

IOT EMULATION WITH COOJA

BA BAGULA & ZENVILLE ERASMUS

ISAT LABORATORY

DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF THE WESTERN CAPE (UWC)

CAPE TOWN – SOUTH AFRICA

ICTP-IoT Workshop, Trieste 16-27 March 2015

Outline

□ What is Cooja

- Emulator vs Simulator
- Main steps
- Hello-world

□ More examples

- UDP-RPL/broadcast
- Z1 sensors
- Sense/Send/Blink
- UDP-RPL/Unicast

□ Energy monitoring

- Timeline
- Energest
- Powertrace
- PowerTracker

□ Networking protocols

- RPL
- LIBP
- Multi-sink

Cooja is an emulator

- According to different sources, an emulator is:
 - ▣ a hardware or software system that enables one computer system (called the *host*) to *behave like another* computer system (called the *guest*): e.g. Cooja enabling your laptop to behave like a Z1 mote.
 - ▣ a system that typically *enables* the host system *to run software or use peripheral devices* designed for the guest system: e.g. Cooja enabling your laptop to run the RPL protocol, LIBP and/or other IoT protocols of interest .

Cooja is an emulator

- According to different sources, an emulator is:
 - ▣ a system that *behaves exactly like* the guest system, and abides by all of the rules of the system being emulated, but operating in a different environment to the environment of the original emulated system.
 - ▣ a *complete replication* of the guest system, right down to being binary compatible with the emulated system's inputs and outputs.

Cooja is not a simulator

- According to different sources, a simulator is:
 - ▣ a hardware or software that that *enables* one computer system (called the *host*) *to behave like another* computer system (called the *guest*), *but* is implemented in an entirely *different way* : e.g. A flight simulator gives you the feeling of flying an airplane, but you are completely disconnected from the reality of flying the plane, and you can bend or break those rules as you see fit. e.g. Fly an Airbus A380 upside down between London and Sydney without breaking it.

Cooja is not a simulator

- According to different sources, a simulator is:
 - ▣ a system that *provides the basic behaviour* of a system *but may not* necessarily *abide by all of the rules* of the system being simulated.
 - ▣ A system designed to *recreate the operation or behaviour* of the guest system. The underlying principles can be the same as the original or different.

What is Cooja?

- ❑ Cooja is a Contiki network emulator
 - An extensible Java-based simulator capable of emulating Tmote Sky (and other) nodes
- ❑ The code to be executed by the node is the exact same firmware you may upload to physical nodes
- ❑ Allows large and small networks of motes to be simulated
- ❑ Motes can be emulated at the hardware level
 - Slower but allows for precise inspection of system behaviour
- ❑ Motes can also be emulated at a less detailed level
 - Faster and allows simulation of larger networks

Cooja (continued)

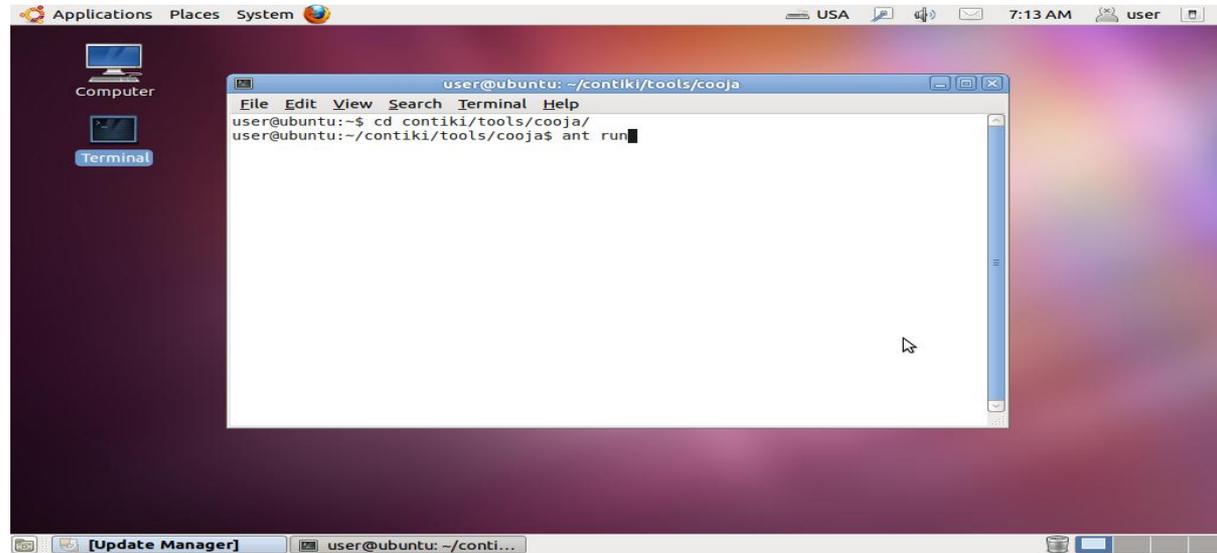
- ❑ Cooja is a highly useful tool for Contiki development
 - ❑ It allows developers to test their code and systems long before running it on the target hardware
 - ❑ Developers regularly set up new simulations to
 - ❑ debug their software
 - ❑ to verify the behaviour of their systems

Main steps

1. Open a terminal window to start Cooja
2. Create a new simulation to run Contiki in simulation and wait for Cooja to start and compile itself
3. Set simulation options
4. Create a new mote type
5. Add motes to the simulation
6. Open a terminal Cooja is a highly useful tool for Contiki development
 1. It allows developers to test their code and systems long before running it on the target hardware
 2. Developers regularly set up new simulations to
 1. debug their software

1. Starting Cooja

- ❑ Open a terminal window

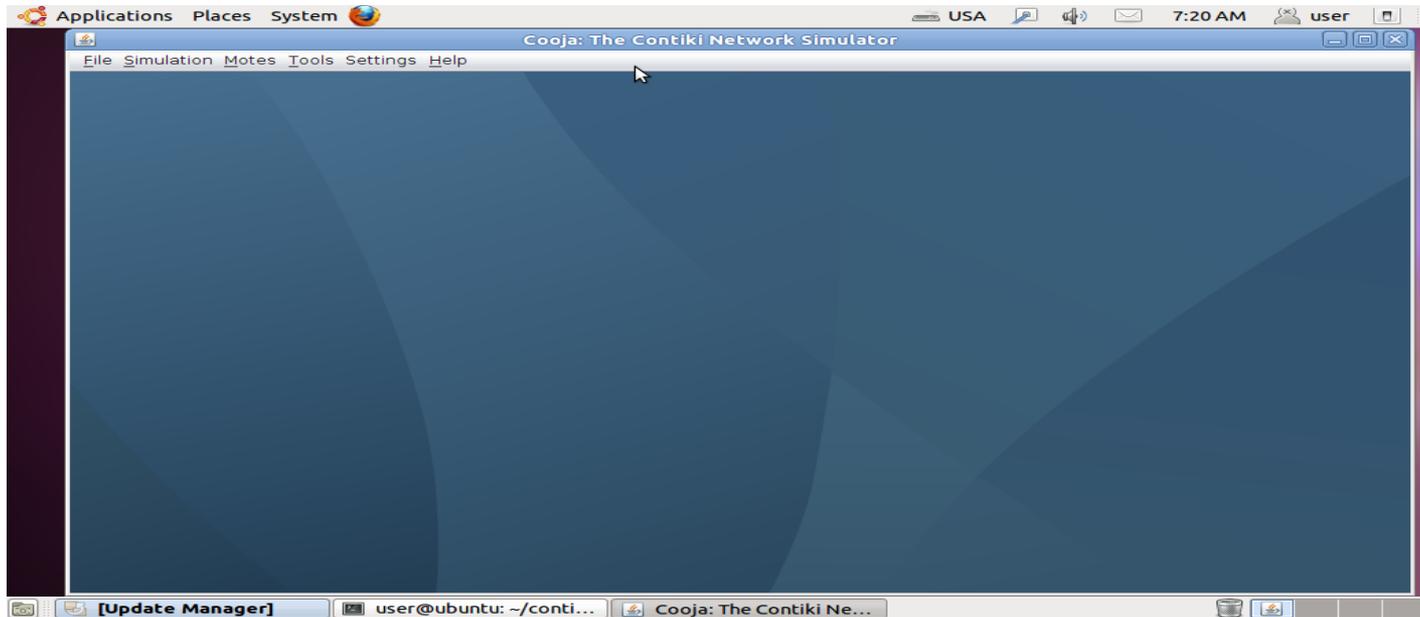


To start Cooja, first **open a terminal window**.

- ✓ **cd contiki/tools/cooja (Cooja directory)**
- ✓ **start cooja by issuing ant run**

Waiting for Cooja to start

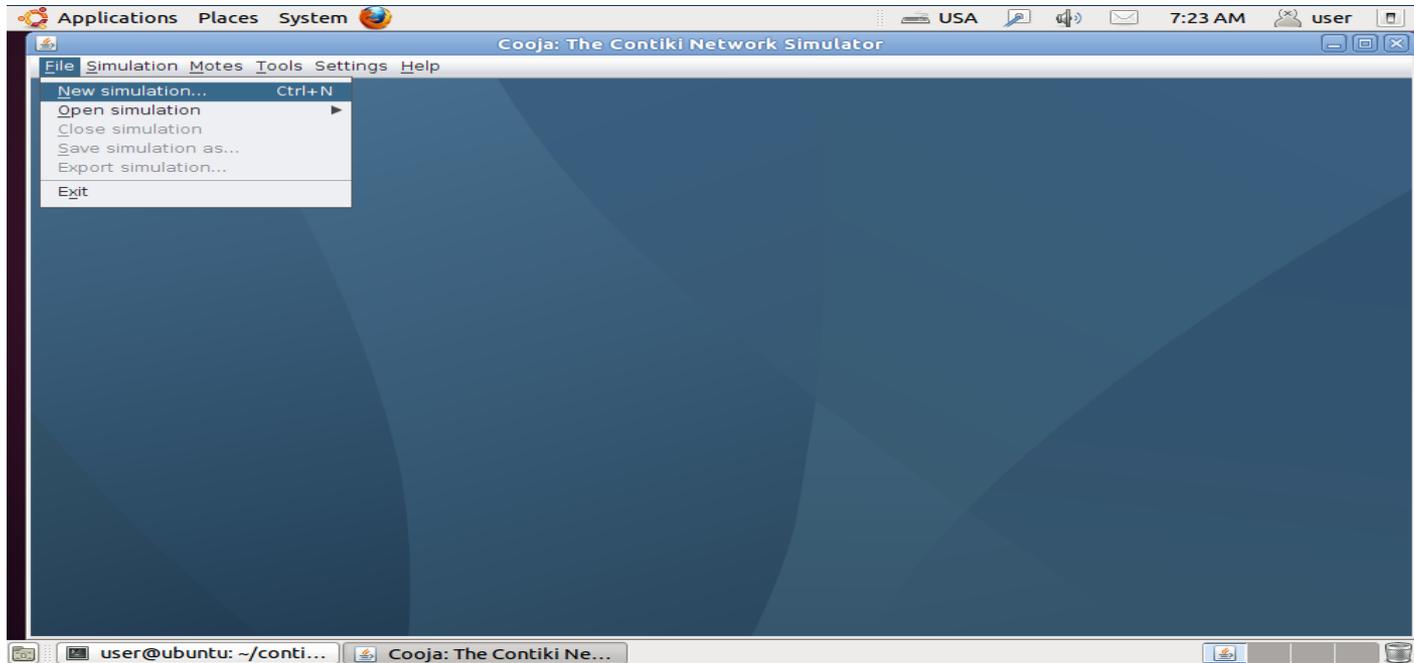
- ❑ When Cooja first starts, it will compile itself. This may take some time



When Cooja is compiled, it will start with a blue empty window.

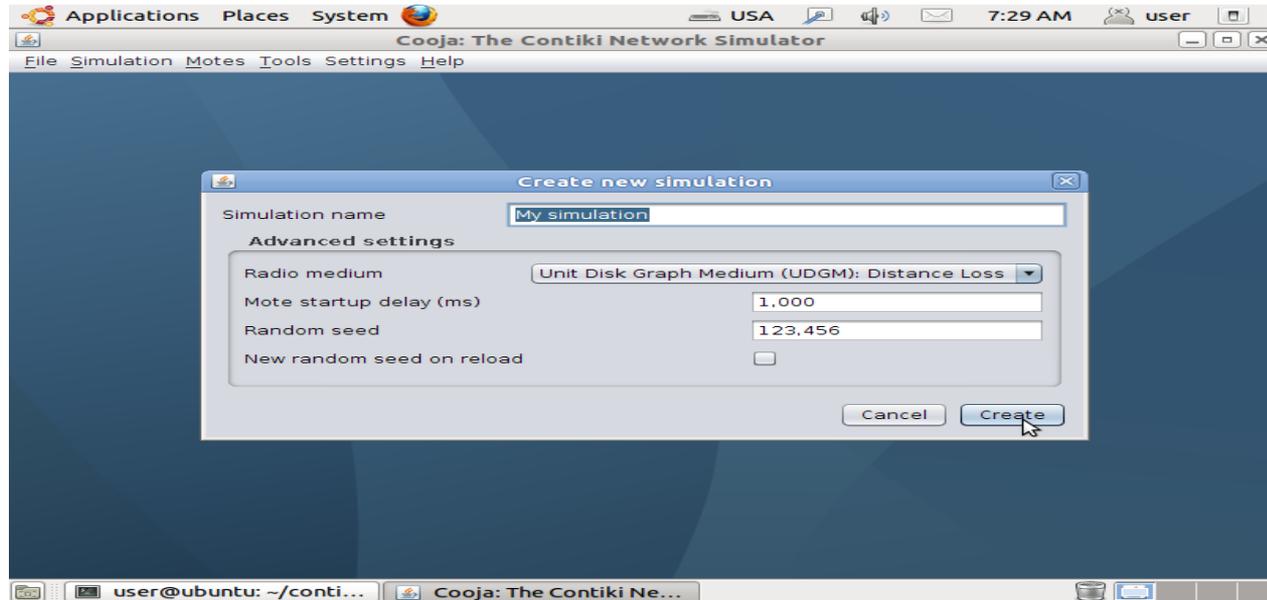
2. Create a new simulation

- ❑ Click the **File** menu and click **New simulation...**



3. Set simulation options

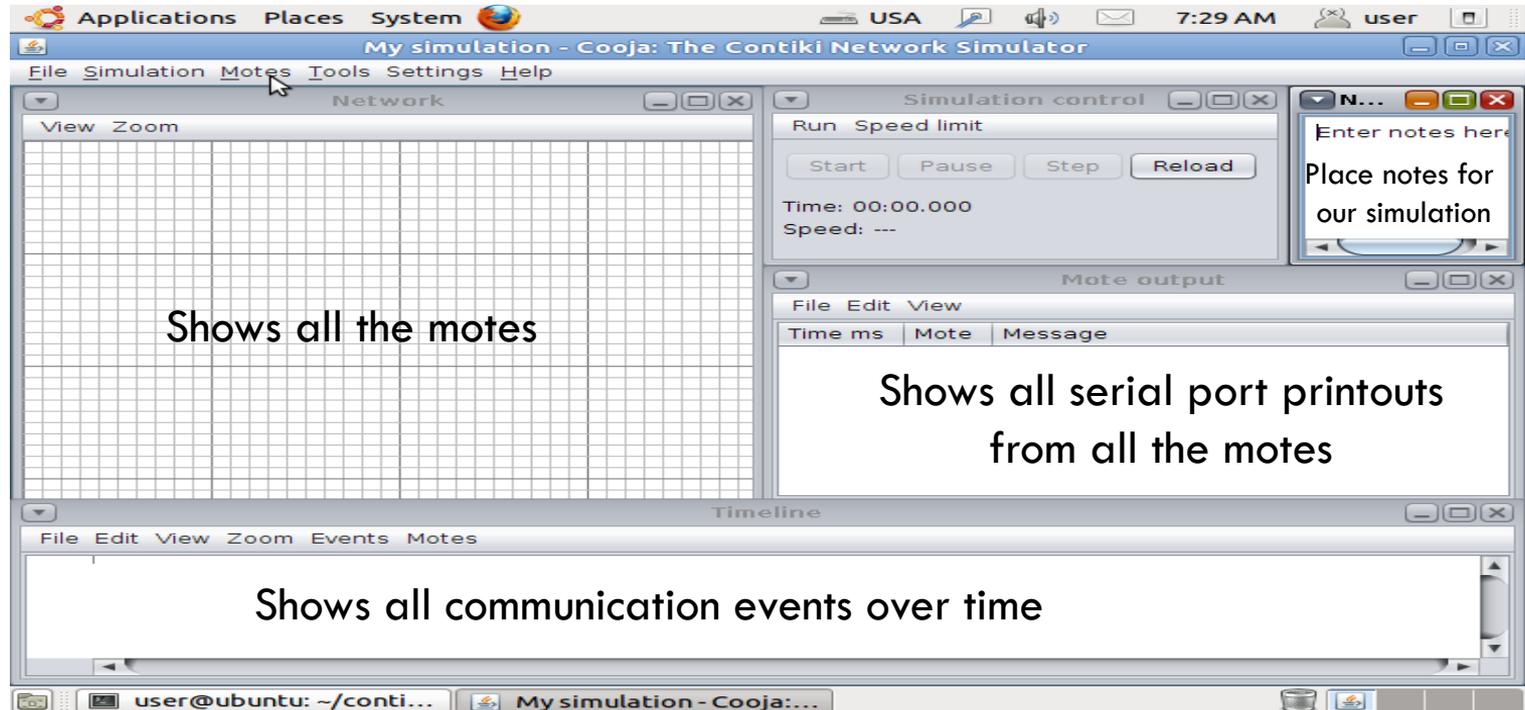
- ❑ Cooja now opens up the **Create new simulation** dialog. Either change the dialog name or stick with My simulation.



- ✓ Click the **Create** button.

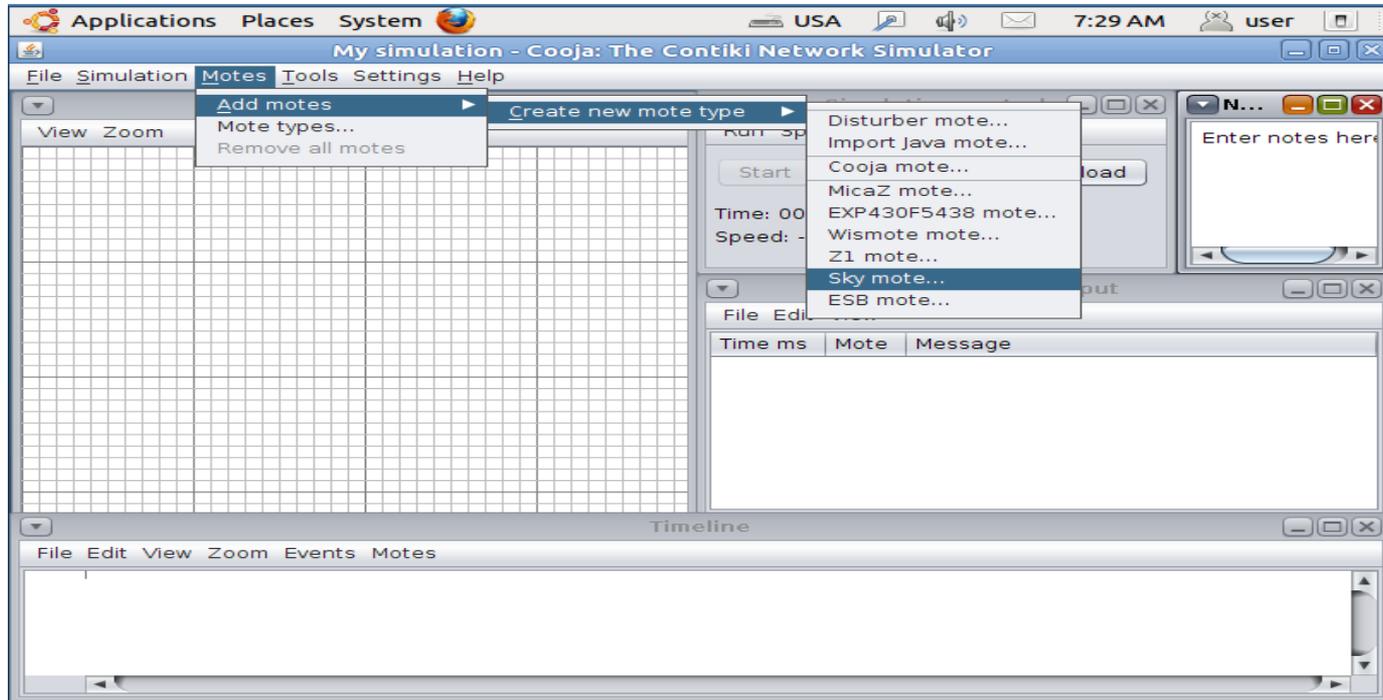
Simulation windows

- ❑ Cooja brings up the new simulation.



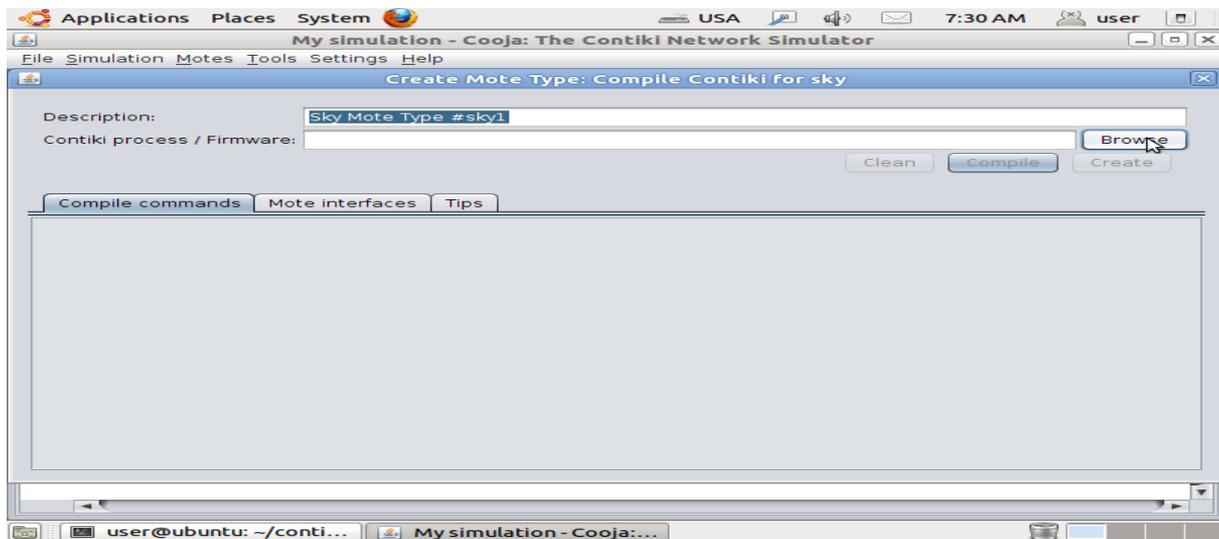
4. Add notes to the simulation

□ Add notes



Create a new mote type

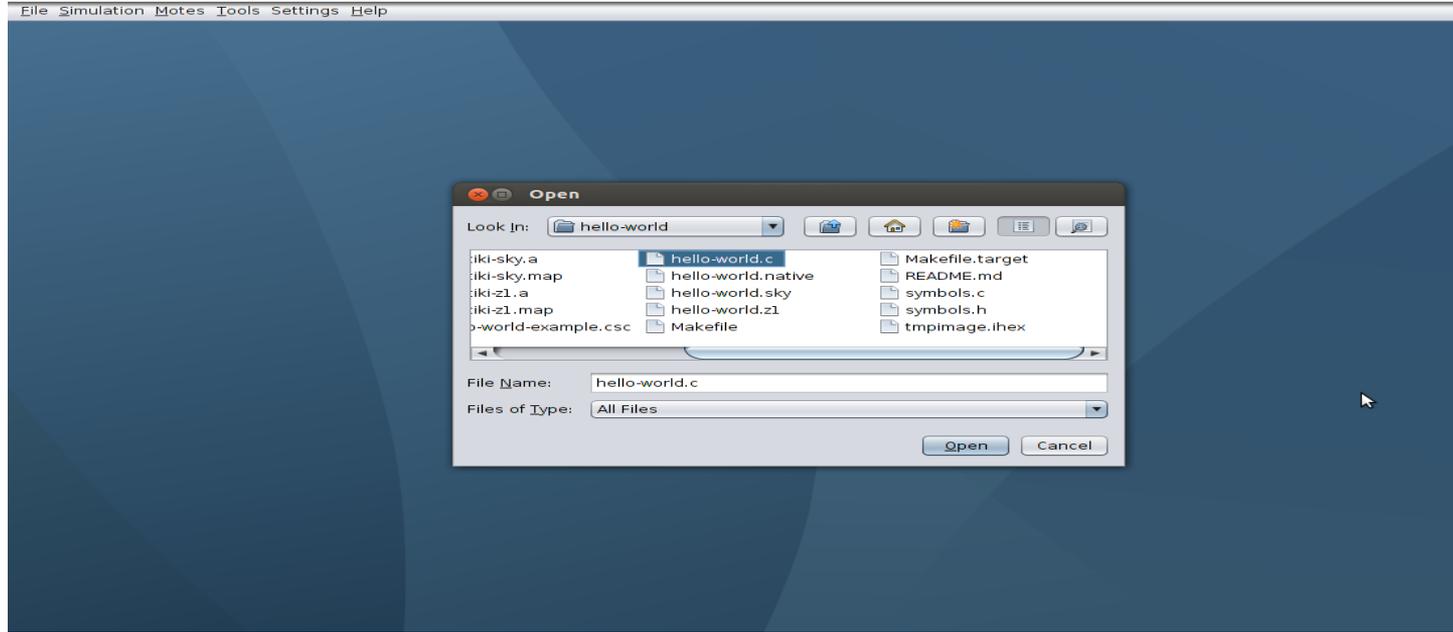
- ❑ Cooja opens up the **Create Mote Type** dialog



- ✓ choose a name for our mote type
- ✓ choose the Contiki application that our mote type will run

5. Find Contiki Application

- ❑ Hello World `/opt/contiki-2.7/examples/hello-world`



- ✓ Specify application C source file → **Open**

6. Compile the Contiki application

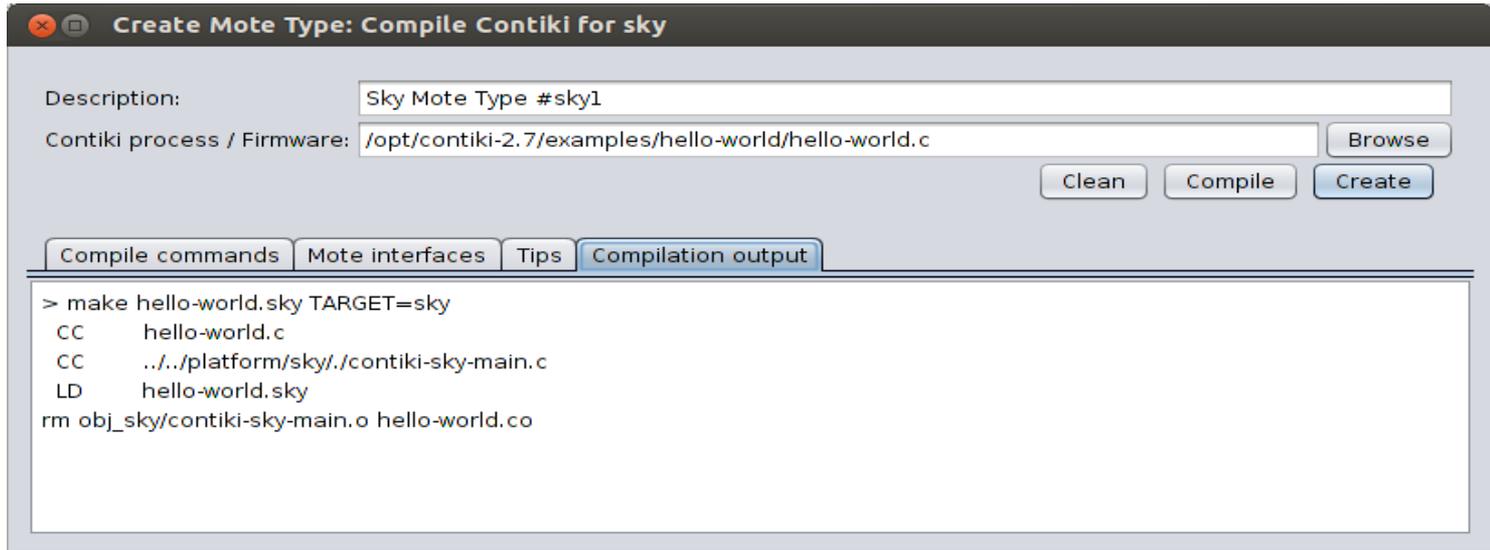
- ❑ Cooja will verify that the selected Contiki application compiles for the platform that we have selected



- ✓ Click on the **Compile** button. This will take some time...
- ✓ Compilation output will show up in the bottom white panel.

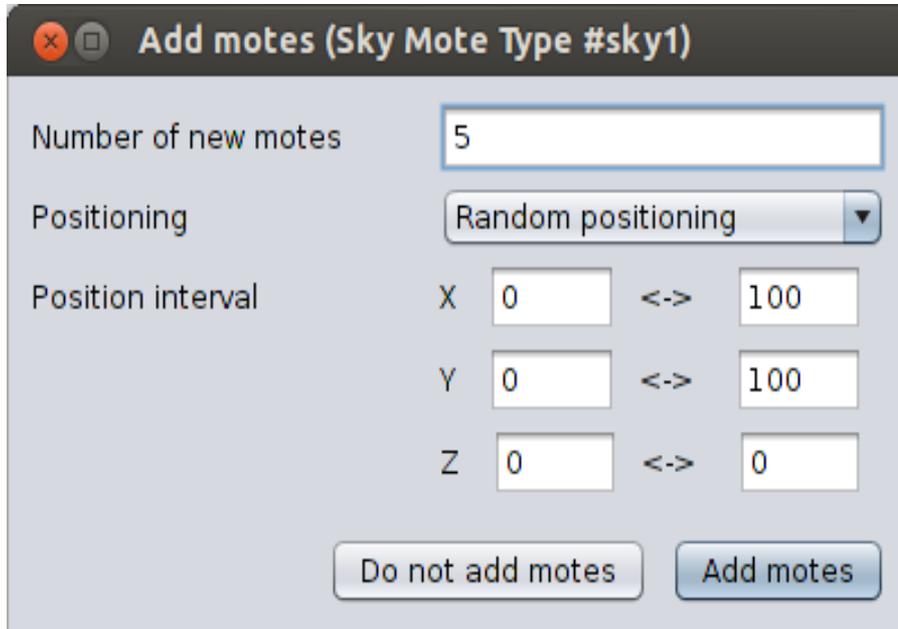
7. Create the mote type

- Click on the **Create** button to create the mote type. The window will close.



8. Add notes to simulation

- ❑ Add notes by changing the number of notes in the **Number of notes** field to 5.



Add notes (Sky Mote Type #sky1)

Number of new notes:

Positioning:

Position interval:

X	<input type="text" value="0"/>	<->	<input type="text" value="100"/>
Y	<input type="text" value="0"/>	<->	<input type="text" value="100"/>
Z	<input type="text" value="0"/>	<->	<input type="text" value="0"/>

- ✓ Click on **Add notes** to add notes to the simulation

9. Start the simulation

- ❑ The 5 added notes are now seen in the simulation window.

The screenshot displays the simulation software interface with several windows:

- Network:** A grid showing 5 nodes labeled 1 through 5.
- Simulation control:** A panel with buttons for Start, Pause, Step, and Reload. It shows Time: 00:00.000 and Speed: ---.
- Notes:** A text area labeled "Enter notes here".
- PowerTracker: 5 notes:** A table showing power statistics for 5 notes.
- Mote output:** A table with columns for Time, Mote, and Message.
- Timeline showing 5 notes:** A vertical timeline with 5 entries.

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 1	NaN%	NaN%	NaN%
Sky 2	NaN%	NaN%	NaN%
Sky 3	NaN%	NaN%	NaN%
Sky 4	NaN%	NaN%	NaN%
Sky 5	NaN%	NaN%	NaN%
AVERAGE	NaN%	NaN%	NaN%

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

10. Pause the simulation

The screenshot displays a simulation environment with several windows:

- Network:** A grid-based network diagram with five nodes labeled "Hello_world" and numbered 1 through 5.
- Simulation control:** A panel with "Start", "Pause", "Step", and "Reload" buttons. It shows "Time: 00:53.795" and "Speed: ---".
- PowerTracker: 5 notes:** A table showing power usage for five nodes.
- Mote output:** A log window showing the simulation's output, with the "View" menu open.
- Timeline showing 5 notes:** A timeline view showing the sequence of events for each node.

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 1	0.68%	0.00%	0.00%
Sky 2	0.69%	0.00%	0.00%
Sky 3	0.68%	0.00%	0.00%
Sky 4	0.68%	0.00%	0.00%
Sky 5	0.68%	0.00%	0.00%
AVERAGE	0.68%	0.00%	0.00%

```
Time      Mote      Message
00:00.650 ID:1      MAC 01:00:00:00:00:00:00:00 Contiki 2.7 started. Node id is set to 1.
00:00.659 ID:1      CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26
00:00.662 ID:1      Starting 'Hello world process'
00:00.663 ID:1      Hello, world
00:00.976 ID:5      Rime started with address 5.0
00:00.983 ID:5      MAC 05:00:00:00:00:00:00:00 Contiki 2.7 started. Node id is set to 5.
00:00.992 ID:5      CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26
00:00.995 ID:5      Starting 'Hello world process'
00:00.996 ID:5      Hello, world
00:01.160 ID:3      Rime started with address 3.0
00:01.168 ID:3      MAC 03:00:00:00:00:00:00:00 Contiki 2.7 started. Node id is set to 3.
00:01.177 ID:3      CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26
00:01.179 ID:3      Starting 'Hello world process'
00:01.180 ID:3      Hello, world
```

✓ **View** → Select Log output: `printf()`'s

11. Get some statistics

- ❑ Mote output window
 - ❑ Printouts from the simulated motes
- ❑ Network window
 - ❑ Shows ongoing network communication
- ❑ Timeline
 - ❑ Shows communication and radio events over time
 - ❑ The small gray lines are ContikiMAC periodically waking up the radio
- ❑ Pause
 - ❑ Click the **Pause** button to pause the simulation

More examples

simple-udp-rpl/broadcast-example.c

The screenshot displays a network simulation interface with several panels:

- Network:** A grid showing a star topology with 10 nodes (labeled 1-10). Node 1 is the central hub, and nodes 2-10 are connected to it by blue lines.
- Simulation control:** A panel with buttons for 'Start', 'Pause', 'Step', and 'Reload'. It shows 'Time: 03:39.035' and 'Speed: ---'.
- Notes:** A text area for entering notes.
- PowerTracker: 10 motes:** A table showing power usage for each mote. Mote 5 is highlighted in red.
- Mote output:** A log window showing the following messages:

```
03:38.869 ID:1 Sending broadcast
03:38.908 ID:2 Data received on port 1234 from port 1234 with length 4
03:38.923 ID:6 Data received on port 1234 from port 1234 with length 4
03:38.948 ID:3 Data received on port 1234 from port 1234 with length 4
03:38.951 ID:7 Data received on port 1234 from port 1234 with length 4
03:38.998 ID:8 Data received on port 1234 from port 1234 with length 4
03:39.008 ID:9 Data received on port 1234 from port 1234 with length 4
03:39.012 ID:5 Data received on port 1234 from port 1234 with length 4
```
- Timeline showing 10 motes:** A timeline view showing events for each of the 10 motes.

More examples

❑ Z1 sensors

The screenshot displays a simulation environment with several windows:

- Network:** A grid showing sensor locations. A coordinate system is at the top left with 'x: 0 y: 0 z: 0'. Five sensors are marked with circled numbers 1 through 5. Sensor 1 is at the origin. Other sensors are labeled with their values: Light = 14, Rel. humidity: 2650%, Temp = 0.0000, and Battery: 95 (0.115 mV).
- Simulation control:** Contains 'Run Speed limit' buttons (Start, Pause, Step, Reload), 'Time: 04:30.323', and 'Speed: ---'.
- Notes:** A text area with the placeholder 'Enter notes here'.
- PowerTracker: 5 notes:** A table showing power consumption for five Z1 sensors.
- Mote output:** A log window showing messages from ID:2 motes, primarily reporting battery levels.
- Timeline showing 5 notes:** A timeline view with five tracks labeled 1 through 5, showing event markers.

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Z1 1	0.65%	0.00%	0.00%
Z1 2	0.65%	0.00%	0.00%
Z1 3	0.64%	0.00%	0.00%
Z1 4	0.65%	0.00%	0.00%
Z1 5	0.65%	0.00%	0.00%
AVERAGE	0.65%	0.00%	0.00%

Time	Mote	Message
04:30.282	ID:2	Battery: 223 (0.272 mV)
04:30.287	ID:2	Battery: 85 (0.103 mV)
04:30.291	ID:2	Battery: 10 (0.012 mV)
04:30.296	ID:2	Battery: 205 (0.250 mV)
04:30.300	ID:2	Battery: 88 (0.107 mV)
04:30.304	ID:2	Battery: 6 (0.007 mV)
04:30.309	ID:2	Battery: 8 (0.009 mV)
04:30.313	ID:2	Battery: 10 (0.012 mV)
04:30.318	ID:2	Battery: 205 (0.250 mV)
04:30.322	ID:2	Battery: 95 (0.115 mV)

More examples

- ❑ Sense, send and blink with receive and blink

The screenshot displays a network simulation interface with several panels:

- Network:** A diagram showing five nodes (1-5) connected in a network topology. Node 1 is at the top, connected to nodes 2 and 3. Node 2 is connected to node 3. Node 3 is connected to nodes 4 and 5. Node 4 is at the bottom, and node 5 is on the left.
- Simulation control:** Contains buttons for Run, Speed limit, Start, Pause, Step, and Reload. It shows a Time of 01:06.298 and Speed: ---.
- PowerTracker: 5 motes:** A table showing power consumption statistics for five motes.
- Message output:** A log window showing messages from different motes, including sensor data and received data.
- Timeline showing 5 motes:** A timeline view showing the activity of five motes over time.

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Z1 1	11.51%	5.21%	0.16%
Z1 2	10.64%	5.20%	0.17%
Z1 3	10.90%	5.17%	0.18%
Z1 4	11.08%	5.21%	0.09%
Z1 5	1.55%	0.00%	0.33%
AVERAGE	9.14%	4.16%	0.19%

```
Time      Mote      Message
01:06.197 ID:3      Sensor raw values: temperature: -1, humidity: -1, battery: 0
01:06.197 ID:3      Exact values-->
01:06.199 ID:3      Temperature: 615 degrees Celsius
01:06.204 ID:3      Rel. humidity: 2650%
01:06.208 ID:3      Battery: 239 (0.291 mV)
01:06.216 ID:3
01:06.216 ID:3
01:06.285 ID:5      Data recv:[103 2 90 10 ], from: 3.0
01:06.295 ID:4      Data recv:[103 2 90 10 ], from: 3.0
01:06.295 ID:4
```

More examples

unicast-example.c

- ipv6
- simple-udp-rpl

The screenshot displays a network simulation interface with several panels:

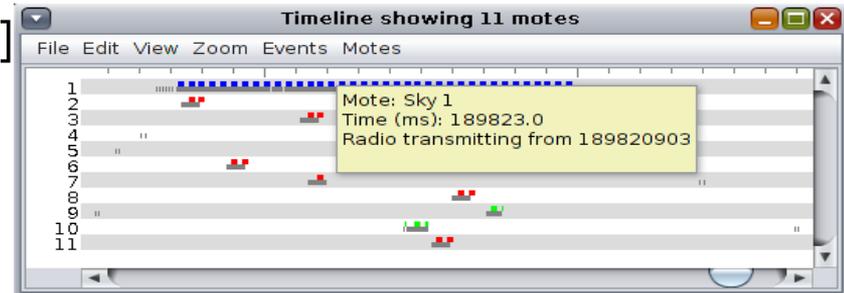
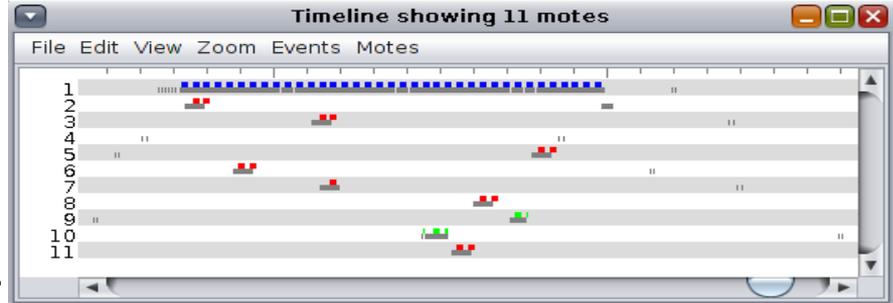
- Network:** A grid-based view showing a network topology with nodes (numbered 1-11) and their IPv6 addresses. A green shaded area represents a specific network region.
- Simulation control:** A panel with buttons for Start, Pause, Step, and Reload. It shows the simulation time as 03:20.264 and the speed as 196.96%.
- PowerTracker: 11 motes:** A table showing the power consumption of 11 motes (Sky 1 to Sky 11) and an average.
- Mote output:** A log window showing the time, mote ID, and message for each event.
- Timeline showing 11 motes:** A timeline view showing the activity of each mote over time.

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 1	1.47%	0.33%	0.10%
Sky 2	1.43%	0.41%	0.04%
Sky 3	1.55%	0.51%	0.05%
Sky 4	1.46%	0.43%	0.05%
Sky 5	1.61%	0.43%	0.10%
Sky 6	1.80%	0.51%	0.17%
Sky 7	1.41%	0.41%	0.03%
Sky 8	1.45%	0.44%	0.05%
Sky 9	2.06%	0.63%	0.17%
Sky 10	1.50%	0.39%	0.11%
Sky 11	1.27%	0.28%	0.05%
AVERAGE	1.54%	0.43%	0.08%

Time	Mote	Message
02:22.896	ID:4	Sending unicast to aaaa::212:7401:1:101
02:23.187	ID:1	Data received from aaaa::212:7404:4:404 on port 1234 ...
02:30.183	ID:11	Sending unicast to aaaa::212:7401:1:101
02:30.312	ID:1	Data received from aaaa::212:740b:b:b0b on port 1234 ...
02:37.482	ID:9	Sending unicast to aaaa::212:7401:1:101
02:37.564	ID:1	Data received from aaaa::212:7409:9:909 on port 1234 ...
02:41.994	ID:2	Sending unicast to aaaa::212:7401:1:101
02:46.684	ID:7	Sending unicast to aaaa::212:7401:1:101
03:03.381	ID:5	Sending unicast to aaaa::212:7401:1:101
03:03.563	ID:1	Data received from aaaa::212:7405:5:505 on port 1234 ...
03:08.878	ID:11	Sending unicast to aaaa::212:7401:1:101
03:09.062	ID:1	Data received from aaaa::212:740b:b:b0b on port 1234 ...
03:19.220	ID:6	Sending unicast to aaaa::212:7401:1:101

Timeline in COOJA

- ❑ Radio ON/OFF
 - No colour: radio off
 - Grey: radio on
- ❑ Radio RX/TX
 - Green: received a packet
 - Blue: packet sent
 - Red: interfered radio (collisions etc.)
- ❑ Right-clicking will reveal additional info.[2]

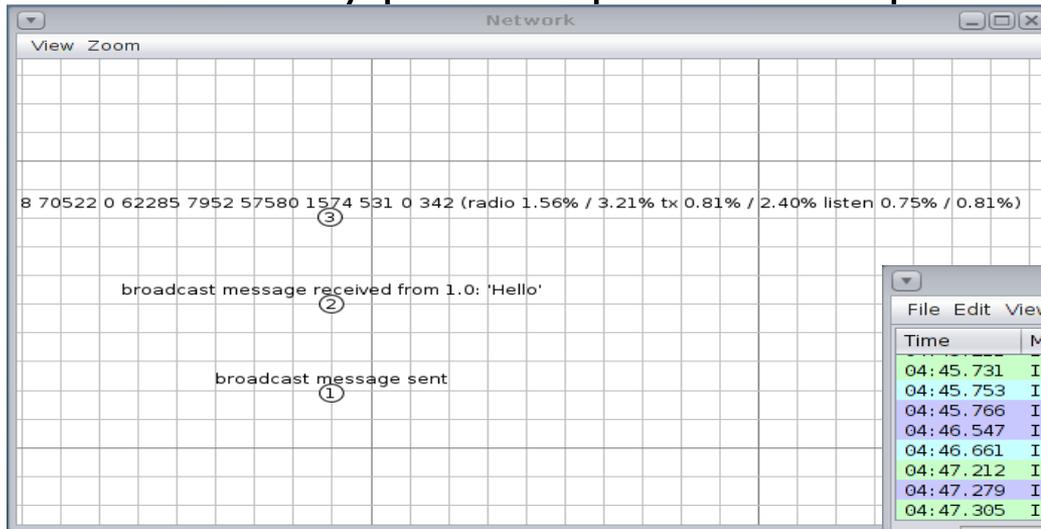


Measure Power Consumption with Energest

- ❑ Can be used for obtaining per-component power consumption on Contiki.
 - (cpu_ON, LPM, TX, RX)
 - i.e. the time the radio was in RX mode (rxon)
- ❑ For RX:
 - $\text{Power(mW)} = (\text{rxend} - \text{rxstart}) * 20\text{mA} * 3\text{V} / 4096 / \text{runtime(seconds)}$
 - If you do not divide by runtime you get the energy consumption during runtime.

Measure Power Consumption: Powertrace

- ❑ Uses Energest along with a periodic difference of the rtimer ticks to get average power over a shorter period of time or for particular network modes[3].
- ❑ Periodically prints out power consumption



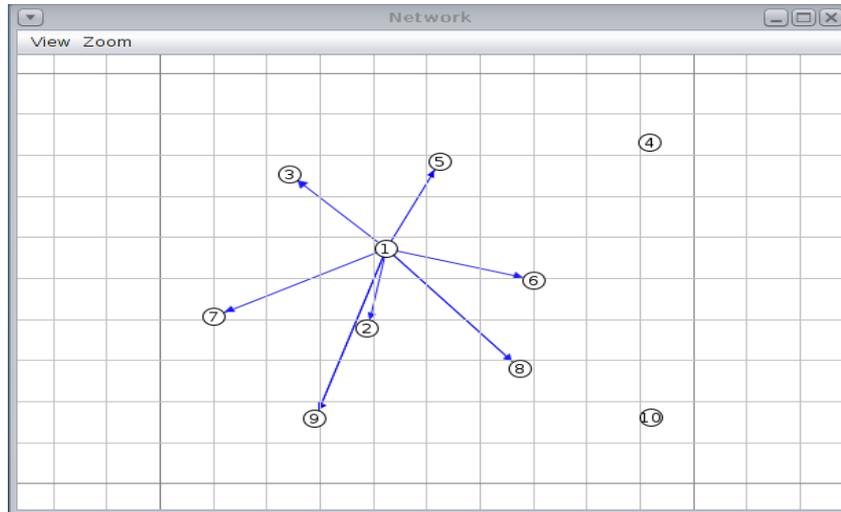
The Mote output window shows a log of network events with columns for Time, Mote, and Message.

Time	Mote	Message
04:45.731	ID:2	broadcast message sent
04:45.753	ID:3	broadcast message received from 2.0: 'Hello'
04:45.766	ID:1	broadcast message received from 2.0: 'Hello'
04:46.547	ID:1	36612 P 1.0 142 682820 8688722 75944 64224 0 56899 3395 62...
04:46.661	ID:3	36612 P 3.0 142 692249 8679337 75918 70522 0 62285 7952 57...
04:47.212	ID:2	36612 P 2.0 142 691741 8679795 72755 67775 0 53302 7806 57...
04:47.279	ID:1	broadcast message sent
04:47.305	ID:2	broadcast message received from 1.0: 'Hello'

Filter:

Measure Power Consumption: PowerTracker

- ❑ A COOJA plugin that measures the average simulated radio duty cycles.
- ✓ `simple-udp-rpl/broadcast-example.c`



PowerTracker: 10 motes

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 1	1.75%	0.55%	0.14%
Sky 2	1.75%	0.55%	0.15%
Sky 3	1.75%	0.55%	0.08%
Sky 4	1.73%	0.55%	0.04%
Sky 5	1.76%	0.55%	0.10%
Sky 6	1.75%	0.55%	0.12%
Sky 7	1.73%	0.55%	0.08%
Sky 8	1.75%	0.55%	0.10%
Sky 9	1.73%	0.55%	0.08%
Sky 10	1.73%	0.55%	0.04%
AVERAGE	1.74%	0.55%	0.09%

Print to console/Copy to clipboard Reset

Network Protocols

- ❑ COOJA has 2 stacks: uIP and Rime
- ❑ Protocol stacks may be interconnected
 - uIP data can be transmitted over Rime and vice versa
- ❑ Cooja can be used to emulate network protocols:
 - RPL
 - LIBP

Introduction to LIBP

- ❑ LIBP, known as the **Least Interference Beaconsing Protocol**, is the implementation of the **Least Interference Beaconsing Algorithm, LIBA**.
- ❑ LIBP extends the beaconsing process widely used by collection protocols with load balancing to improve the Ubiquitous Sensor Network (USN) energy efficiency[4].
- ❑ The process involving the least interference paradigm allows the selection of **a parent node** that has the **smallest number of children**. This is **a point of least traffic flow interference**.
- ❑ The parent selection model chooses the first parent node heard from, whereby the sensor nodes hear from a set of neighbours and select the least burdened (in number of children) as the parent node.

LIBP – Rime startup

The screenshot displays a network simulation interface with three main panels: Network, Simulation control, and Mote output.

Network Panel: Shows a network diagram with 10 nodes (ID 1-10) on a grid. Node 5 is the central hub, connected to nodes 2, 3, 8, and 9. Node 9 is further connected to node 7. Nodes 1, 4, 6, and 10 are isolated.

Simulation control Panel: Contains buttons for 'Start', 'Pause', 'Step', and 'Reload'. It displays 'Time: 00:23.247' and 'Speed: 421.03%'.

Mote output Panel: A log window showing the startup sequence for Rime on various nodes. The log entries are as follows:

Time	Mote	Message
00:00.343	ID:8	Rime started with address 8.0
00:00.351	ID:8	MAC 08:00:00:00:00:00:00:00 Contiki 2.7 started. Node i...
00:00.360	ID:8	CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26
00:00.362	ID:8	Starting 'Test LIBP process'
00:00.506	ID:2	Rime started with address 2.0
00:00.514	ID:2	MAC 02:00:00:00:00:00:00:00 Contiki 2.7 started. Node i...
00:00.521	ID:6	Rime started with address 6.0
00:00.523	ID:2	CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26
00:00.525	ID:2	Starting 'Test LIBP process'
00:00.529	ID:6	MAC 06:00:00:00:00:00:00:00 Contiki 2.7 started. Node i...
00:00.538	ID:6	CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26
00:00.540	ID:6	Starting 'Test LIBP process'
00:00.619	ID:4	Rime started with address 4.0
00:00.626	ID:4	MAC 04:00:00:00:00:00:00:00 Contiki 2.7 started. Node i...
00:00.635	ID:4	CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26
00:00.638	ID:4	Starting 'Test LIBP process'
00:00.653	ID:1	Rime started with address 1.0
00:00.660	ID:1	MAC 01:00:00:00:00:00:00:00 Contiki 2.7 started. Node i...

Upon network startup, Rime started with address 8.0 for Node ID 8.

The image details the radio channel as 26 and the channel check rate of 8 Hz.

LIBP – sink mote

The screenshot displays a network simulation interface with three main windows:

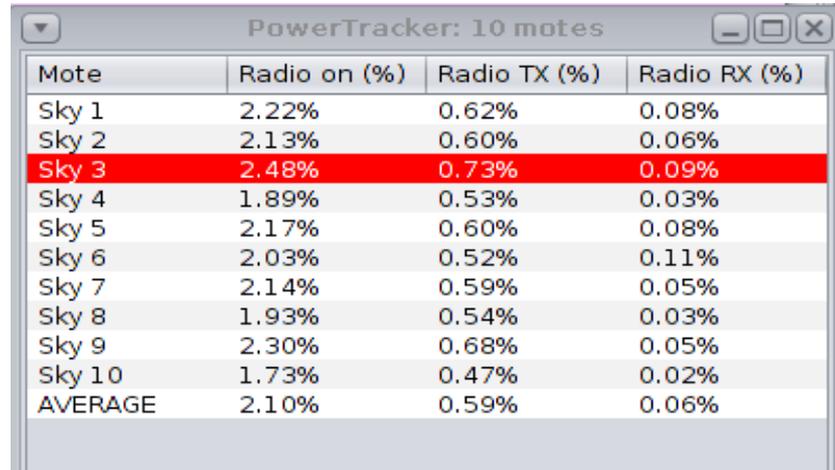
- Network:** A grid-based network topology with 10 nodes (ID 1-10). Nodes 1, 2, 3, 4, 5, 6, 7, 8, and 9 are connected. Nodes 1 and 2 are highlighted in green, and nodes 3 and 4 are highlighted in blue. The connection between nodes 1 and 2 is labeled "100.0%".
- Simulation control:** A window with buttons for "Start", "Pause", "Step", and "Reload". It shows "Time: 02:15.386" and "Speed: 416.67%".
- Mote output:** A log window showing messages from various motes. The messages are color-coded: green for "Sending", blue for "Sink got message from...", and purple for "Hi from sink thread".

The Mote output log contains the following messages:

Time	Mote	Message
00:01.189	ID:3	CSMA-CA/RTT/MAC, channel check rate 8 Hz, radio channel 20
00:01.189	ID:3	Starting 'Test LIBP process'
02:00.401	ID:8	Sending
02:00.405	ID:8	#L 9 1
02:00.673	ID:1	Hi from sink thread
02:00.673	ID:1	Sink got message from 8.0, seqno 0, hops 4: len 8 'Hell...
02:00.923	ID:1	Sending
02:00.923	ID:1	Sink got message from 8.0, seqno 0, hops 4: len 8 'Hell...
02:04.630	ID:4	Sending
02:04.633	ID:4	#L 2 1
02:04.922	ID:1	Sink got message from 4.0, seqno 0, hops 2: len 8 'Hell...
02:09.665	ID:6	Sending
02:09.668	ID:6	#L 3 1
02:10.047	ID:1	Sink got message from 6.0, seqno 0, hops 2: len 8 'Hell...
02:12.424	ID:2	Sending
02:12.427	ID:2	#L 1 1
02:12.548	ID:1	Sink got message from 2.0, seqno 0, hops 1: len 8 'Hell...

Sink mote with ID 1 – After 2 minutes, 0 seconds and 673 milliseconds, ID 1 broad-casted to the network that it is sink by sending **“Hi from sink thread”**.

LIBP - PowerTracker



Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 1	2.22%	0.62%	0.08%
Sky 2	2.13%	0.60%	0.06%
Sky 3	2.48%	0.73%	0.09%
Sky 4	1.89%	0.53%	0.03%
Sky 5	2.17%	0.60%	0.08%
Sky 6	2.03%	0.52%	0.11%
Sky 7	2.14%	0.59%	0.05%
Sky 8	1.93%	0.54%	0.03%
Sky 9	2.30%	0.68%	0.05%
Sky 10	1.73%	0.47%	0.02%
AVERAGE	2.10%	0.59%	0.06%

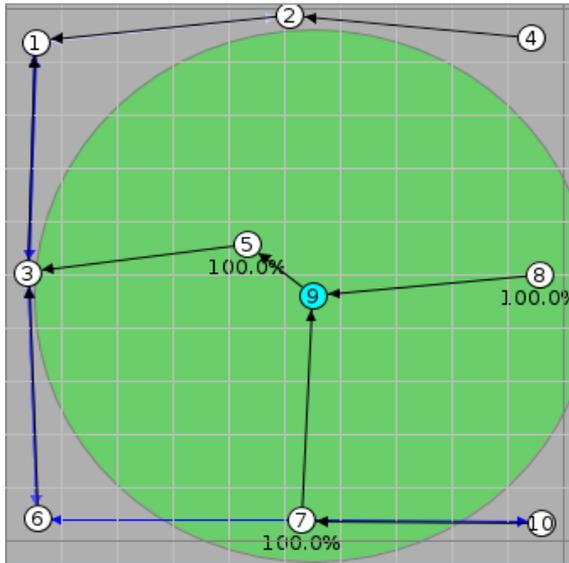
PowerTracker after **5 minutes**.

Sky 3 used the most power by being on most of the time.

It's Radio TX is also the highest with a value of **0.73%**.

It has the second highest Radio RX of **0.09%**.

LIBP – Parent with children



Node 3 has 2 children, namely nodes 5 and 6.

These nodes also have children.

Sky 10 used the **least power** with its

Radio on at 1.73%

Radio TX at 0.47% and

Radio RX of 0.02% (least percentage).

Sky 10 is ranked along the bottom of the tree,

has no children and is only active when it has

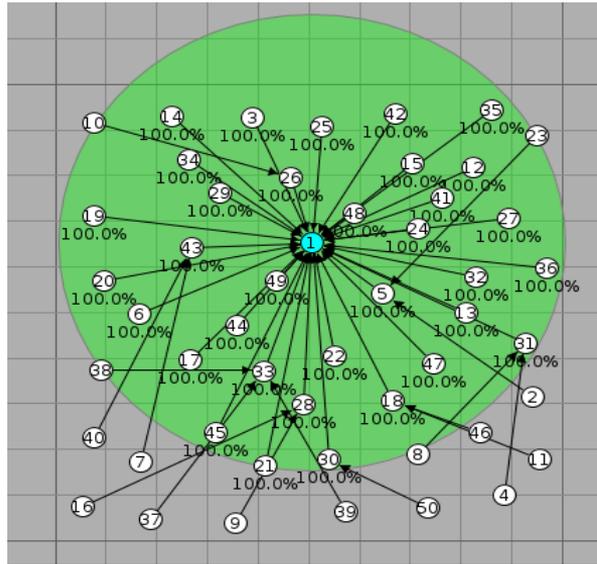
to send its data, unlike the other motes.

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 1	2.22%	0.62%	0.08%
Sky 2	2.13%	0.60%	0.06%
Sky 3	2.48%	0.73%	0.09%
Sky 4	1.89%	0.53%	0.03%
Sky 5	2.17%	0.60%	0.08%
Sky 6	2.03%	0.52%	0.11%
Sky 7	2.14%	0.59%	0.05%
Sky 8	1.93%	0.54%	0.03%
Sky 9	2.30%	0.68%	0.05%
Sky 10	1.73%	0.47%	0.02%
AVERAGE	2.10%	0.59%	0.06%

LIBP – large network

- ❑ Starting COOJA with “`ant run`” will give you the default Java maximum memory
 - 5 – 10 emulated nodes
- ❑ If you use “`ant run_bigmem`” you will be able to simulate/emulate larger networks.

LIBP – large network



PowerTracker: 50 motes

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Contiki 1	100.00%	0.21%	0.37%
Contiki 2	100.00%	0.01%	0.22%
Contiki 3	100.00%	0.01%	0.38%
Contiki 4	100.00%	0.01%	0.17%
Contiki 5	100.00%	0.03%	0.57%
Contiki 6	100.00%	0.01%	0.47%
Contiki 7	100.00%	0.01%	0.19%
Contiki 8	100.00%	0.01%	0.28%
Contiki 9	100.00%	0.01%	0.20%
Contiki 10	100.00%	0.01%	0.28%
Contiki 11	100.00%	0.01%	0.20%
Contiki 12	100.00%	0.01%	0.38%
Contiki 13	100.00%	0.01%	0.53%
Contiki 14	100.00%	0.01%	0.33%
Contiki 15	100.00%	0.01%	0.43%
Contiki 16	100.00%	0.01%	0.14%
Contiki 17	100.00%	0.01%	0.52%
Contiki 18	100.00%	0.04%	0.48%
Contiki 19	100.00%	0.01%	0.36%
Contiki 20	100.00%	0.01%	0.40%

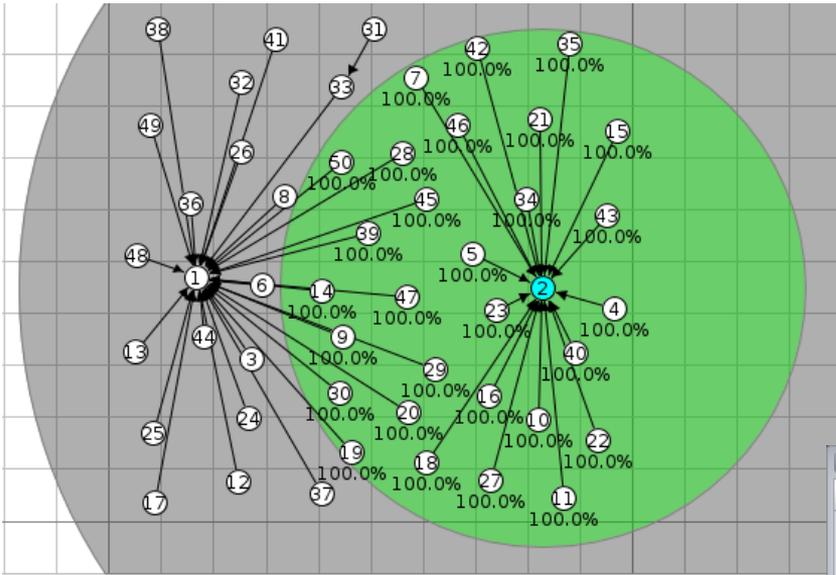
Print to console/Copy to clipboard Reset

Compiled **50 Cooja Motes**

(Simulated motes – run as native java code)

Downside: Cannot do any power profiling

LIBP – 2 sinks



Node ID 1 : Primary Sink

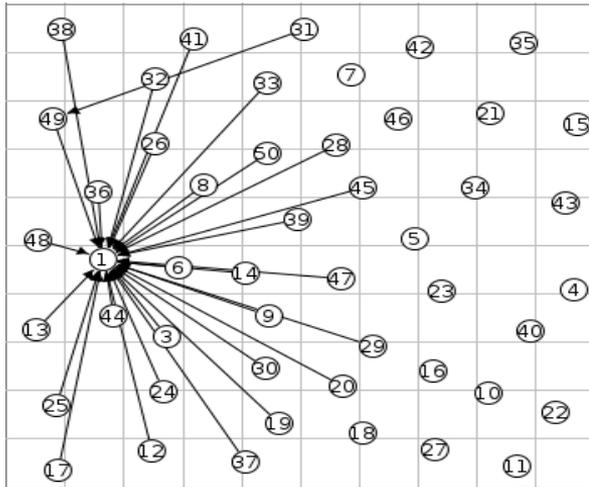
Node ID 2 : Secondary Sink

Load balanced network with 2
sink nodes

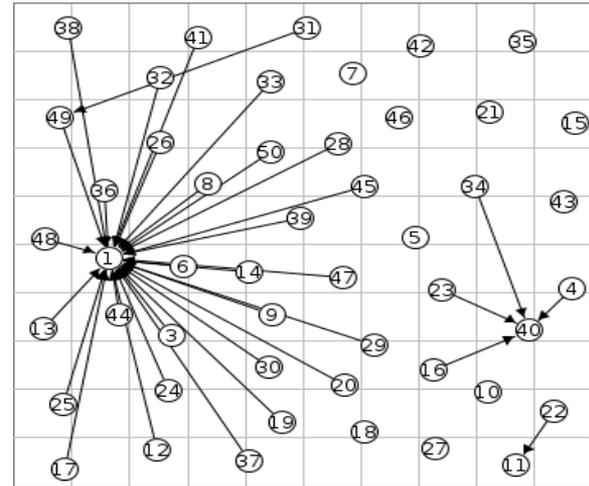
```
Mote output
File Edit View
Time ms  Mote  Message
12840669 ID:2   SECONDARY SINK NODE[0]: 2
12840669 ID:2   CHILD ELEMENT[1]: 10
12840689 ID:43  Sending
12840690 ID:26  Sending
12840691 ID:2   parent :2
12840691 ID:2   Sink got message from 43.0, seqno 158, hops 1: len 1 '2'
12840691 ID:2   Child: 43->Parent 2
12840691 ID:2   SECONDARY SINK NODE[0]: 2
Filter:
```

LIBP – 2 sinks

- The 2 sink nodes help with load balancing and network recovery
 - e.g. when one node goes offline, it's children (orphaned nodes) attempt to connect to the other sink

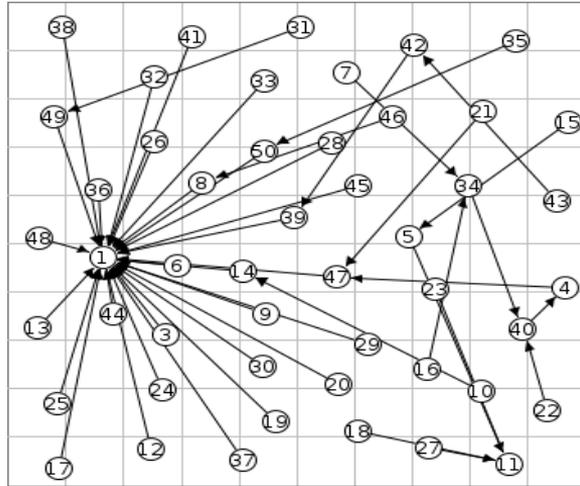


Secondary sink - offline

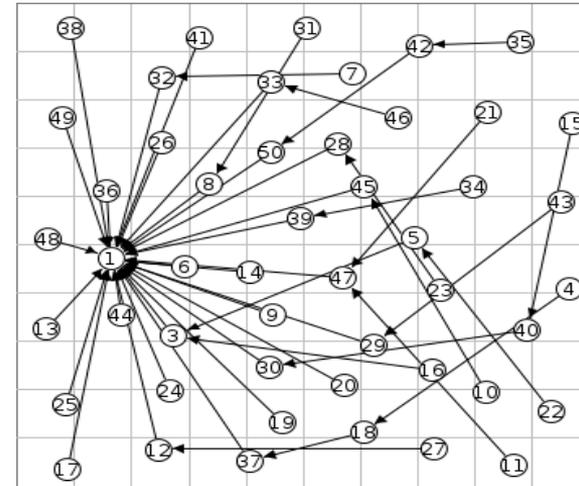


Recovery in process

LIBP – 2 sinks



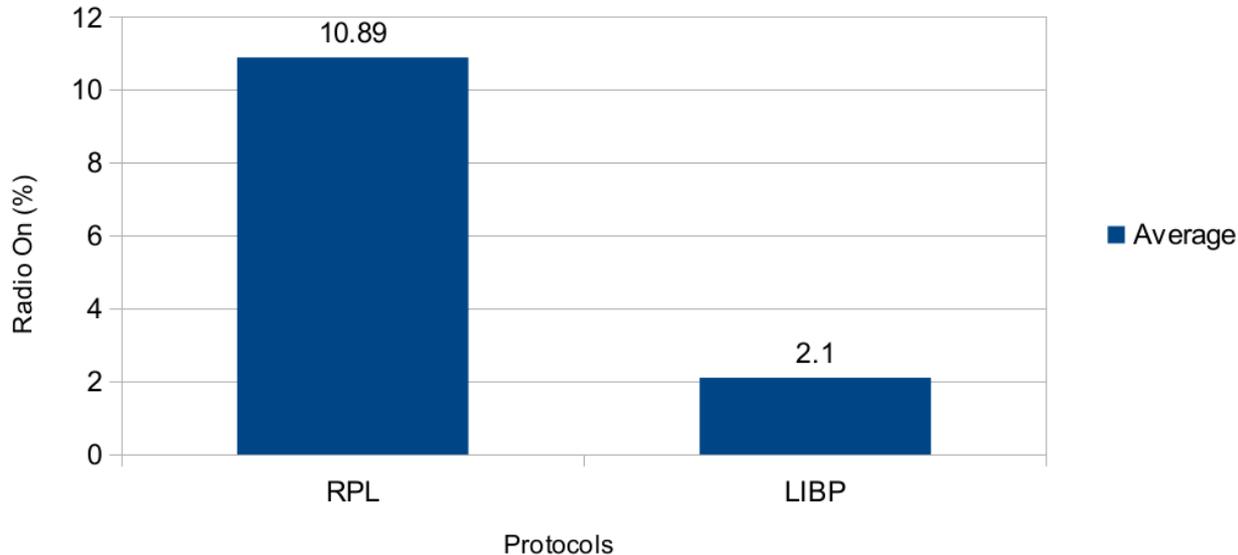
Orphaned nodes also attempt to connect with each other



Network has recovered and every node is making use of Node 1 as its sink node

Recovery in process

Comparison of RPL and LIBP Radio On Averages



LIBP uses less power amongst **10 Skymotes** in relation to keeping their radios on, thus creating a more energy efficient network.

References

- [1] Alan, A., & Pritsker, B. (n.d.). Why Simulation Works. *Proceedings of the 1989 Winter Simulation Conference*. Retrieved from <http://www.sfu.ca/~vdabbagh/p1-pritsker.pdf>
- [2] Voigt, T. *Contiki COOJA Crash Course*. Swedish Institute of Computer Science. Retrieved from <https://www.sics.se/~thiemo/seniot09cccc-slides.pdf>
- [3] Kopf, D. *What is Difference Between The Energest and PowerTrace*. Retrieved from <http://contiki-developers.narkive.com/VMpBTquh/what-is-difference-between-the-energest-and-powertrace>
- [4] Bagula, A., Djenouri, D., Karbab, E. *Ubiquitous Sensor Network Management: The Least Interference Beaconing Model*. In *Proceedings of PIMRC 2013*, Pages 2352-2356, 2013.
- [5] Lutando Ngqakaza & Antoine Bagula, “**Least Path Interference Beaconing Protocol (LIBP): A Frugal Routing Protocol for The Internet-of-Things**”, in *proceedings of the IFIP Wired/Wireless Internet Communications WWIC 2014*, Lecture Notes in Computer Science Volume 8458, 2014, pp 148-161, 2014.