COOJA SIMULATOR USING CONTIKI OS

**Introduction:**

Contiki is an open source operating system, and can be freely used. Its full source code is available which can be used both in commercial and non-commercial systems for the Internet of Things. Tiny low-cost and/or low-power microcontrollers can be connected to the Internet through this software. It functions as a powerful toolbox for creating complex wireless systems. It makes powerful low-power Internet communication available while supporting fully standard IPv6 and IPv4, along with the recent low-power wireless standards. Since its applications are written in standard C, with the Cooja simulator, development is easy and fast.

Contiki networks could be emulated before burned into hardware, and a single download could provide an access to an entire development environment through Instant Contiki. The best part of this is that it can run on a range of low-power wireless devices, which can be easily purchased online and affordable by all.

COOJA is included in Contiki since version 2.0. In the Contiki source tree, COOJA resides in

* tools/cooja Simulator code
* platform/cooja Native platform with glue drivers for Cooja

**Get Started with Contiki**

Step 1:

Begin by downloading Instant Contiki, installing VMWare Player, and booting up Instant Contiki. Downloading the file may take a lot of time, after all the file size is 1GB! After you’re done, unzip the file and change the directory to the desktop.

Next, download and install VMWare. Though it requires registration, its free to download. Following this step, a reboot may be mandatory.

Start Instant Contiki by running InstantContiki2.6.vmx. Wait for the virtual Ubuntu Linux boot up.

Log in to Instant Contiki. The default password is user.

Step 2:

Now, you have got to compile and start the Contiki network simulator, Cooja. For this, open a terminal window.

Go to the Cooja directory, i.e, cd contiki/tools/cooja, and start Cooja with the command:

ant run

It may take some time for Cooja to compile itself. Once that’s done, a blue empty window would pop up.  
To well-verse in this, lets consider an example simulation.

Step 3:

Run Contiki in simulation.

First, create a new simulation by going to File🡪 New simulation. You may notice a ‘create new simulation’ dialog. Name the simulation or simply let it be as it is and click the ‘Create’ button.

Cooja creates a new simulation. All the motes in the simulated network are shown in the The Network window, at the top left of the screen. The Timeline window, at the bottom of the screen, lists all communication events in the simulation over time. This window stands very handy for understanding what goes on in the network. The right side of the screen contains the Mote output window which shows all serial port printouts from all the motes. The Notes window on the top right is where we can include notes for our simulation. And the Simulation control window basically is where we start, pause, and reload our simulation.

* Add motes to the simulation

We must add one or more motes in order to simulate our network. Find ‘Add motes’ on the Motes menu. Click Create new mote type since this is the first time we are creating the mote and selecet desirable mote type. For now, we click Sky Mote, to create an emulated Tmote Sky mote type.

You can choose a name for the mote type in the dialog that follows. Let’s stick to the suggested name and browse.

* Find example Contiki application

You could go to the directory /home/user/contiki/examples/ipv6/simple-udp-rpl to know how to do simple UDP communications over IPv6. It holds a number of Contiki applications.

* Specify application C source file

The file broadcast-example.c contains a simple Contiki application that randomly broadcasts a UDP packet to its neighbours. Click the button Open to choose the file.

* Compile Contiki and the application

Click the Compile button. Now Cooja will have to verify that the selected Contiki application compiles for the platform that we have selected. This will take at least a minute or two. The white panel at the bottom of the window will represent the compilation output.

If you want to create a mote type, you can do so and add it to the simulation, by indicating the number of motes to be added. The network window shows how many motes were added. You can click on the Start button to start the simulation.

Your first Cooja simulation with Contiki nodes is now created that sends periodic IPv6/UDP packets with ContikiMAC sleepy router functionality.

**Commandline options**

* -quickstar = Simulation Load Simulation at startup. This must be the first parameter.
* -nogui = Simulation Run simulation without GUI. This must be the first parameter.
* -applet = Run as applet. Must be the first parameter.
* -log4j = File Load Log4j-Config from file.
* -contiki = Path Set Contiki path
* -external\_tools\_config = File Set Cooja config file.

**Contiki features:**

Now that we have discussed the how it works, let’s focus on a few features of Contiki that makes it special and unique.

* Memory Allocation

Contiki is highly memory efficient and provides a set of mechanisms for memory allocation like memory block allocation memb, a managed memory allocator mmem, and standard C memory allocator malloc. It was designed especially for tiny systems, with a few kilobytes of memory available.

* Full IP Networking

Offering a full IP network stack with standard IP protocols such as UDP, TCP, and HTTP, in addition to the new low-power standards like 6lowpan, RPL, and CoAP, it is fully certified under the IPv6 Ready Logo program.

* Power Awareness

Low-power systems are suitable for Contiki, a pair of AA batteries are enough to run the system for years. Contiki also provides mechanisms for estimating the system power consumption of low-power systems and for understanding where the power was spent.

* Dynamic Module Loading

Contiki supports dynamic loading and linking of modules at run-time. This is useful in applications in which the behaviour is intended to be changed after deployment.

* The Cooja Network Simulator

Contiki devices often make up large wireless networks. Developing and debugging software for such networks is really hard. Cooja, the Contiki network simulator, makes this tremendously easier by providing a simulation environment that allows developers to both see their applications run in large-scale networks or in extreme detail on fully emulated hardware devices.

* Sleepy Routers

In wireless networks, nodes may need to relay messages from others to reach their destination. With Contiki, even relay nodes, so-called routers, can be battery-operated thanks to the ContikiMAC radio duty cycling mechanism which allows them to sleep between each relayed message. Some call this sleeping routers, we call it sleepy routers.

* Hardware Platforms

Contiki runs on a wide range of tiny platforms, ranging from 8051-powered systems-on-a-chip through the MSP430 and the AVR to a variety of ARM devices. There are also a number of more exotic platforms thrown in there for good measure. Read more about Contiki hardware platforms:

* Protothreads

To save memory but provide a nice control flow in the code, Contiki uses a mechanism called protothreads. Protothreads is a mixture of the event-driven and the multi-threaded programming mechanisms. With protothreads, event-handlers can be made to block, waiting for events to occur.

* Coffee flash file system

For devices that has an external flash memory chip, Contiki provides a lightweight flash file system, called Coffee. With Coffee, application programs can open, close, read from, write to, and append to files on the external flash, without having to worry about flash sectors needing to be erased before writing or flash wear-leveling. The performance of Coffee is within 95% of the raw throughput of the flash memory.

* The Contiki shell

Contiki provides an optional command-line shell with a set of commands that are useful during development and debugging of Contiki systems. With Unix-style pipelines, shell commands can be combined in powerful ways. Applications can define their own shell commands that work together with existing commands.

* The Rime Stack

In situations when bandwidth is at a premium or where the full IPv6 networking stack is overkill, Contiki provides a tailored wireless networking stack called Rime. The Rime stack supports simple operations such as sending a message to all neighbours or to a specified neighbour, as well as more complex mechanisms such as network flooding and address-free multi-hop semi-reliable scalable data collection. Everything runs with sleepy routers to save power.

* Standards and Hardware

Contiki operates on a range of different hardware platforms. This made it possible to port it easily to new hardware. Also, it was designed to use well-known and well-tested standards such as IPv4, IPv6, and HTTP.

* Open Source

Contiki is open source, available to all and can be used in both commercial and non-commercial systems, without restrictions.

* Community Support

Contiki is backed with a bunch of options for support. Besides, there are plenty of examples in the Contiki source code tree to help you get started with your own code.