Deluge T2 - Programming Manual

Chieh-Jan Mike Liang
Razvan Musaloiu-E.

May 21, 2007

1 Introduction

Deluge is a reliable data dissemination protocol for large objects, such as program binaries. Together with a bootloader, Deluge provides a way to reprogram sensor motes in a network. Deluge is maintained by Jonathan Hui, and Deluge 2.0 is the most recent version. Documentations on Deluge 2.0 are available at `http://www.cs.berkeley.edu/~jwhui/research/deluge/`.

Deluge T2 is an effort to port Deluge 2.0 from TinyOS 1 to TinyOS 2. Since the code from Deluge 2.0 is reused as much as possible, the behavior and the usage of Deluge T2 should be similar to Deluge 2.0. Having said that, it would be helpful to read the Deluge 2.0 manual and related documentations.

Deluge T2 is still in experimental phase. One current limitation is platform support. Deluge T2 has been developed and tested on tmote sky (*telosb*) only. In addition, Deluge T2 comes with 2 flash volumes by default. However, more volumes can be added, if necessary. There are also minor details that will be improved in future releases.

2 Quick Start

This section introduces the basics of reprogramming with an example. In addition, it provides a quick test for software prerequisite. The latest TinyOS 2 CVS tree and Python 2.4 with pySerial support are recommended for running Deluge T2.

To start the example, we run a `burn` script provided in `tinyos-2.x/apps/tests/deluge/Blink`. For example,

```
% ./burn /dev/ttyUSB0
```

This `burn` script programs the directly-connected mote with one version of blink. Then, it injects and reprograms the mote with another version of blink. At this point, you can try to retrieve program image versioning information. The script to interface with the mote is provided in `tinyos-2.x/tools/`. For example,

```
% tos-deluge.py /dev/ttyUSB0 -p 0
```

You should see something similar to the output below.

```
Pinging node ...
Connected to Deluge node.
--------------------------------------------------
Stored image 0
  Prog Name:   BlinkAppC
  Compiled On: Thu May 17 00:36:33 2007
  Platform:    telosb
  User ID:     mike
  Host Name:   sprite
  User Hash:   0xC50D8DA4L
```

```
    Num Pages:   24/24

    Size:        26512
    UID:         2302157803
    Version:     6
  --------------------------------------------------
```

The usage of `delugy.py` is available by running the script without any arguments, and it will be discussed in section 4.

3 Reprogramming a Network

This section illustrates the procedure to reprogram a network. Specifically, we will see how program images are injected and how versioning information is retrieved.

3.1 Setting Up the Motes

We first install both TOSBoot and a program that runs Deluge T2. For simplicity, we use the golden image as the program. The golden image is provided in `tinyos-2.x/apps/tests/deluge/GoldenImage`, and it does nothing except initializing Deluge T2. This step can be done by compiling and programming the mote normally. For example,

```
% make telosb install,0 bsl,/dev/ttyUSB0
```

Deluge T2 makes sure the mote ID remain persistent over image reprogramming. You can test the installation by interacting with the mote through `deluge.py`.

3.2 Preparing Your Application

In most cases, the only two files you need to modify are the top-level wiring file and the Make file. You need to make sure `DelugeC` component is included. In addition, the Make file should have the following lines:

```
TINYOS_NP=BNP
CFLAGS += -DTOSH_DATA_LENGTH=100
```

Finally, compile your application without installing it on the mote. For example,

```
% make telosb
```

3.3 Injecting Your Application

Before a program image is disseminated in the network, we need to first inject it to the base station. For example,

```
% tos-deluge.py /dev/ttyUSB0 -i 1 apps/Blink/build/telosb/tos_image.xml
```

You should see something similar to the output below.

```
Pinging node ...
Connected to Deluge nodes.
--------------------------------------------------
Stored image 1
  No proper Deluge image found!
--------------------------------------------------
Ihex read complete:
  Total bytes = 25526
  Sections = 2
```

```
--------------------------------------------------
Replace image with:
  Prog Name:   BlinkAppC
  Compiled On: Mon May 07 00:01:43 2007
  Platform:    telosb
  User ID:     mike
  Host Name:   sprite
  User Hash:   0xC50D8DA4L
  Num Pages:   24/24

  Size:        26512
  UID:         507153792
  Version:     0
--------------------------------------------------
```

3.4 Reprogramming with New Image

After you decide which program image you want to reprogram, you can first test on the base station by issuing the reboot command. For example,

```
% tos-deluge.py /dev/ttyUSB0 -r 1
```

After a few moments, the mote will begin quickly flashing the LEDs to signify the reprogramming process.

Now, you can have the base station disseminate a program image to the rest of the network. For example,

```
% tos-deluge.py /dev/ttyUSB0 -d 1
```

This command instructs the base station to notify the whole network of the availablility of a new program image. This notification is currently done via TinyOS dissemination service, and it triggers all motes in the network to get the new program image. Upon receiving the complete image over-the-air, each node automatically reboots and reprograms itself.

4 Deluge T2 Python Toolchain

Different from Deluge 2.0, Deluge T2 toolchain is written in Python. However, as demonstrated in the previous section, the usage is very similar.

4.1 -p –ping

This command is useful for checking the status of program images on a mote. It provides information such as program name, compile time, size of the image, and so on.

4.2 -i –inject

This command creates a program image from the supplied `tos_image.xml` file, and it injects the image into specified volume on the mote. All versioning information is kept on the mote, so no state is stored on the PC.

4.3 -r –reboot

This command sets up the mote to reprogram itself after reboot, and then it reboots the mote. This command is applicable only to the directly connected mote.

4.4 -d –dissemination

This command instructs the directly connected mote to disseminate an image to the network. This image is specified by the volume ID. Upon successfully receiving an image, motes in the network automatically reprogram themselves.

4.5 -e –erase

This command erases a flash volume on the directly connected mote.

4.6 -s –reset

This command resets versioning information of a specific image on the directly connected mote.