IDIT-SDN (v0.1): Installation Guide (for Linux 64 bits - Ubuntu 20.04.6 LTS - April, 2023)

Gustavo N. Segura[1], **Arsenia Chorti** [2], **Cintia B. Margi**[3]

[1]Escuela de Ingeniería Eléctrica – Universidad de Costa Rica
Montes de Oca – San José – Costa Rica

[2]ETIS UMR8051, CY Université, ENSEA, CNRS, F-95000, Cergy, France

[3]Escola Politécnica – Universidade de São Paulo, São Paulo, SP, Brazil

gustavoalonso.nunez@ucr.ac.cr, arsenia.chorti@ensea.fr, cintia@usp.br

Contents

1. Introduction

IDIT-SDN is an intrusion detection framework for Software Defined Wireless Sensor Network (SDWSN). This tool is completely open and freely available, designed to be independent of the operating system and its functions. Although, it is currently implemented on IT-SDN v 0.41 [Alves et al. 2019, Alves et al. 2017]

IT-SDN comprises SDN-enabled WSN nodes developed under Contiki OS [Dunkels et al. 2004]. We provide sample code for data generating nodes (named enabled-nodes throughout the document) and sink-nodes. The controller is developed in C and C++ with Qt (we call this software controller-pc). The controller-pc software must connect to a WSN node for communicating with the other nodes in the network. The node programmed with the interfacing software is called controller-node.

In order to compile and run IDIT-SDN nodes and controller, you need to follow these steps (described in the next sections):

2. Obtain Contiki OS;
3. Obtain IDIT-SDN;
4. Obtain the GCC toolchain for MSP430;
5. Obtain Qt framework;
6. IDIT-SDNcompilation;
7. Simulating IDIT-SDN;

2. Requirements

2.1. Obtain Contiki OS

Download Contiki release 3.0 available at `https://github.com/contiki-os/contiki/releases/tag/3.0`

After downloading, place the contiki directory in the home folder `/home/USERNAME`.

Alternatively, you can get Contiki release 3.0 in the command line, as follows:

```
$ cd /home/USERNAME
$ wget https://github.com/contiki-os/contiki/archive/3.0.zip
$ unzip 3.0.zip
```

You need to get MSPSim to emulate instruction level of MSP430 series micropocessor on Cooja (the Contiki Network Simulator). Download it from `https://github.com/contiki-os/mspsim` and unpack it in directory `/home/USERNAME/contiki-3.0/tools/mspsim/`.

To run Cooja, you will need Java JDK 8 as well as the ant build tool. If you need to install them, use the following commands in a terminal.

```
~$ sudo apt-get install ant
~$ sudo apt-get update
~$ sudo apt-get install oracle-java8-set-default
```

Or you can try:

```
~$ sudo apt-get install ant
~$ sudo apt-get update
~$ sudo apt-get install openjdk-8-jdk
~$ sudo update-alternatives --config java
```

To test if download and extraction succeeded, run the command `ant run` on folder `/home/USERNAME/contiki-3.0/tools/cooja`:

```
~/contiki-3.0/tools/cooja$ ant run
```

If everything was installed correctly, it will start with a blue empty window (Cooja: The Contiki Network Simulator) as shown in Figure 1.

You can get more information about Cooja at `https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja`.

2.2. Obtain IDIT-SDN

Download the IDIT-SDN framework from `https://github.com/gnunezucr/idit-sdn`

When downloaded, extract and place the directory on `/home/USERNAME`.

2.3. Obtain the GCC toolchain for MSP430

We are using MSP430-based platforms (such as Tmote) and thus the following packages are needed: `binutils-msp430`, `gcc-msp430`, `msp430-libc`, `msp430mcu` and `mspdebug` to compile and run enabled-nodes and controller-node developed to

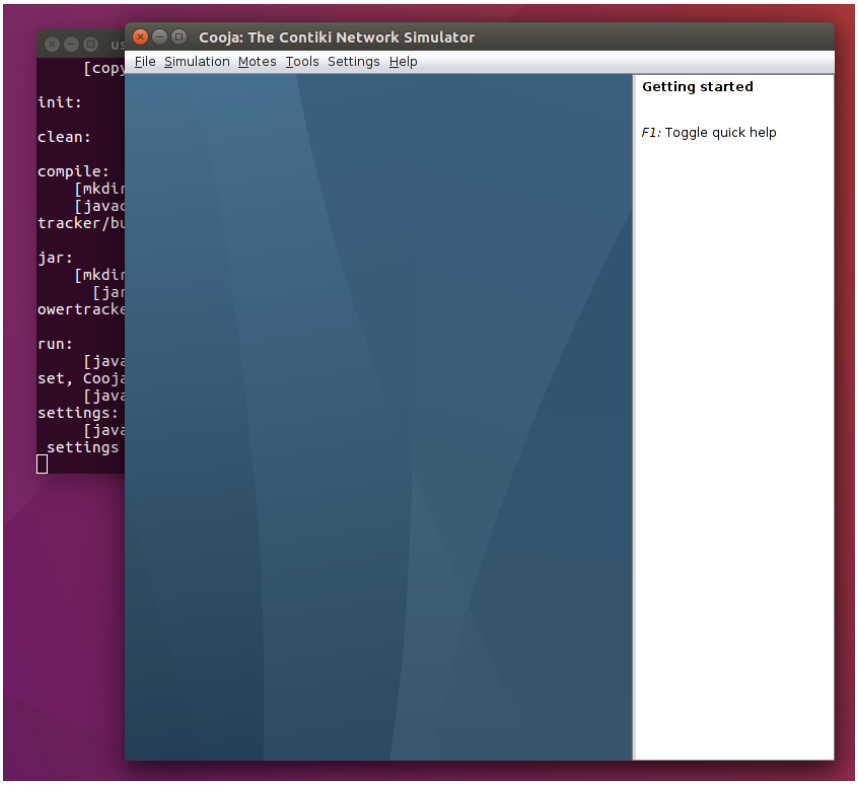

Figure 1. Cooja execution interface.

ContikiOS.

You can run the following command to install all packages:

```
$ sudo apt-get install build-essential binutils-msp430 gcc-msp430 msp430-libc
msp430mcu mspdebug
```

You can get more information about MSP430-gcc at `https://github.com/jlhonora/mspgcc-install`.

If all previous steps succeed, you should be able to compile IT-SDN as instructed in Section 3.1.

2.4. Obtain Qt framework

To compile and run the `controller-pc` software you need to download and install QT framework with Qt 5.9 available at `https://download.qt.io/archive/qt/5.9/5.9.0/`. Figure 2 shows Qt Creator interface.

You also need to install the package:

```
$ sudo apt-get install libgl1-mesa-dev
```

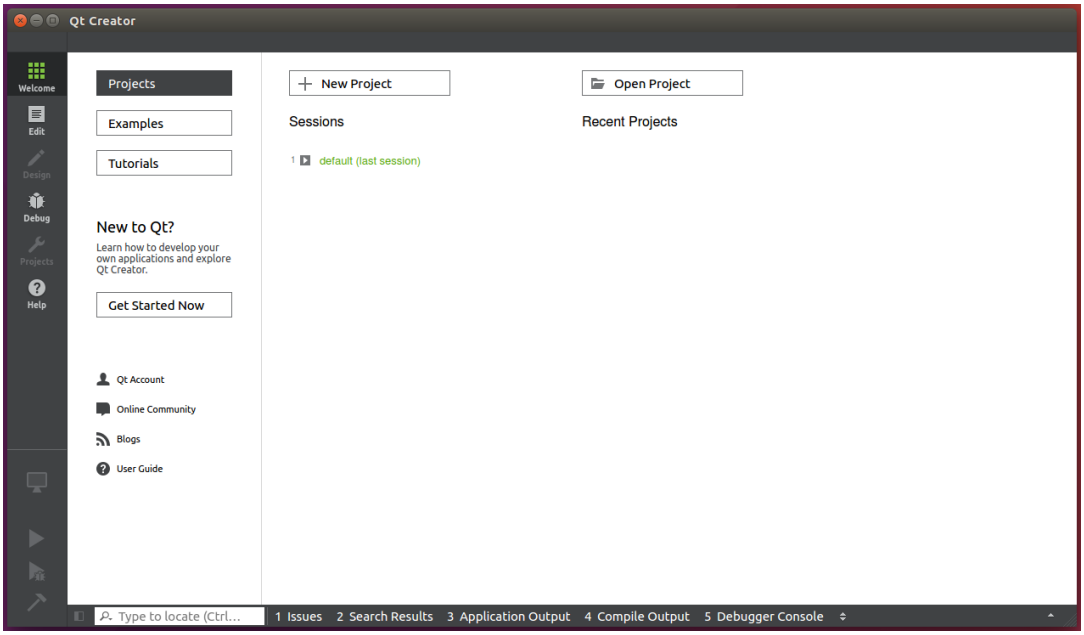

Figure 2. Qt Creator opened.

3. IDIT-SDNcompilation

In the next sections, we describe how to compile and execute IDIT-SDN.

3.1. Compile enabled-nodes and controller-node

You should change the first line `Contiki=...` on the `Makefile_enabled_node` and `Makefile_controller_node`, in folder `idit-sdn/applications`, to indicate the path to reach Contiki source files, such as shown below:

```
CONTIKI=/home/USERNAME/contiki-3.0
```

The compilation script is located in the applications directory. Please, verify the script has the appropriate permission to execute.

```
$ cd ~/idit-sdn/applications/
$ ./compile.sh
```

By default, the script compiles the code for the sky mote (TelosB mote). However, the target platform can be changed by typing its name as script parameter of the `compile.sh` file. Example:

```
$ ./compile.sh PLATFORM
```

If compilation succeeded, you will see the size of the three firmwares (enabled-node, sink-node and controller-node), as shown in Figure 3.

3.2. Compile controller-pc

Before compiling the controller-pc, you need to indicate Contiki path. To do so, create a file named `controller-pc.pro.contiki` and save on folder `/home/USERNAME/idit-sdn/controller-server/controller-pc` with the following content:

```
rm controller-node.co obj_sky/contiki-sky-main.o
   text    data     bss     dec     hex filename
  41614     188    9400   51202    c802 attack-fdff.sky
  32854     170    9004   42028    a42c controller-node.sky
  48162     200    9374   57736    e188 enabled-node.sky
  40766     188    9322   50276    c464 management-sink.sky
  40966     188    9346   50500    c544 sink-node.sky
```

Figure 3. Compiling enabled-node, sink-node and controller-node.

```
CONTIKI=/home/USERNAME/contiki-3.0
```

To compile and run controller-pc, open the Qt Creator and click on button [Open Project]. Navigate to the path `/home/USERNAME/idit-sdn/controller-server/controller-pc` and select the project `controller-pc.pro`.

You will see the Configure Project window, such as the one depicted in Figure 4). Then, click on [Configure Project].

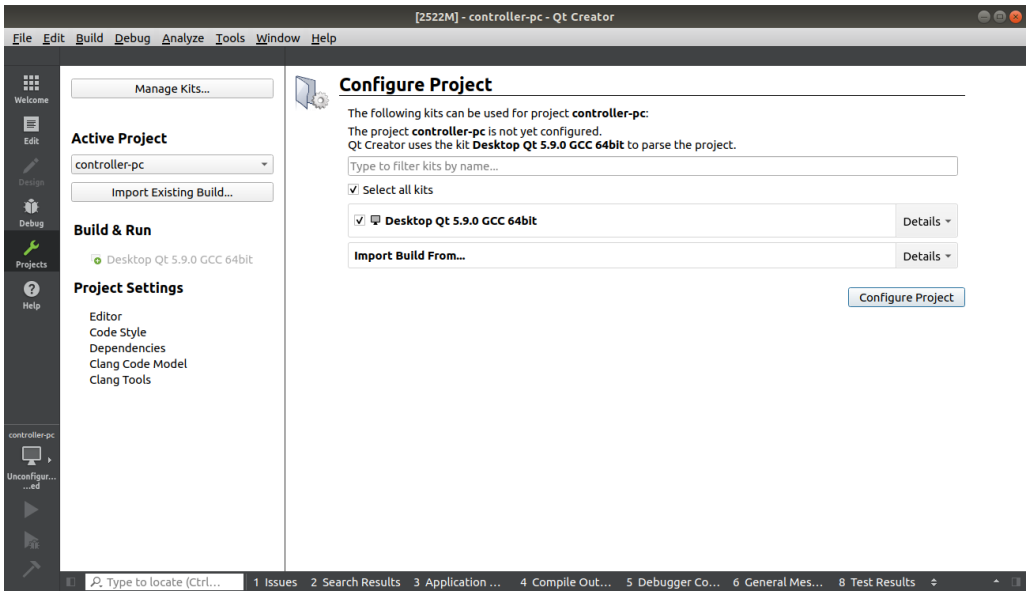

Figure 4. Configuring project kit to controller-pc.

Now you are ready to compile the controller-pc. Click on menu Build→Build All.

4. Simulating IDIT-SDN

4.1. Running controller-pc

After compiling, click on menu Build→Run on Qt Creator to run controller-pc. You will see the controller-pc window as shown in Figure 5.

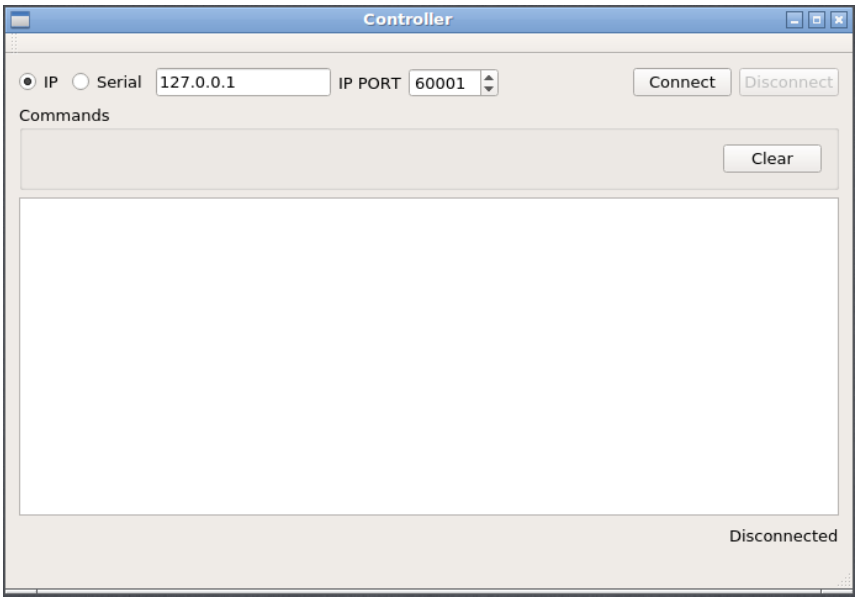

Figure 5. First window for controller-pc.

4.2. Starting Cooja

Open a terminal window, go to the Cooja directory and then open Cooja as follows.

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

After Cooja opened, go to the File menu and click Open simulation → Browse... Navigate to `/home/USERNAME/idit-sdn/simulation/` and open file `ITSDN_n36_s1_a1_3_GRID.csc`, as shown in Figure 6 and Figure 6.

4.3. Connecting controller-pc to controller-node

Once Cooja simulation is up, you need to connect the controller-pc to controller-node. On controller-pc window, verify the IP PORT is set on 60001 (Figure 5), then, click [Connect] button. Once they are connected, the controller interface will change as depicted in Figure 8.

4.4. Running Cooja Simulation using GUI

If the controller-pc is successfully connected to Cooja simulation, now you can click on [Start] on Cooja to start, such as shown in Figure 9. The Mote output window contains serial output from all nodes. IT-SDN nodes print messages to ease the computation of network metrics and statistics, as well as messages regarding node status and message transmission. These features can be activated by defining, respectively, the macros SDN_METRIC or DEBUG_SDN in the application makefile.

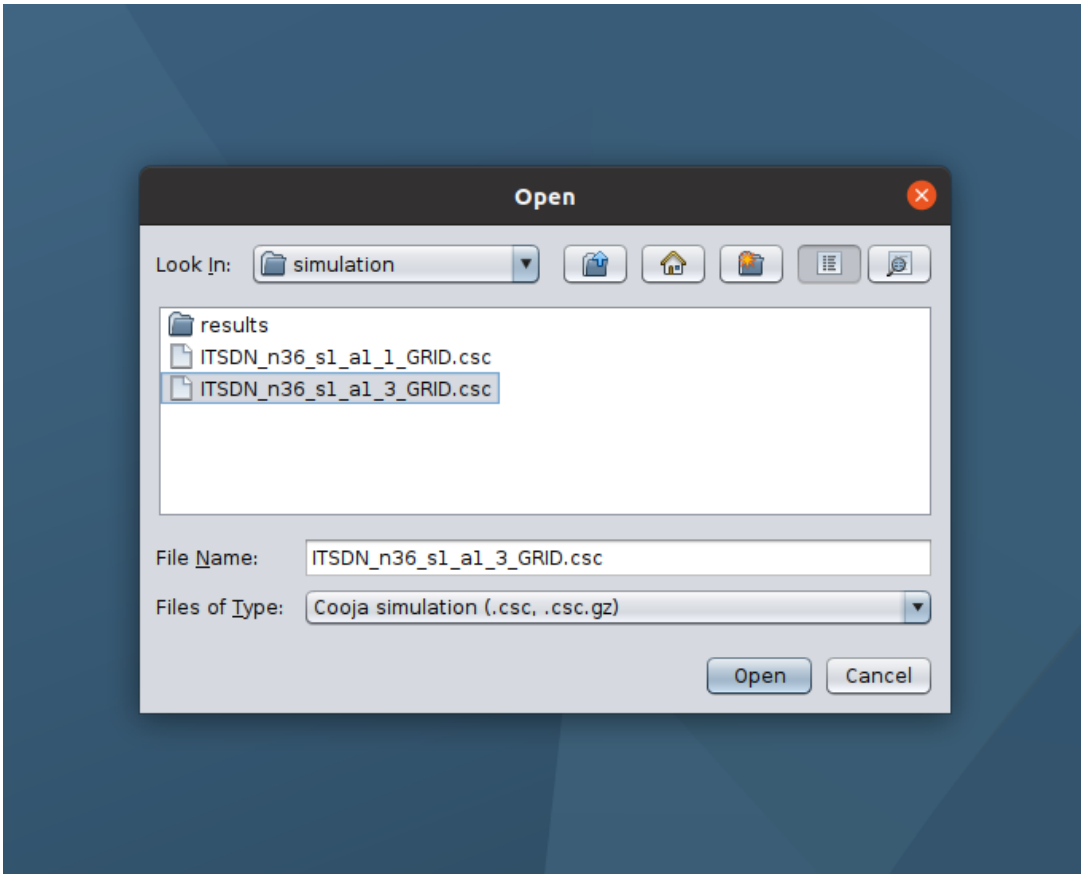

Figure 6. Opening Cooja simulation.

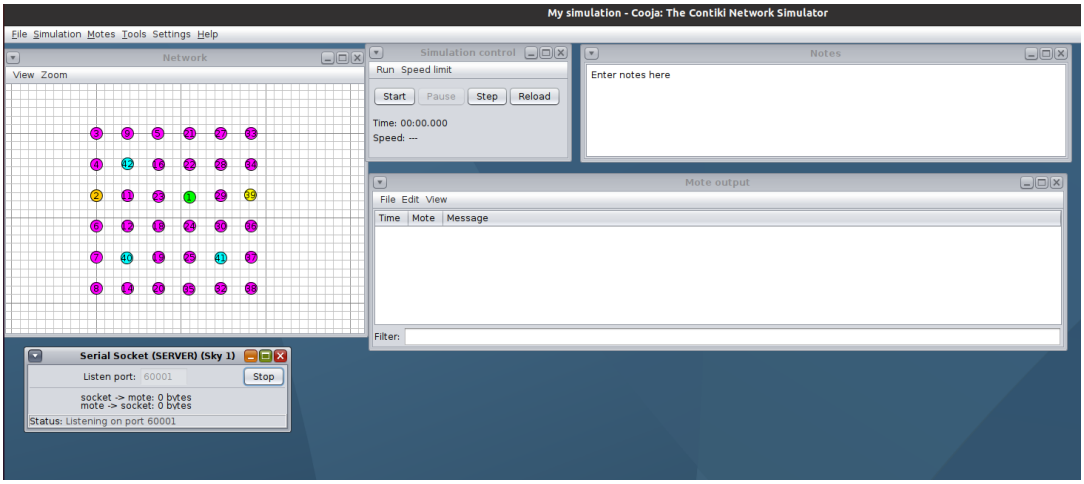

Figure 7. Cooja simulation with 36 nodes.

4.5. Simulation using script (no GUI)

You can also use an script to run Cooja simulator backwards. This is useful for long simulations and to run multiple replications in parallel. We provide an script that works with the examples we include in the code folder, but it is also configurable. To have a better understanding about how to use simulation through scripts in Cooja, please check this site: `https://github.com/contiki-os/contiki/wiki/`

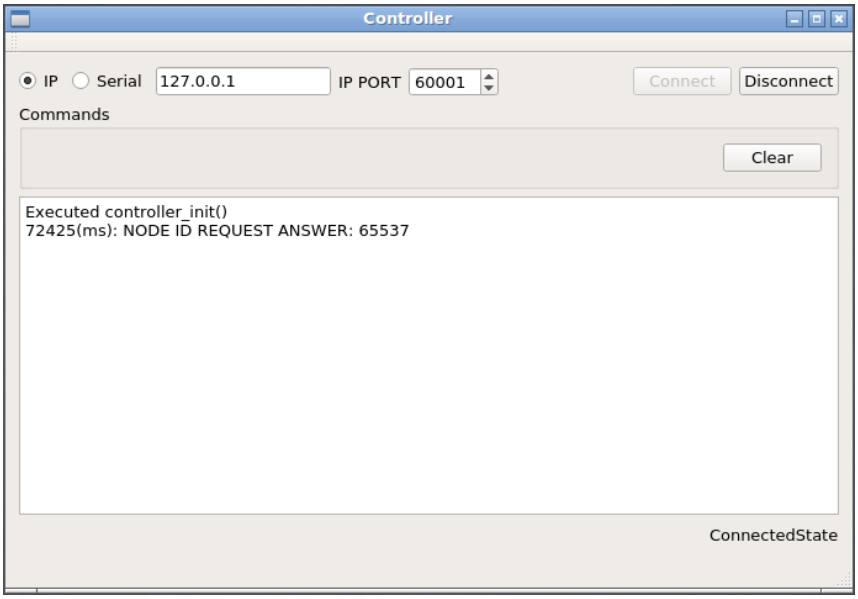

Figure 8. Connecting the controller-pc to controller-node.

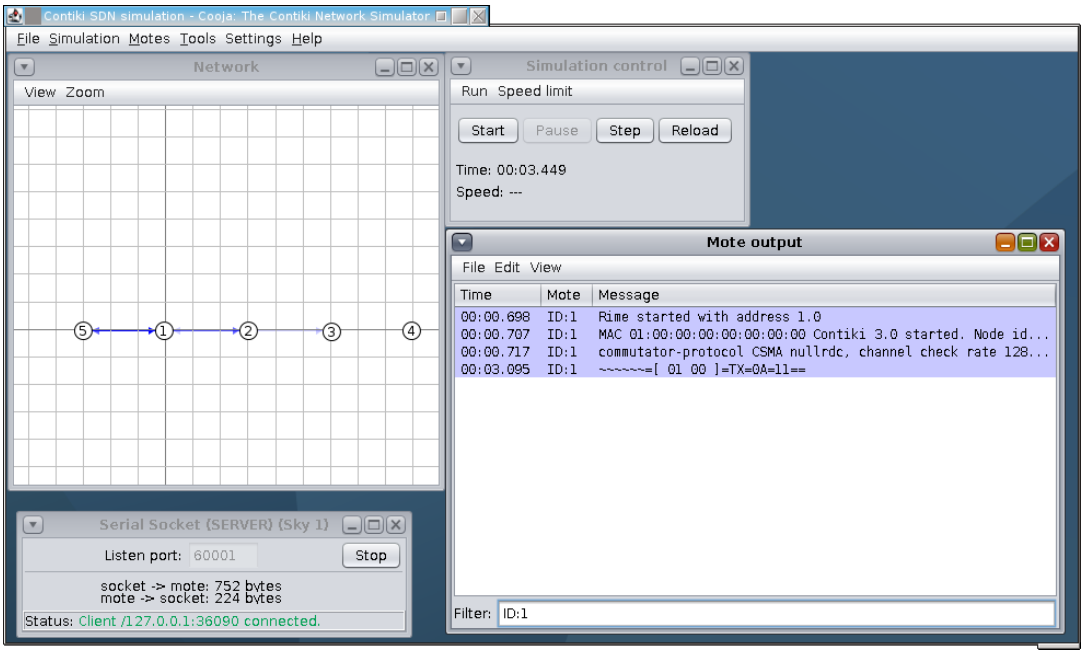

Figure 9. Starting Cooja simulation.

`Using-Cooja-Test-Scripts-to-Automate-Simulations`

The script we provide is named as `run_simulations.sh` and it is located in the folder `/home/USERNAME/idit-sdn/simulation`. Before using the script, you should check the paths in the file `/home/USERNAME/idit-sdn/path_simulation.sh` and set them according to your user paths.

Then, you should create a copy of Contiki's folder but changing its name to `contiki-3.01`. If you want to run more than one simulation in parallel, you should

create as many copies of Contiki's folder as simulations you want. For example, if you want to run three simulations in parallel, you should have something such as shown in Figure 10.

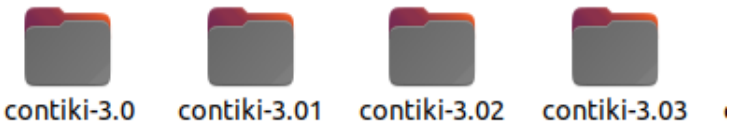

Figure 10. Contiki's folder replications

The parameters `MIN_ITER` and `MAX_ITER` defines the number of the first and the last simulation. If you want 10 replications, you should set them as `MIN_ITER=1` and `MAX_ITER=10`. Then, the parameter `COOJA_INSTANCES` defines the number of simulations in parallel.

The parameters `nodes_v` and `topologies` defines the number of nodes and the topology variation. For the examples we provide, the number of nodes is 36 and there are two topologies: `a1_1` and `a1_3`. However, you can create new simulations environments (.csc files) following the same nomenclature and use this script without changes. The parameter `SIM_TIME_MS` defines the simulation time in milliseconds. The parameters in the script should look like shown in Figure 11.

```
# Simulation set configuration
MIN_ITER=1
MAX_ITER=3
COOJA_INSTANCES=3 #max simulations running in parallel
COOJA_CURRENT_INSTANCE=1

# nodes_v=(36 100)
nodes_v=(36)
# topologies=(a1_1 a2_1 a3_1 a1_3 a2_3 a3_3)
topologies=(a1_3)
# SIM_TIME_MS=1800000 # 30 minutes
# SIM_TIME_MS=10800000  # 3 h
SIM_TIME_MS=18000000 # 5 h
# SIM_TIME_MS=36000000 # 10 h
```

Figure 11. Simulation script parameters

To run the simulations script you type the next commands:

```
$ cd ~/idit-sdn/simulation/
$ ./run_simulation.sh
```

The script will compile IT-SDN and controller server code, do the connection with Cooja and start the simulation. If everything went well, you should see somehting like shown in Figure 12.

The logs will be in in the folder `/home/USERNAME/idit-sdn/simulation/results`, such as shown in Figure 13.

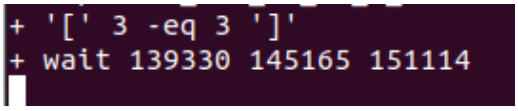

Figure 12. Simulation script parameters

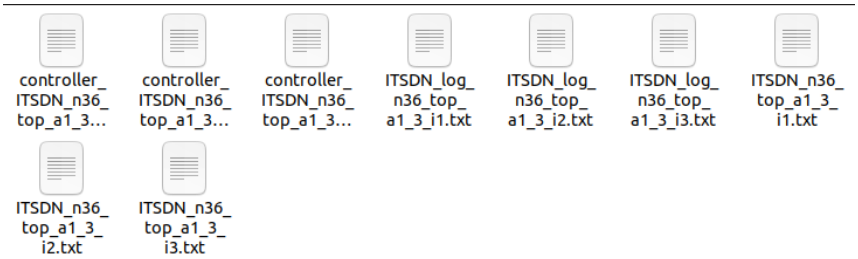

Figure 13. Simulation script parameters

References

Alves, R. C. A., Oliveira, D., Segura, G. N., and Margi, C. B. (2017). IT-SDN: Improved architecture for SDWSN. In *XXXV Simpósio Brasileiro de Redes de Computadores*. Available at `https://sites.google.com/usp.br/cintia/it-sdn`.

Alves, R. C. A., Oliveira, D. A. G., Nunez Segura, G. A., and Margi, C. B. (2019). The Cost of Software-Defining Things: A Scalability Study of Software-Defined Sensor Networks. *IEEE Access*, 7:115093–115108.

Dunkels, A., Gronvall, B., and Voigt, T. (2004). Contiki - a lightweight and flexible operating system for tiny networked sensors. In *LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, pages 455–462, Washington, DC, USA. IEEE Computer Society.