Deluge T2

From TinyOS Wiki

Note: This tutorial describes the Deluge T2 from the latest CVS.

Contents

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 here (http://www.cs.berkeley.edu/~jwhui/deluge/documentation.html) .

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 works on Tmote Sky (telosb), MicaZ , Iris and mulle. In addition, Deluge T2 comes with 4 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.

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.

The Deluge T2 tools needs PySerial (http://pyserial.sourceforge.net/) to talk to the serial port. In Debian the packet is called python-serial.

Quick Start

This section introduces the basics of reprogramming with an example. In addition, it provides a quick test for software prerequisite.

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, for a telos:

```
 % ./burn bsl,/dev/ttyUSB0 serial@/dev/ttyUSB0:115200 telosb
```

Or for a micaz:

```
 % ./burn mib510,/dev/ttyS0 serial@/dev/ttyS0:57600 micaz
```

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 serial@/dev/ttyUSB0:115200 -p 0
```

You should see something similar to the output below.

```
Pinging node ...
-------------------------------------------------
Currently Executing:
  Prog Name:   BlinkAppC
  UID:         0x2623906A
  Compiled On: Fri May 16 16:17:21 2008
  Node ID:     1

Stored image 1
  Prog Name:   BlinkAppC
  UID:         0x2623906A
  Compiled On: Fri May 16 16:17:21 2008
  Platform:    telosb
  User ID:     tinyos2
  Host Name:   bluephase
  User Hash:   0x587C9C16
  Size:        37920
  Num Pages:   34
-------------------------------------------------
```

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

Note: the MIB520 is providing two serial ports to the operating system. The first one is exclusively used to reprogram the mote while the second is used exclusively to talk to the user application running on the mote. The older MIB510 only provides one serial which is used for both operations.

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.

Setting Up the Basestation

A basestation is a special mote that will run a part of Deluge T2 that allows it to receive images over the serial and then disseminate them to the rest of the nodes. A basestation program is provided in tinyos-2.x/apps/tests/deluge/Basestation. The program only includes DelugeC components and uses the -DDELUGE_BASESTATION that enables the Deluge T2 basestation behavior. T2. This step can be done by compiling and programming the mote normally. For example,

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

Setting Up the Client Motes

We'll now install some initial code for the rest of the motes. For simplicity, we use the golden image as the program. The golden image is provided in tinyos-2.x/apps/tests/deluge/GoldenImage, and, similar with Basestation, it does nothing except initializing Deluge T2. This step can be done by compiling and programming the mote normally. For example:

```
% make telosb install,1 bsl,/dev/ttyUSB0
```

If the motes need to be queried over the serial about their state then the CFLAGS=-DDELUGE_LIGHT_BASESTATION needs to be used. For example:

```
% CFLAGS=-DDELUGE_LIGHT_BASESTATION make telosb install,1 bsl,/dev/ttyUSB0
```

Each mote (including the basestation) should have a different ID. Deluge T2 makes sure the mote ID remain persistent over image reprogramming.

Note: a mote will not attempt to reprogram itself if the voltage is below a certain threshold. For the current platforms this is 2.7V.

Preparing Your Application

In most cases, the only two files you need to modify are the top-level wiring file and the Makefile. First is to make sure that the application includes the DelugeC component:

```
components DelugeC;
```

In addition, the Makefile should have the following line:

```
 BOOTLOADER=tosboot
```

The applications should also contain a volumes-XXX.xml file that defines the GOLDENIMAGE, DELUGE1, DELUGE2, and DELUGE3. For AT45DB a correct volumes-at45db.xml (MicaZ, TelosA, Epic) is:

```
<volume_table>
  <volume name="GOLDENIMAGE" size="65536" base="0" />
  <volume name="DELUGE1" size="65536"/>
  <volume name="DELUGE2" size="65536"/>
  <volume name="DELUGE3" size="65536"/>
</volume_table>
```

For STM25P (TelobsB) a correct volumes-stm25p.xml is:

```
<volume_table>
  <volume name="GOLDENIMAGE" size="65536" base="983040" />
  <volume name="DELUGE1" size="65536"/>
  <volume name="DELUGE2" size="65536"/>
  <volume name="DELUGE3" size="65536"/>
</volume_table>
```

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

```
 % make telosb
```

Injecting a New Image

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

```
 % tos-deluge serial@/dev/ttyUSB0:115200 -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
  Prog Name:   BlinkAppC
  UID:         0x2623906A
  Compiled On: Fri May 16 16:17:21 2008
  Platform:    telosb
  User ID:     tinyos2
  Host Name:   bluephase
  User Hash:   0x587C9C16
  Size:        37920
  Num Pages:   34
--------------------------------------------------
Create image: /home/tinyos2/local/bin/tos-build-deluge-image -i 1 build/telosb/tos_image.xml
Ihex read complete:
  37372 bytes starting at 0x4A00
     32 bytes starting at 0xFFE0
  37404 bytes in 2 sections
CRCs:
  0x8065 0x45C9 0x951D 0x872D 0xF099 0xDC7B 0xD9A6
  0x1250 0xAE64 0xE0BB 0x43C3 0x17A0 0x2341 0x152A
  0xA317 0xC737 0xFB15 0x8164 0xD852 0x539C 0x4C3F
  0xE881 0x8D85 0x9F50 0xC379 0xFDAA 0xC91A 0x5037
  0xEC42 0x24A2 0x4AB8 0xEAD3 0x7A9F 0xAA5F
--------------------------------------------------
37920 bytes in 48.71 seconds (778.4516 bytes/s)
--------------------------------------------------
Replace image with:
  Prog Name:   BlinkAppC
```

```
   UID:         0x2623906A
   Compiled On: Fri May 16 16:17:21 2008
   Platform:    telosb
   User ID:     tinyos2
   Host Name:   bluephase
   User Hash:   0x587C9C16
   Size:        37920
   Num Pages:   34
 -------------------------------------------------
```

Reprogramming with a New Image

With the image in the external flash of the basestation we can now ask it to disseminate it to the rest of the network. For example,

```
 % tos-deluge serial@/dev/ttyUSB0:115200 -d 1
```

This command instructs the basestation to notify the whole network of the availability of a new program image. This notification is currently done via Drip, a 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 serial@/dev/ttyUSB0:115200 -dr 1
```

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.

-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.

-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.

-d --disseminate

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

-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 accomplish using a reboot.

-e --erase

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

-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*.

-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.

-r --reprogram

This command sets up the directly-connected mote to reprogram itself after reboot, and then it reboots the mote. This command doesn't make much sense since the basestation is a dedicated mote which can be reprogrammed using the serial port.