

Deluge T2 - Programming Manual

Chieh-Jan Mike Liang
Razvan Musaloiu-E.

January 14, 2008

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 nodes 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. Although the code from Deluge 2.0 is reused as much as possible, the behavior and the usage of Deluge T2 are not entirely identical to Deluge 2.0. Having said that, it would still 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 on Tmote Sky (*telosb*) and MicaZ only. In addition, Deluge T2 comes with 2 flash volumes by default. However, more volumes can be added, if necessary. There are also some minor details that will be improved in future releases.

2 Tools Installation

Deluge T2 requires a few Python scripts that not yet included in the official `tinyos-tools` RPM package. On the CVS, the scripts are located in `tinyos-2.x/tools/tinyos/misc`. The steps to install them are the following:

```
% cd $TOSROOT/tools
% ./Bootstrap
...
% ./configure
...
% cd tinyos/misc
% make ; make install
...
```

By default, the files will be installed in `/usr/local/bin`. If desired, the `--prefix` parameter from `configure` can be used to indicate a different path.

3 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 first compile `tosboot` provided in `tinyos-2.x/tos/lib/tosboot`. For example,

```
% make telosb
```

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

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

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/tinyos/misc`. For example,

```
% tos-deluge /dev/ttyUSB0 telosb -p 0
```

You should see something similar to the output below.

```
Pinging node ...
-----
Currently Executing:
  Prog Name:  BlinkAppC
  UID:       0xA274CBBE
  Compiled On: Mon Jan 14 13:09:38 2008
  Node ID:   1

Stored image 0
  Prog Name:  BlinkAppC
  UID:       0xD7EEFACE
  Compiled On: Mon Jan 14 12:40:06 2008
  Platform:  telosb
  User ID:   tinyos2
  Host Name: bluephase
  User Hash: 0x587C9C16
  Size:     33504
  Num Pages: 30
-----
```

The usage of `tos-deluge` is available by running the script without any arguments, and it will be discussed in section 5.

4 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.

4.1 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,

```
% CFLAGS=-DDELUGE_BASESTATION make telosb install,0 bs1,/dev/ttyUSB0
```

`CFLAGS=-DDELUGE_BASESTATION` indicates that the current mote will act as a base station, which requires an additional component to accept user commands from the serial port. Normally, only one mote in the network needs to be the base station, and other motes are reprogrammed over-the-air. If error occurs when running the command above, you might need to compile `tosboot` as shown in section 3. Deluge T2 makes sure the mote ID remain persistent over image reprogramming. You can test the installation by interacting with the mote through `tos-deluge`.

4.2 Preparing Your Application

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

```
TINYOS_NP=BNP
```

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

```
% make telosb
```

4.3 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 /dev/ttyUSB0 telosb -i 1 apps/Blink/build/telosb/tos_image.xml
```

You should see something similar to the output below.

```
Pinging node ...
Connected to Deluge nodes.
No valid image was detected.
-----
Create image: /home/tinyos2/local/bin/tos-build-deluge-image -i 1 build/telosb/tos_image.xml
Ihex read complete:
  35916 bytes starting at 0x4A00
   32 bytes starting at 0xFFE0
  35948 bytes in 2 sections
CRCs:
  0xCC0E 0xE57B 0xC224 0xE4FA 0x63EB 0xEBA5 0xE50B
  0x8073 0x345E 0x3BBF 0xF415 0xA7DB 0xA67C 0x6953
  0x6638 0x3D76 0x7218 0x0E8F 0x9C93 0xD679 0x7A50
  0xBFCD 0x8B57 0x7A47 0x323E 0x29FF 0x5EFF 0xB0AE
  0x1392 0x4A82 0xACFB 0x69D7 0x8C07
-----
36816 bytes in 28.23 seconds (1303.9932 bytes/s)
-----
Replace image with:
  Prog Name:   BlinkAppC
  UID:        0x76D2FB1B
  Compiled On: Sun Jan 13 21:47:54 2008
  Platform:   telosb
  User ID:    tinyos2
  Host Name:  bluephase
  User Hash:  0x587C9C16
  Size:       36816
  Num Pages:  33
-----
```

4.4 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 /dev/ttyUSB0 telosb -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 /dev/ttyUSB0 telosb -d 1
```

This command instructs the base station to notify the whole network of the availability 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. After all motes receive the image over-the-air, you can instruct the base station to disseminate the command to reprogram in the network. For example,

```
% tos-deluge /dev/ttyUSB0 telosb -dr 1
```

5 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 similar.

5.1 -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.

5.2 -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.

5.3 -r -reprogram

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

5.4 -d -disseminate

This command instructs the base station mote to disseminate an image to the network. This image is specified by the volume ID.

5.5 -dr -disseminate-and-reprogram

This command asks the motes in the network not only to disseminate an image but also to start running it. This is accomplished using a reboot.

5.6 -e -erase

This command erases a flash volume on the base station mote.

5.7 -s -stop

The effect of `-d` and `-dr` is continuous which means a new mote will become “infected” if he is nearby. This command stops the “infection”.

5.8 -ls -local-stop

When `-d` or `-dr` are in effect, the volume used by them is locked. This command can be used to unlock the volume in order to erase or inject a new image.