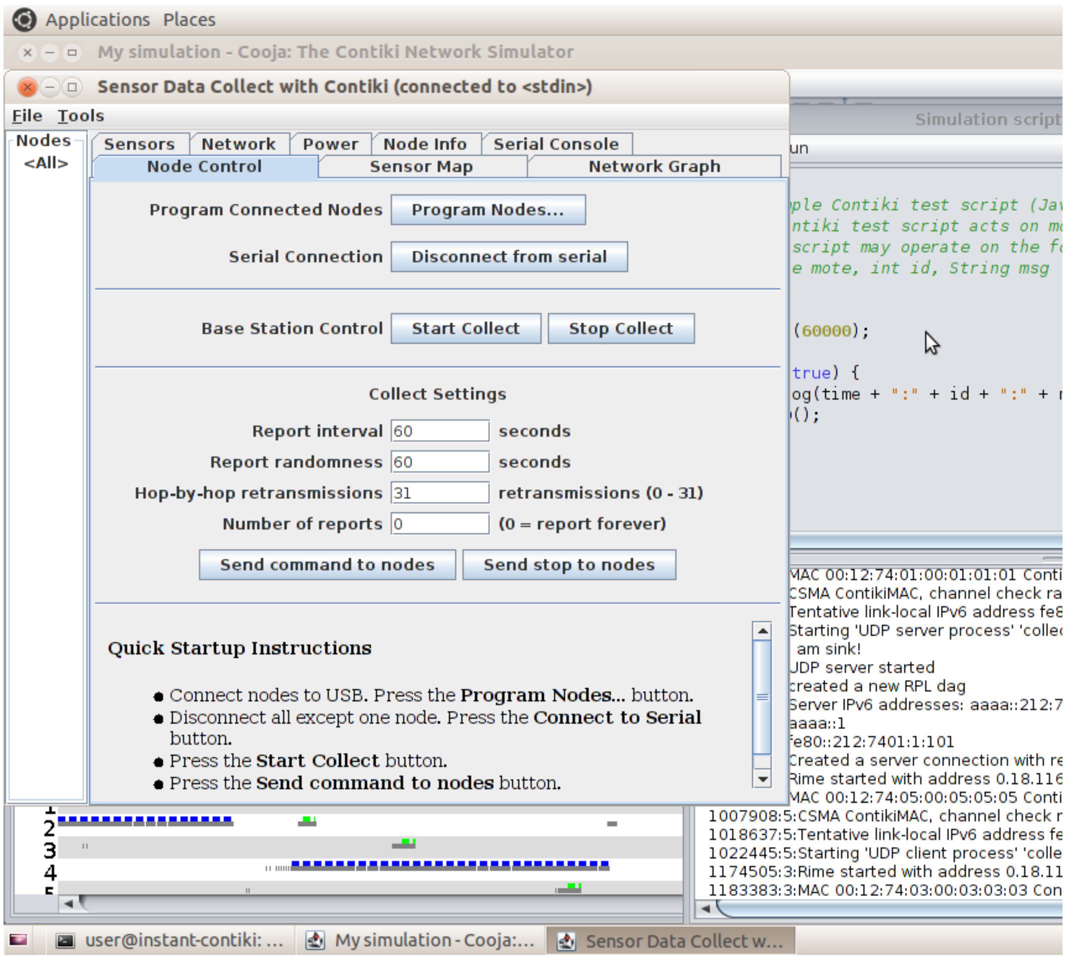
```
Time       Mote
00:14.311  ID:5
00:15.389  ID:3
00:15.426  ID:5
00:15.902  ID:4
00:16.371  ID:6
00:17.028  ID:4
00:17.640  ID:4
00:17.680  ID:5
```

```
666043:1:MAC 00:12:74:01:00:01:01:01 Contiki-2.6-900-ga6227e1 started. Node id is se
674798:1:CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26, CCA threshold -4
685527:1:Tentative link-local IPv6 address fe80:0000:0000:0000:0212:7401:0001:0101
689337:1:Starting 'UDP server process' 'collect common process'
690318:1:I am sink!
691853:1:UDP server started
695572:1:created a new RPL dag
699991:1:Server IPv6 addresses: aaaa::212:7401:1:101
701052:1:aaaa::1
703928:1:fe80::212:7401:1:101
710066:1:Created a server connection with remote address :: local/remote port 5688/87
990275:5:Rime started with address 0.18.116.5.0.5.5.5
999153:5:MAC 00:12:74:05:00:05:05:05 Contiki-2.6-900-ga6227e1 started. Node id is se
1007908:5:CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26, CCA threshold -
1018637:5:Tentative link-local IPv6 address fe80:0000:0000:0000:0212:7405:0005:0505
1022445:5:Starting 'UDP client process' 'collect common process'
1174505:3:Rime started with address 0.18.116.3.0.3.3.3
1183383:3:MAC 00:12:74:03:00:03:03:03 Contiki-2.6-900-ga6227e1 started. Node id is s
```

Figure 12. Cooja Simulation Script Editor

Cooja has sophisticated tools for collecting data from motes, however, it is not immediately obvious how to enable this in a simulation. For data collection in a network with a sink and several senders the collection should be performed from the viewpoint of the sink. This will then display data for all the senders in the network. To enable the Sensor Data Collect view there are two options. Firstly, right-click the sink mote, select **'Mote tools for Sky 1'** and then **'Collect View'**. Alternatively use the **'Tools'** dropdown menu, then **‘Collect View’** and finally **‘Sky 1’**. The result is a window as shown in in Figure 13. There are many details within this window which are for use with physical nodes as well as for simulation purposes. To quickly enable Sensor Collection for a Cooja Simulation simply click **‘Start Collect’**. This will take a few seconds to respond at which point ‘Send command to nodes’ should be clicked. The simulation can now be started as in previous examples.

It should be noted that all changes made regarding the Script Editor and Sensor Data Collect will be saved within the simulation providing the simulation is saved using the **‘File’** dropdown menu. It should also be noted that the simulation time must be long enough for the data collection to begin, with 1 minute not long enough. In this example a time of 5 minutes or 300000 milliseconds is used.



Applications
Places
My simulation - Cooja: The Contiki Network Simulator
Sensor Data Collect with Contiki (connected to <stdin>)
File
Tools
Nodes
<All>
Sensors
Network
Power
Node Info
Serial Console
Node Control
Sensor Map
Network Graph
Simulation script
un
user@instant-contiki: ...
My simulation - Cooja:...
Sensor Data Collect w...

Program Connected Nodes: Program Nodes...

Serial Connection: Disconnect from serial

Base Station Control: Start Collect | Stop Collect

Collect Settings

Report interval	60	seconds
Report randomness	60	seconds
Hop-by-hop retransmissions	31	retransmissions (0 - 31)
Number of reports	0	(0 = report forever)

Send command to nodes | Send stop to nodes

Quick Startup Instructions

- Connect nodes to USB. Press the **Program Nodes...** button.
- Disconnect all except one node. Press the **Connect to Serial** button.
- Press the **Start Collect** button.
- Press the **Send command to nodes** button.

```
ple Contiki test script (Ja
ntiki test script acts on m
script may operate on the f
e mote, int id, String msg

(60000);

true) {
og(time + ":" + id + ":" + 
();
```

```
MAC 00:12:74:01:00:01:01:01 Conti
CSMA ContikiMAC, channel check ra
Tentative link-local IPv6 address fe8
Starting 'UDP server process' 'colle
 am sink!
UDP server started
created a new RPL dag
Server IPv6 addresses: aaaa::212:7
aaaa::1
fe80::212:7401:1:101
Created a server connection with re
Rime started with address 0.18.116
MAC 00:12:74:05:00:05:05:05 Conti
1007908:5:CSMA ContikiMAC, channel check r
1018637:5:Tentative link-local IPv6 address fe
1022445:5:Starting 'UDP client process' 'colle
1174505:3:Rime started with address 0.18.11
1183383:3:MAC 00:12:74:03:00:03:03:03 Con
```

Figure 13. Cooja Sensor Data Collect

Once the simulation is complete a great amount of data will be available from the Sensor Collect View. Some of this information is displayed graphically as shown in Figure 14 which displays the average power consumption of the motes in the network. However, of potentially greater use is the Node Info as shown in Figure 15. This provides a significant amount of data which could then allow a user to produce their own analysis and graphs. It should be noted that in regard to Objective Functions, Contiki defaults to the use of MRHOF with the ETX metric, however, OF0 can also be used. This shall be demonstrated later in this document. The ability to observe RPL routing is discussed in the next section.

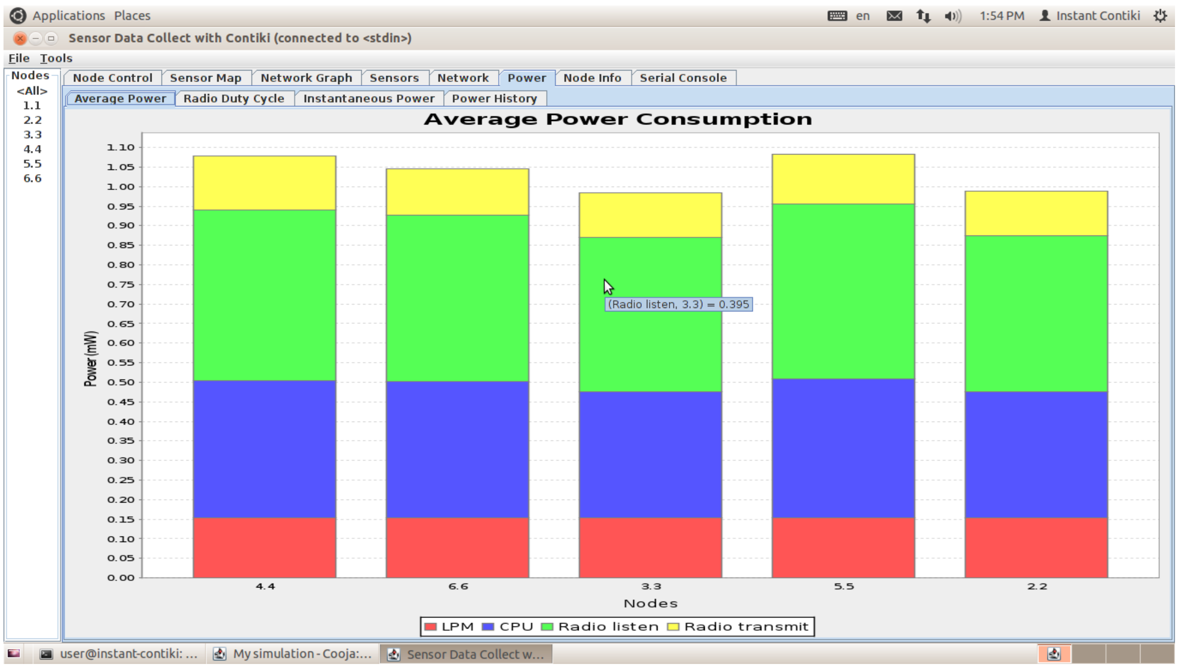
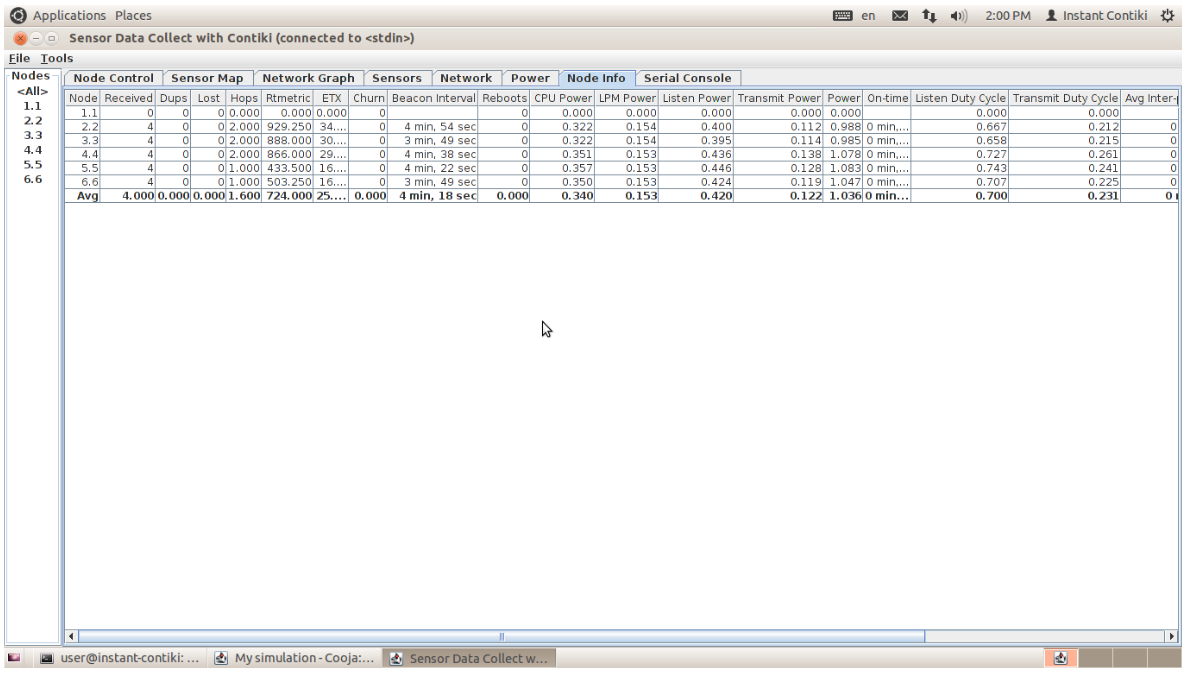


Figure 14. Cooja Average Power Consumption Graph



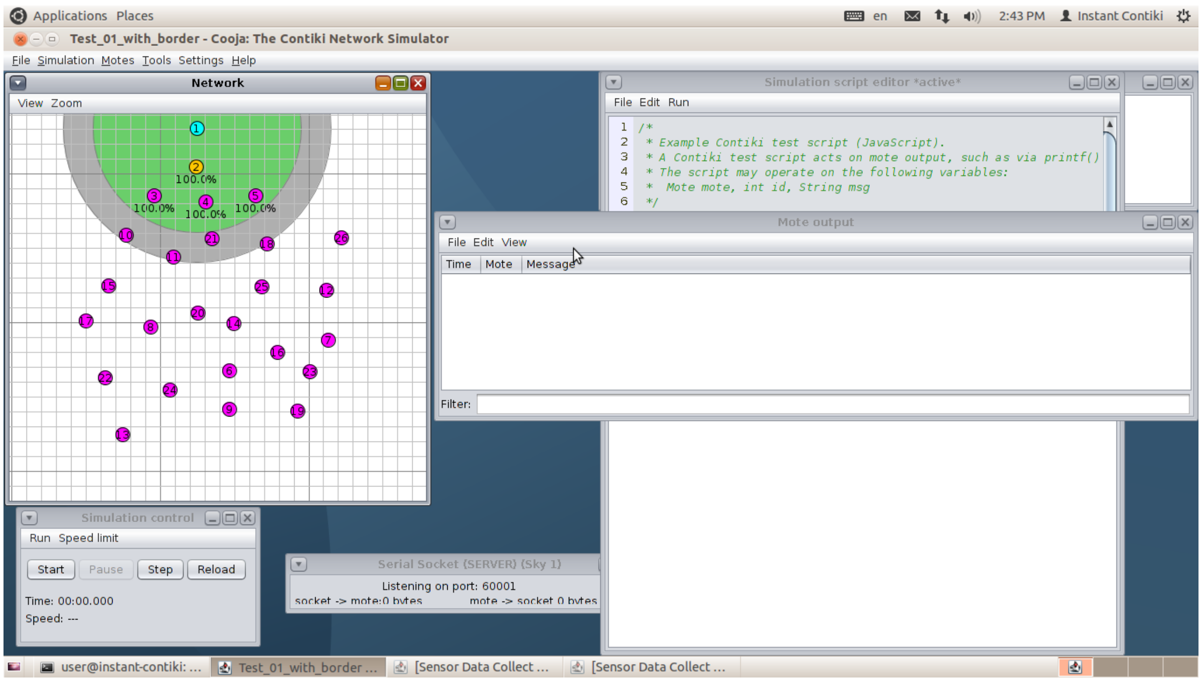
Node	Received	Dups	Lost	Hops	Rtmetric	ETX	Churn	Beacon Interval	Reboots	CPU Power	LPM Power	Listen Power	Transmit Power	Power	On-time	Listen Duty Cycle	Transmit Duty Cycle	Avg Inter-
1.1	0	0	0	0.000	0.000	0.000	0		0	0.000	0.000	0.000	0.000	0.000		0.000	0.000	
2.2	4	0	0	2.000	929.250	34....	0	4 min, 54 sec	0	0.322	0.154	0.400	0.112	0.988	0 min,...	0.667	0.212	0
3.3	4	0	0	2.000	888.000	30....	0	3 min, 49 sec	0	0.322	0.154	0.395	0.114	0.985	0 min,...	0.658	0.215	0
4.4	4	0	0	2.000	866.000	29....	0	4 min, 38 sec	0	0.351	0.153	0.436	0.138	1.078	0 min,...	0.727	0.261	0
5.5	4	0	0	1.000	433.500	16....	0	4 min, 22 sec	0	0.357	0.153	0.446	0.128	1.083	0 min,...	0.743	0.241	0
6.6	4	0	0	1.000	503.250	16....	0	3 min, 49 sec	0	0.350	0.153	0.424	0.119	1.047	0 min,...	0.707	0.225	0
Avg	**4.000**	**0.000**	**0.000**	**1.600**	**724.000**	**25....**	**0.000**	**4 min, 18 sec**	**0.000**	**0.340**	**0.153**	**0.420**	**0.122**	**1.036**	**0 min...**	**0.700**	**0.231**	**0**

Figure 15. Cooja Sensor Collect Node Info

1.4 IPv6 Routing

In Cooja it is possible to demonstrate routing and interaction with and between the motes in a network, such as by pinging them. To do this we can set up a simple network simulation. This has a Border Router bridged to the RPL network in order to assign IPv6 addresses and routing within the network. Once this is established the routing table within the Border Router can be demonstrated in order to prove the establishment of routes. It is then possible to ping even the most remote nodes within the network and also demonstrate those outwith radio range and therefore excluded from the network. What should be noted, however, is that memory and processor constraints can provide erratic results with regard to establishing a bridge to the Border Router.

The network setup for this demonstration can be seen in Figure 16. Mote 1 in this case is a Border Router mote which is compiled using firmware to be found in **/contiki/examples/ipv6/rpl-border-router/border-router.c**. To establish the bridge with the Border Router, first right click the Border Router mote 1 and select **'Mote tools for Sky1'** then **'Serial Socket (SERVER)'**. This will create a serial port on the Border Router which is accessible via UDP port number 60001 on the local machine [3].



```
/*
 * Example Contiki test script (JavaScript).
 * A Contiki test script acts on mote output, such as via printf()
 * The script may operate on the following variables:
 *  Mote mote, int id, String msg
 */
```

```
Listening on port: 60001
socket -> mote:0 bytes          mote -> socket 0 bytes
```

Figure 16. Cooja Border Router Demo

The simulation can now be started, however, a Terminal window should be open ready to enter the following commands which will establish the bridge to the Border Router:

```
user@instant-contiki:~/contiki$ cd tools
user@instant-contiki:~/contiki/tools$ make tunslip6
make: `tunslip6' is up to date.
user@instant-contiki:~/contiki/tools$ sudo ./tunslip6 -a 127.0.0.1 aaaa::1/64
[sudo] password for user:
```

This will the return the following, meaning a bridge has been established and IPv6 addresses assigned with prefix **aaaa::/64**. The Border Router's IPv6 address of **aaaa::212:7401:1:101** can be seen.

```
slip connected to ``127.0.0.1:60001''
opened tun device ``/dev/tun0''
ifconfig tun0 inet `hostname` up
ifconfig tun0 add aaaa::1/64
ifconfig tun0 add fe80::0:0:0:1/64
ifconfig tun0

tun0     Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
         inet addr:127.0.1.1  P-t-P:127.0.1.1  Mask:255.255.255.255
         inet6 addr: fe80::1/64 Scope:Link
         inet6 addr: aaaa::1/64 Scope:Global
         UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:500
         RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

*** Address:aaaa::1 => aaaa:0000:0000:0000
Got configuration message of type P
Setting prefix aaaa::
Server IPv6 addresses:
 aaaa::212:7401:1:101
 fe80::212:7401:1:101
```
[3]

Figure 17 shows the network with IPv6 Addresses now established and traffic moving between all nodes and the Border Router.

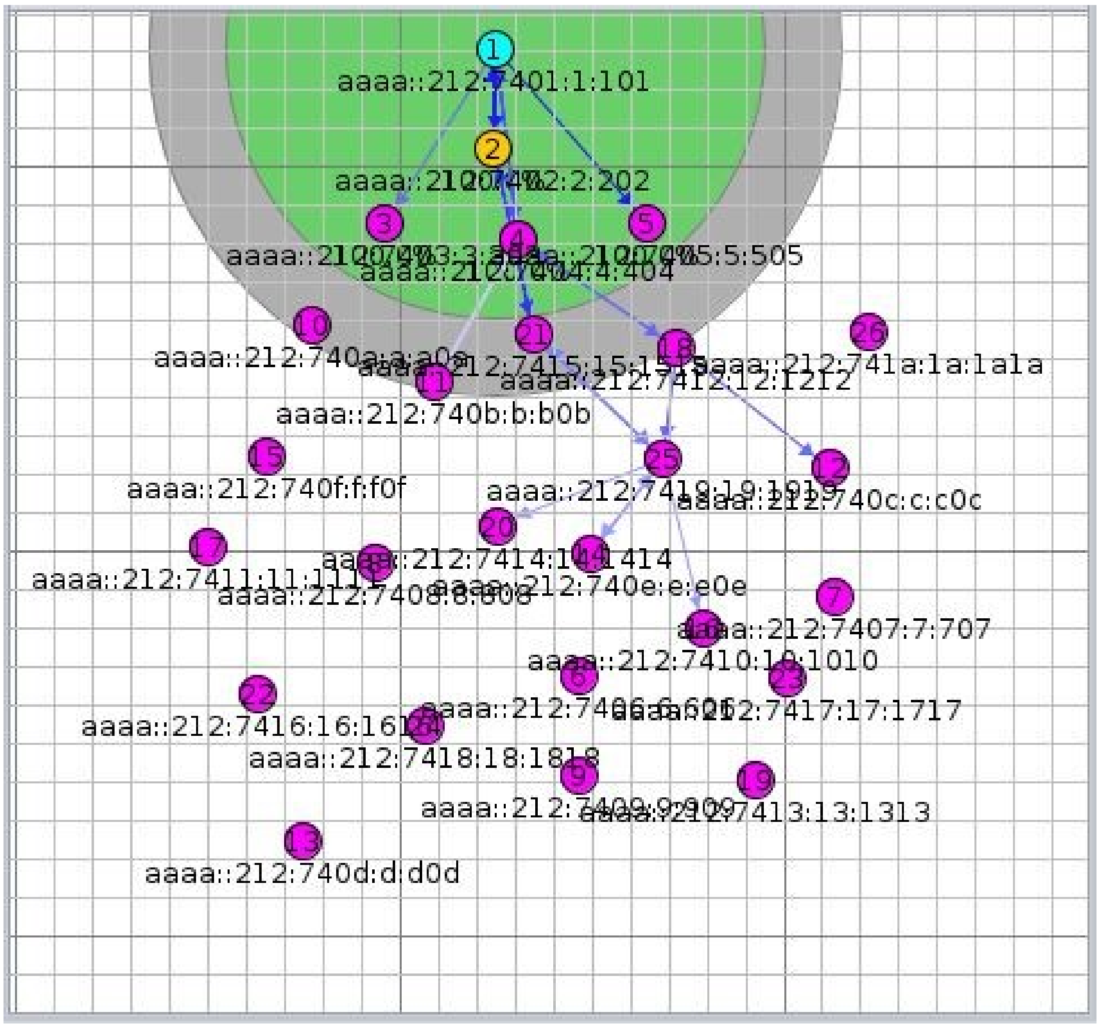


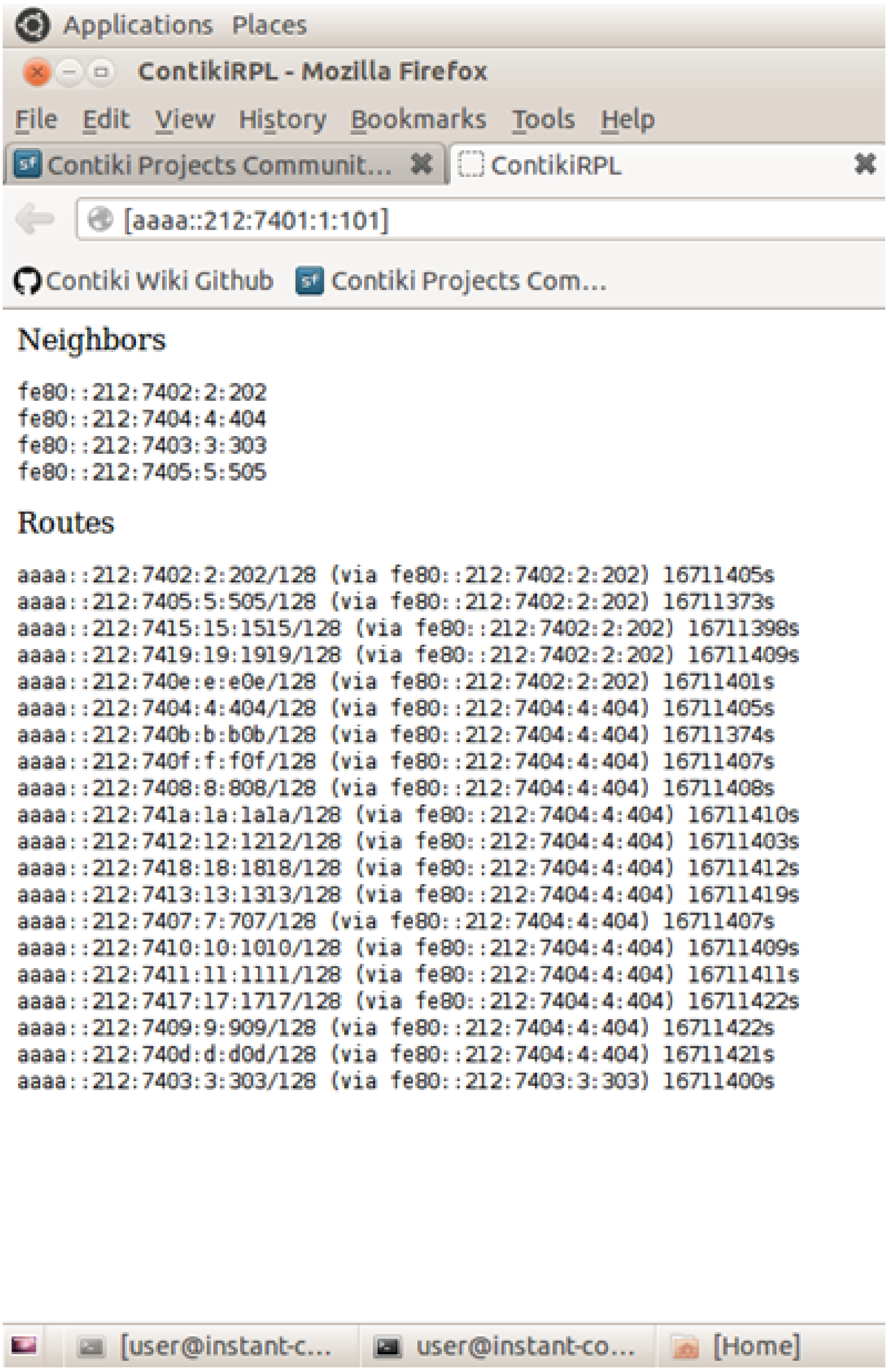
Figure 17. Cooja Border Router Demo with IPv6 Addressing

While the simulation is running, the bridge can be used to Ping motes in the network as follows. Although not all are successful due to the volatility of the network:

```
user@instant-contiki:~$ ping6 -c 5 -s 8 aaaa::212:7401:1:101
PING aaaa::212:7401:1:101(aaaa::212:7401:1:101) 8 data bytes
16 bytes from aaaa::212:7401:1:101: icmp_seq=1 ttl=64 time=26.8 ms
16 bytes from aaaa::212:7401:1:101: icmp_seq=2 ttl=64 time=21.2 ms
16 bytes from aaaa::212:7401:1:101: icmp_seq=3 ttl=64 time=19.1 ms
16 bytes from aaaa::212:7401:1:101: icmp_seq=4 ttl=64 time=31.8 ms
16 bytes from aaaa::212:7401:1:101: icmp_seq=5 ttl=64 time=15.9 ms

--- aaaa::212:7401:1:101 ping statistics ---
```

```
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/mdev = 15.969/22.996/31.819/5.660 ms

user@instant-contiki:~$ ping6 -c 5 -s 8 aaaa::212:7413:13:1313
PING aaaa::212:7413:13:1313(aaaa::212:7413:13:1313) 8 data bytes
16 bytes from aaaa::212:7413:13:1313: icmp_seq=1 ttl=59 time=4126 ms
16 bytes from aaaa::212:7413:13:1313: icmp_seq=2 ttl=59 time=7826 ms
16 bytes from aaaa::212:7413:13:1313: icmp_seq=3 ttl=59 time=6816 ms
16 bytes from aaaa::212:7413:13:1313: icmp_seq=4 ttl=59 time=5841 ms
16 bytes from aaaa::212:7413:13:1313: icmp_seq=5 ttl=59 time=5043 ms

--- aaaa::212:7413:13:1313 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4018ms
rtt min/avg/max/mdev = 4126.973/5931.060/7826.502/1298.239 ms, pipe 5

user@instant-contiki:~$ ping6 -c 5 -s 8 aaaa::212:7404:4:404
PING aaaa::212:7404:4:404(aaaa::212:7404:4:404) 8 data bytes
16 bytes from aaaa::212:7404:4:404: icmp_seq=1 ttl=63 time=177 ms
16 bytes from aaaa::212:7404:4:404: icmp_seq=2 ttl=63 time=139 ms
16 bytes from aaaa::212:7404:4:404: icmp_seq=3 ttl=63 time=666 ms
16 bytes from aaaa::212:7404:4:404: icmp_seq=4 ttl=63 time=308 ms
16 bytes from aaaa::212:7404:4:404: icmp_seq=5 ttl=63 time=481 ms

--- aaaa::212:7404:4:404 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 139.917/354.555/666.296/196.464 ms
```

Finally, tt is also possible to display the Border Router's routing table by opening up a web page and entering the IPv6 address of the Border Router as in Figure 18. This shows a snapshot of routes established, however, these may be in a constant state of flux, especially with regard to motes on the fringes of the network.



Neighbors

```
fe80::212:7402:2:202
fe80::212:7404:4:404
fe80::212:7403:3:303
fe80::212:7405:5:505
```

Routes

```
aaaa::212:7402:2:202/128 (via fe80::212:7402:2:202) 16711405s
aaaa::212:7405:5:505/128 (via fe80::212:7402:2:202) 16711373s
aaaa::212:7415:15:1515/128 (via fe80::212:7402:2:202) 16711398s
aaaa::212:7419:19:1919/128 (via fe80::212:7402:2:202) 16711409s
aaaa::212:740e:e:e0e/128 (via fe80::212:7402:2:202) 16711401s
aaaa::212:7404:4:404/128 (via fe80::212:7404:4:404) 16711405s
aaaa::212:740b:b:b0b/128 (via fe80::212:7404:4:404) 16711374s
aaaa::212:740f:f:f0f/128 (via fe80::212:7404:4:404) 16711407s
aaaa::212:7408:8:808/128 (via fe80::212:7404:4:404) 16711408s
aaaa::212:741a:1a:1a1a/128 (via fe80::212:7404:4:404) 16711410s
aaaa::212:7412:12:1212/128 (via fe80::212:7404:4:404) 16711403s
aaaa::212:7418:18:1818/128 (via fe80::212:7404:4:404) 16711412s
aaaa::212:7413:13:1313/128 (via fe80::212:7404:4:404) 16711419s
aaaa::212:7407:7:707/128 (via fe80::212:7404:4:404) 16711407s
aaaa::212:7410:10:1010/128 (via fe80::212:7404:4:404) 16711409s
aaaa::212:7411:11:1111/128 (via fe80::212:7404:4:404) 16711411s
aaaa::212:7417:17:1717/128 (via fe80::212:7404:4:404) 16711422s
aaaa::212:7409:9:909/128 (via fe80::212:7404:4:404) 16711422s
aaaa::212:740d:d:d0d/128 (via fe80::212:7404:4:404) 16711421s
aaaa::212:7403:3:303/128 (via fe80::212:7403:3:303) 16711400s
```

Figure 18. Border Router Routing Table

1.5 Adjusting Objective Functions

One of the key features of RPL is the use of different Objective Functions in the establishment of routes. Cooja can be manipulated to use either of MRHOF or OF0 quite easily, however, it involves careful manipulation of the source code. In order to swap between the two objective functions the following instructions must be followed.

/home/user/contiki/core/net/rpl

Change **Makefile.rpl** as follows:

```
CONTIKI_SOURCEFILES += rpl.c rpl-dag.c rpl-icmp6.c rpl-timers.c \
        rpl-mrhof.c rpl-ext-header.c
```

This should be left unchanged to use **MRHOF** or changed to **rpl-of0.c** in the event that **Objective Function Zero** is to be used.

Change **rpl-conf.h** as follows:

```
/*
 * The objective function used by RPL is configurable through the
 * RPL_CONF_OF parameter. This should be defined to be the name of an
 * rpl_of object linked into the system image, e.g., rpl_of0.
 */
#ifdef RPL_CONF_OF
#define RPL_OF RPL_CONF_OF
#else
/* ETX is the default objective function. */
#define RPL_OF rpl_mrhof
#endif /* RPL_CONF_OF */
```

This should be left unchanged to use **MRHOF** or changed to **rpl-of0.c** in the event that **Objective Function Zero** is to be used.

As you will no doubt be aware, the use of MRHOF brings the possibility of many different metrics. When using MRHOF in Cooja the metric will default to ETX unless the Metric Container is changed.

Currently, the only alternative to ETX available in Cooja is the Energy metric. To utilise this, change **rpl-conf.h** as follows:

```
/*
 * Select routing metric supported at runtime. This must be a valid
 * DAG Metric Container Object Type (see below). Currently, we only
 * support RPL_DAG_MC_ETX and RPL_DAG_MC_ENERGY.
 * When MRHOF (RFC6719) is used with ETX, no metric container must
 * be used; instead the rank carries ETX directly.
 */
#ifdef RPL_CONF_DAG_MC
#define RPL_DAG_MC RPL_CONF_DAG_MC
#else
#define RPL_DAG_MC RPL_DAG_MC_NONE
#endif /* RPL_CONF_DAG_MC */
/*
```

This should be left unchanged if ETX is to be used. In the event that the Energy metric is to be used this should be changed to **RPL_DAG_MC_ENERGY**.

References

[1] Contiki, “Contiki: The Open Source Operating System for the Internet of Things,” 2015. [Online]. Available: http://www.contiki-os.org/. [Accessed: 09-Nov-2015].

[2] VMware, “VMware Virtualization for Desktop & Server, Application, Public & Hybrid Clouds,” 2016. [Online]. Available: http://www.vmware.com/uk. [Accessed: 19-Feb-2016].

[3] A. Sehgal, “Using the Contiki Cooja Simulator,” *Jacobs University Bremen*, 2013. [Online]. Available: http://cnds.eecs.jacobs-university.de/courses/iotlab-2013/cooja.pdf. [Accessed: 15-Feb-2016].