Manual for the Cooja-TWIST plugin

The Cooja-TWIST plugin lets you use the TWIST testbed directly from the Cooja simulator. It makes it easier to upload your code and observe its execution remotely with all the power of Cooja. The code of the plugin is under a BSD licence, hosted as a contikiproject in the Contiki tree, i.e., if you download contiki, it is under `contikiprojects/sics.se/coojatwist`. The plugin works with both Unix systems and Windows.

A. Setup

1. Pull the latest Contiki from git:

```
git clone git://contiki.git.sourceforge.net/gitroot/contiki/contiki
```

2. Set your CONTIKI_HOME environment variable:

```
export CONTIKI_HOME=/... absolute path to your contiki .../
```

3. Compile Cooja

```
cd $CONTIKI_HOME/tools/cooja
ant jar
```

4. Get the Cooja-TWIST plugin:

```
svn co
https://contikiprojects.svn.sourceforge.net/svnroot/contikiprojects/sics.se/co
ojatwist coojatwist
```

5. Compile the plugin:

```
cd coojatwist
ant jar
```

6. Prepare your application. Some changes are required so Contiki uses the TinyOS serial output format:
`cp coojatwist/contikisources/uart1-putchar.c`

`$CONTIKI_HOME/cpu/msp430/dev/uart1-putchar.c`

Call the following line in your application to enable TinyOS frames:
`uart1_tinyos_frames(1);`

Compile application.

7. Start Cooja (`ant` from the Cooja directory), go into “Settings -> Cooja extensions”, find your coojatwist plugin directory, and select it. Click on save so this setting survives this cooja session.

8. Create an account on TWIST and book a job (slot)
`https://www.twist.tu-berlin.de:8000/`

9. Make sure ‘cURL’, with support for SSL, is installed in your system

B. Use

1. Create a new simulation via “File -> New simulation”

2. Open the plugin via “Tools -> TWIST Testbed”

3. Set the SSH password to `neniadgosp.` Set your TWIST username and account.

4. Generate TWIST motes in Cooja by choosing “Cooja actions -> Generate motes”

5. Connect the “mote output” plugin to the TWIST motes via “Cooja actions -> Connect motes”

6. Upload your firmware to TWIST via “TWIST actions -> Upload single binary to all motes”. Select the .sky file obtained by compiling your application.

7. Set the speed limit to 100%, start the “simulation”. The “mote output” plugin shows the actual motes output streamed in live. If your applications uses Contiki annotations, arrows and text can also be displayed in the “network” plugin.

C. Multi-binary support and node exclusions

1. In part B, all motes receive the same firmware. In this example, we will show how to specify different binary files for specific motes.

2. Open up a new TWIST Testbed plugin Window

3. Select “Cooja Actions → Generate motes” as previously

4. Use “Config Actions → Add Binary”. This will open up a file chooser identical to that of step 6 in part B, the only difference being that this will not automatically trigger the flashing process on TWIST. Instead, the binary file will be added to a list of firmwares. When the first binary is chosen, it is automatically applied to all nodes.

5. Use “Config Actions → Add Binary” a second time, and select a second binary file.

6. Notice that every node in the configuration table will be associated with the first firmware chosen, but that you can click on any line and select an alternative firmware from the drop-down list that appears. Change the binary for a few files.

7. To exclude a node from the flashing list, uncheck the associated checkbox.

8. Select “TWIST Actions → Upload binaries as set in table”, which will flash the binary files according to the selection in the table, and omitting all unchecked lines.

Acknowledgements

This work was funded by the EIT ICT Labs and the CONET Network of Excellence under the contract FP7-2007-2-224053 (CONET). We are thankful for the contributions of Sébastien Dawans from CETIC in Belgium.