Deluge 2.0 - TinyOS Network Programming

Jonathan Hui

July 28, 2005

Contents

1 Introduction **2**
1.1 Deluge 2.0 2

2 Quick Start **3**
2.1 The Software 3
2.2 Setting Up Your TinyOS Node 3
2.2.1 Formatting the Flash 3
2.2.2 Installing Deluge 3
2.2.3 Pinging the Node 4
2.3 Installing the Golden Image 4
2.4 Preparing Your Code 5
2.5 Injecting a New Program Image 5
2.6 Reprogramming with a New Program Image 6
2.7 Congratulations 8

3 Deluge Java Toolchain **8**
3.1 Ping 8
3.2 Inject 8
3.3 Reboot 8
3.4 Erase 9
3.5 Reset 9
3.6 Dump 9

4 Network Programming API **9**
4.1 Deluge API 9
4.2 NetProg API 9

5 Network Programming Details **10**
5.1 TOSBoot 10
5.2 Golden Image 11
5.3 Theory of Operation 11

6 Frequently Asked Questions **12**

1 Introduction

Deluge provides an efficient method for disseminating large data objects, such as program binaries, to many nodes within a wireless sensor network. Combining this mechanism with a bootloader and command dissemination allows for the wireless programming of nodes. The following features are included with Deluge:

- **Multihop support:** Wirelessly program all nodes in a multihop network without touching your nodes.
- **Epidemic propagation:** Continuous propagation effort by all nodes helps ensure reachability of those nodes with intermittent connectivity.
- **Redundant data integrity checks:** Overlapping CRC calculations to help ensure integrity of program images at all nodes.
- **Store multiple program images:** With each node storing multiple program images, quickly switch your network between different programs without continuous downloading.
- **Golden image:** A program image with minimal support for network programming stored in a safe location on external flash. Always have a piece of code that will allow for recovery.
- **Isolated bootloader:** A piece of code that is guaranteed to execute after each reset independent of the TinyOS application. The bootloader is responsible for programming the microcontroller and recovers from programming errors by loading the Golden Image.
- **Rollback gesture:** Nodes crashed without setting a watchdog timer? Load the Golden Image simply by flicking the reset switch a few times and they're ready to accept a new program wirelessly.
- **Small RAM footprint:** Less than 150 bytes.

1.1 Deluge 2.0

Deluge 2.0 tackles lessons learned from its initial version. The main goal of this version is to enhance robustness and useability. Deluge 2.0 is currently supports the following platforms: Mica2, Mica2-dot, MicaZ, Telos rev. A, Telos rev. B, Tmote Sky, and TOSSIM platforms. The following is a list of notable improvements found in Deluge 2.0.

- Verify image before programming: TOSBoot first verifies the CRCs in the image before programming to help ensure the program is valid.
- Check system voltage before programming: Make sure the system voltage is above the required threshold to ensure that bits are written correctly.
- Rollback to Golden Image on failure: TOSBoot will automatically attempt to load the Golden Image if a programming attempt has failed, bringing the node into a network programmable state.
- Hardware write protect: Support for hardware write protection to prevent application code from modifying the Golden Image[1].
- Verbose information: Know everything about your program images including program name, compile time, hostname that compiled the code, and user that compiled the code.
- Protected data structures: Metadata data-structures include CRCs generated by the host PC to limit propagation of corrupt data-structures.
- Auto-detection of TOSBase: The Deluge java toolchain operates exactly the same whether a Deluge or TOSBase node is connect.
- Auto-resume incomplete images: Auto-resume of incomplete images, using program name, UNIX time, and user hash to identify identical images.

[1]Only available on some platforms

- Auto-detect identical images: Auto-detection of identical images so that images are not accidentally downloaded more than once by the user.
- Operator error protection: Metadata information clearly displayed and now requires user confirmation to continue execution.
- Reset images: Reset Deluge images before shifting your nodes from network to network and prevent unintentional infection "cross-pollination" between networks.
- Download Deluge images from any node: The program image can be downloaded from any node, including all metadata.

2 Quick Start

In this section, you will learn how to add network programming support to TinyOS hardware, your application, and how to inject a new program image and reboot a set of network programmable nodes.

2.1 The Software

Deluge 2.0 is included with TinyOS 1.1.14 and higher and we recommend that you start with this version of the distribution. You can also check out the latest version of the source tree directly from Sourceforge.

Deluge requires a bootloader, TOSBoot, to reprogram the node. To ensure that TOSBoot installs correctly when installing a TinyOS application, try building and installing `Blink` on a TinyOS node by going to `tinyos-1.x/apps/Blink` and typing the following:

```
% make <platform> install
```

where `<platform>` is the TinyOS platform you are working with. Once installed, reset your node by turning it off then on. You should notice TOSBoot's execution by displaying a count-down sequence on the LEDs, with the red LED turning off last. Once all the LEDs have turned off, the Blink application should start blinking the red LED.

2.2 Setting Up Your TinyOS Node

2.2.1 Formatting the Flash

Deluge 2.0 uses the newly proposed flash storage abstraction[2]. In the new abstraction, space allocation of flash memory is done by formatting the flash, much like formatting a hard-drive in a desktop computer. To format the flash for Deluge 2.0, compile and install the `FlashFormat` application provided in `tinyos-1.x/apps/TestDeluge`. After reset, the yellow LED will turn on to signify that formatting is in progress. After formatting, the green LED will turn on to signify success. Format time depends on the platform and can range between milliseconds and tens of seconds. If you see the red LED turn on, the formatting failed and you should post the situation to the TinyOS help mailing list.

2.2.2 Installing Deluge

Next, compile and install the `DelugeBasic` application provided in `tinyos-1.x/apps/TestDeluge`. The install process will install both the basic Deluge application and TOSBoot, the TinyOS bootloader for Deluge. Be sure to set the node ID appropriately when installing the application. For example:

```
% make <platform> install,9
```

will set the node ID to 9 when installing the application. Deluge will save the node ID so that it remains persistent across reboots between different program images.

[2]The details are outlined in TEP103 of the TinyOS 2.x working group

2.2.3 Pinging the Node

With Deluge installed, you can now use the Deluge java toolchain to make sure that Deluge is running on the node. First, make sure that either Serial Forwarder is running or the `MOTECOM` environment variable is set appropriately and that your node is directly connected to your PC via the UART (refer to the TinyOS tutorial for details). Then ping the node with the following command:

```
% java net.tinyos.tools.Deluge --ping
```

The ping response will display information about the currently executing image and what images are stored on your node. You should see something similar to the output below.

```
Pinging node ...
Connected to Deluge node.
--------------------------------------------------
  Currently Executing:
    Prog Name:   DelugeBasic
    Compiled On: Thu Jul 21 20:35:10 PDT 2005
    User Hash:   0x87916b92
  Stored Image 0 - (Golden Image)
    Prog Name:   N/A
    Compiled On: N/A
    Platform:    N/A
    User ID:     N/A
    Hostname:    N/A
    User Hash:   N/A
    Num Pages:   N/A
  Stored Image 1
    Prog Name:   N/A
    Compiled On: N/A
    Platform:    N/A
    User ID:     N/A
    Hostname:    N/A
    User Hash:   N/A
    Num Pages:   N/A
  Stored Image 2
    Prog Name:   N/A
    Compiled On: N/A
    Platform:    N/A
    User ID:     N/A
    Hostname:    N/A
    User Hash:   N/A
    Num Pages:   N/A
--------------------------------------------------
```

2.3 Installing the Golden Image

The Golden Image is a trusted TinyOS application that should contain minimal functionality to support Deluge. In this step, you will install `DelugeBasic` as the Golden Image. However, you may install your own TinyOS application as the Golden Image for your specific deployment. In the build directory for your platform, you will find a `tos_image.xml` file for the application. This file contains metadata about the application, as well as the binary code. To install the `tos_image.xml` binary for `DelugeBasic` into the Golden Image slot, execute the following command:

```
% java net.tinyos.tools.Deluge --inject --tosimage=tos_image.xml --imgnum=0
```

You will notice that the Deluge java toolchain will warn you about writing to the Golden Image slot. This is expected as the Golden Image is a special application and should only be overwritten when necessary. Simply confirm the operation and execution will continue.

After installing the Golden Image, you should repeat the process of installing Deluge and the Golden Image on a second node to demonstrate reprogramming over the air.

2.4 Preparing Your Code

In this section, you will modify `Blink` and add Deluge support to it. As you will find with many applications, the only file which requires modification is the top level wiring file, in this case `Blink.nc`. The modified version of this file is shown below:

```
 (1)  configuration Blink {
 (2)  }
 (3)  implementation {
 (4)    components Main, BlinkM, SingleTimer, LedsC, DelugeC;
 (5)    Main.StdControl -> DelugeC;
 (6)    Main.StdControl -> SingleTimer.StdControl;
 (7)    Main.StdControl -> BlinkM.StdControl;
 (8)    BlinkM.Timer -> SingleTimer.Timer;
 (9)    BlinkM.Leds -> LedsC;
(10)  }
```

In `Blink.nc`, DelugeC is added to the list of components in Line (4) and is wired to StdControl in Line (5). With these changes, compile Blink *without* installing it on the node as follows:

```
% make <platform>
```

2.5 Injecting a New Program Image

In this section, you will inject the newly compiled Blink application. Locate your `tos_image.xml` file. Using the Deluge java toolchain, execute the following command:

```
% java net.tinyos.tools.Deluge --inject --tosimage=tos_image.xml --imgnum=1
```

Note that Deluge allows a node to store multiple program images simultaneously. In this example, we are replacing image number 1. Again, when the Deluge java toolchain asks, confirm the continuation of this operation. The following is example output when executing this command. The exact values on the size of the binary, number of sections, and number of pages is platform dependent.

```
Pinging node ...
Connected to Deluge node.
--------------------------------------------------
Ihex read complete:
  Total bytes = 20386
  Sections = 2
--------------------------------------------------
Replace empty image with:
  Image: 1
    Prog Name:   Blink
    Compiled On: Thu Jul 21 20:39:24 PDT 2005
    Platform:    telosb
    User ID:     jwhui
    Hostname:    fx08722
    User Hash:   0x87916b92
Continue operation? (y/[n])
Injecting page [12] of [19] ...
--------------------------------------------------
```

With the new program image injected into the network, execute the ping command again. Your output, showing the presence of the newly injected image, should look similar to the following:

```
Pinging node ...
Connected to Deluge node.
--------------------------------------------------
  Currently Executing:
    Prog Name:   DelugeBasic
    Compiled On: Thu Jul 21 20:35:10 PDT 2005
    User Hash:   0x87916b92
  Stored Image 0 - (Golden Image)
    Prog Name:   DelugeBasic
    Compiled On: Thu Jul 21 20:35:10 PDT 2005
    Platform:    telosb
    User ID:     jwhui
    Hostname:    fx08722
    User Hash:   0x87916b92
    Num Pages:   19/19
  Stored Image 1
    Prog Name:   Blink
    Compiled On: Thu Jul 21 20:39:24 PDT 2005
    Platform:    telosb
    User ID:     jwhui
    Hostname:    fx08722
    User Hash:   0x87916b92
    Num Pages:   19/19
  Stored Image 2
    Prog Name:   N/A
    Compiled On: N/A
    Platform:    N/A
    User ID:     N/A
    Hostname:    N/A
    User Hash:   N/A
    Num Pages:   N/A
--------------------------------------------------
```

2.6 Reprogramming with a New Program Image

In this section, you will reprogram your network with the newly injected application. Execute the reboot command as follows:

```
% java net.tinyos.tools.Deluge --reboot --imgnum=1
```

Again, we specify which image to program with by setting `--imgnum` to 1. After a few moments, the node will begin quickly counting through the LEDs, signify the programming process. Once complete, the nodes will display the count-down sequence and execute `Blink`, demonstrating that the network successfully reprogrammed with the Blink application. Node specific TinyOS state, including `TOS_LOCAL_ADDRESS` and `TOS_GROUP_ID`, are saved and restored across reboots. The expected output after issuing the reboot command is as follows:

```
Pinging node ...
Connected to Deluge node.
--------------------------------------------------
  Reboot From Image:
    Prog Name:   DelugeBasic
    Compiled On: Thu Jul 21 20:35:10 PDT 2005
    User Hash:   0x87916b92
  To Image: 1
    Prog Name:   Blink
    Compiled On: Thu Jul 21 20:39:24 PDT 2005
    Platform:    telosb
    User ID:     jwhui
    Hostname:    fx08722
    User Hash:   0x87916b92
    Num Pages:   19/19
Continue operation? (y/[n]) y
Sending reboot message ...
Reboot message sent.
--------------------------------------------------
```

After rebooting, execute the ping command again. The output should signify that the currently executing image is `Blink` as follows:

```
Pinging node ...
Connected to Deluge node.
--------------------------------------------------
  Currently Executing:
    Prog Name:   Blink
    Compiled On: Thu Jul 21 20:39:24 PDT 2005
    User Hash:   0x87916b92
  Stored Image 0 - (Golden Image)
    Prog Name:   DelugeBasic
    Compiled On: Thu Jul 21 20:35:10 PDT 2005
    Platform:    telosb
    User ID:     jwhui
    Hostname:    fx08722
    User Hash:   0x87916b92
    Num Pages:   19/19
  Stored Image 1
    Prog Name:   Blink
    Compiled On: Thu Jul 21 20:39:24 PDT 2005
    Platform:    telosb
    User ID:     jwhui
    Hostname:    fx08722
    User Hash:   0x87916b92
    Num Pages:   19/19
  Stored Image 2
    Prog Name:   N/A
    Compiled On: N/A
    Platform:    N/A
    User ID:     N/A
    Hostname:    N/A
    User Hash:   N/A
    Num Pages:   N/A
--------------------------------------------------
```

You have just successfully reprogrammed a network of nodes wirelessly. To demonstrate the usefulness of the Golden Image slot, reset your node repeatedly in succession. After repeated resets, TOSBoot will flash all three LEDs simultaneously and reprogram your node. Connect this node to your computer and ping it using the Deluge java tool. You should see that its executing image is now `DelugeBasic` again.

2.7 Congratulations

You now have the basic tools and skill set to fully utilize Deluge to wirelessly program your nodes.

3 Deluge Java Toolchain

The Deluge java toolchain is a simple command line utility that allows interaction with nodes via a direct serial connection or through TOSBase. Either Serial Forwarder must be running or the MOTECOM environment variable must be appropriately set. The java toolchain provides the ability to ping the status of a node, inject a new image, inject a reboot command, and extract a program image.

3.1 Ping

You can ping the network with the following command:

```
% java net.tinyos.tools.Deluge --ping
```

The ping command is useful for checking the status of the network. A ping reply will provide information about the currently executing image and the status of each image stored in external flash. For each stored image, the following information is given:

- **Program Name**: the string of the top level TinyOS application component.
- **Compile Time**: the exact time at which the application was compiled.
- **Platform**: The TinyOS hardware platform the application was built for.
- **User ID**: The UNIX user login that compiled the application.
- **Hostname**: The UNIX hostname that compiled the application.
- **User Hash**: A hash of the User ID and Hostname.
- **Num Pages**: Size of the image in units of Deluge pages.

3.2 Inject

You can inject a new program image with the following command:

```
% java net.tinyos.tools.Deluge --inject --tosimage=<file> --imgnum=<imgnum>
```

A program image is injected by specifying the `--tosimage` option and a `tos_image.xml` file. The Deluge image to use is specified by `imgnum`. All versioning information is kept in the network and queried when needed. Thus, no state is kept on the desktop PC itself.

3.3 Reboot

You can reboot with a new program image with the following command:

```
% java net.tinyos.tools.Deluge --reboot --imgnum=<imgnum>
```

The Deluge image to reprogram the network with is specified `imgnum`. After issuing the command, it may take several seconds before the nodes begin programming themselves with the new image. The actual rebooting process is signified by the quick counting of the LEDs. Also, while the inject and reboot commands are separate, it is not necessary to wait for all nodes to receive the new program image before issuing the reboot command. Nodes will wait to download the entire image before reprogramming.

3.4 Erase

You can erase an image with the following command:

```
% java net.tinyos.tools.Deluge --erase --imgnum=<imgnum>
```

The erase command is most useful if you wish to cancel the propagation from an injected image.

3.5 Reset

You can reset the versioning information for a given image with the following command:

```
% java net.tinyos.tools.Deluge --reset --imgnum=<imgnum>
```

Removing versioning information for an image from a node ensures that unintentional "cross-pollination" will not occur, where a program image from one network will infect another. The reset command is not epidemic and only affects the currently connected node.

3.6 Dump

You can extract a program image with the following command:

```
% java net.tinyos.tools.Deluge --dump --imgnum=<imgnum> --outfile=<xml>
```

The dump command will extract the entire `tos_image.xml` file from the node. This command is especially useful for retrieving the program binary and metadata when needed and can be done with any node in the network that supports Deluge.

4 Network Programming API

This section outlines the application programming interface to Deluge and the network programming layer. These interfaces can be used to extend and customize the functionality of network programming. For example, functionality can easily be extended to support heterogeneous networks where nodes execute different binaries.

4.1 Deluge API

Deluge only provides the `StdControl` interface. Specifically, `StdControl.start()` and `StdControl.stop()` are used to specify when the Deluge service is active or not. Disabling the Deluge service also disables its epidemic property and nodes with the Deluge service disabled will not be able to propagate/receive new program images or reboot commands to/from neighboring nodes.

4.2 NetProg API

The NetProg component provides both `StdControl` and `NetProg` interfaces. The `NetProg` interface has the following specification:

```
interface NetProg {
  command result_t reboot();
  command result_t programImgAndReboot(uint8_t imgNum);
}
```

The `reboot` command is used to reboot the node with out reprogramming it. A successful call to `reboot` will not return and TOSBoot will begin the normal boot process.

The `programImgAndReboot` command takes an `imgNum` which specifies which Deluge image to program the node with. A successful call to `programImgAndReboot` will not return, as the node will program itself with the specified program and reboot. However, it will return `FAIL` if the program image is invalid. An invalid image usually means that the entire image has not yet been received by the node and is incomplete.

With the NetProg interface, the basic functionality of the network programming implementation can be extended. The default implementation is simple and assumes a homogeneous network with all nodes running the same application code. However, the `NetProg` interface allows for heterogeneous networks. Rather than using the Deluge java toolchain to inject reboot commands, the application code can directly reprogram the node to any stored program image. Deluge's multiple program image support allows different application code to coexist in the network. With some additional logic to determine which program image a specific node should use, a heterogeneous network with different application code can exist.

`NetProg.init()` is responsible for restoring necessary node state, including `TOS_LOCAL_ADDRESS` and `TOS_GROUP_ID`. Using these values before calling `NetProg.init()` can cause some unintended effects. We suggest wiring `NetProg.StdControl` to `Main.StdControl` and never reference `TOS_LOCAL_ADDRESS` and `TOS_GROUP_ID` in `StdControl.init()` of any other module.

5 Network Programming Details

This section provides a deeper presentation of the necessary mechanisms required to support network programming. Deluge provides an efficient method for disseminating large data objects, such as program binaries, to many nodes within a wireless sensor network. Combining Deluge with a bootloader (TOSBoot) and command dissemination allows for the wireless programming of nodes.

5.1 TOSBoot

The TinyOS implementation of the bootloader, TOSBoot, provides the set of mechanisms necessary to program the microcontroller with a stored program image. Because programming a microcontroller requires first erasing the program flash, installing a new application must be done carefully. Thus, TOSBoot takes into account many design decisions to achieve the highest level of safety that the hardware can provide.

TOSBoot is a self contained program that executes whenever the microcontroller exits the reset state. Being a self-contained application means that any non-volatile state left behind by TinyOS applications does not affect TOSBoot's execution. The installation of TOSBoot on a node only occurs through a physical connection and never over the network. TOSBoot does not require the use of an interrupt vector (with the exception of the power on reset), executing with interrupts disabled.

Parameters are passed to TOSBoot through non-volatile memory. Using non-volatile memory assures correct operation even during unexpected interruptions or power failures during the startup process. Parameters include the number of consecutive interruptions during startup, whether or not to program the microcontroller, and the location of the binary in external flash to program the microcontroller with.

As a visual notification of the boot process, TOSBoot initiates a count-down sequence on the LEDs if the platform supports it. When the count-down expires, it first checks if it should program the microcontroller with a new binary. If programming is requested, TOSBoot will erase the program flash and write the new binary to it. On completion, the programming bit is reset and TOSBoot jumps to the first instruction of the application. If no programming is requested, only the jump into the application is executed.

TOSBoot expects a program binary format that differs from standard `SREC` and `IHEX` formats. More standard formats include excessive addressing information, significantly increasing the size of the image. Instead, the TOSBoot binary format includes a single field for address and length for each contiguous block of data. The Deluge java toolchain removes redundant addressing information before injecting the image.

TOSBoot does a number of things to help protect the node from a wide range of failures during programming.

1. TOSBoot checks the supply voltage to ensure that it is at a safe level before programming the node. Many microcontrollers can operate on a supply voltage level that is lower than that required for writing to the program flash. If an attempt to write is made when the supply voltage is low, the result is undefined.

2. TOSBoot utilizes CRC information stored by Deluge to verify the image. Before programming TOSBoot verifies all CRCs. If the CRCs do not match, TOSBoot will abort the programming process and continue booting the node as normal.

3. TOSBoot checks the address of each write to avoid overwriting TOSBoot. While some microcontrollers provide a write-protect region of program flash, the msp430 microcontroller used on some TinyOS platforms does not write protect the program flash.

4. If an error occurs while programming the node, TOSBoot will retry a few more times. If a successful programming still does not occur, TOSBoot will try program the node with the Golden Image. By automatically reverting to the Golden Image, the node will be left in a network programmable state for recovery.

5.2 Golden Image

The Golden Image is a complete and trusted TinyOS application with the intention that it is stored at a read-only location in external flash. As a result of this work, platforms supporting hardware write-protection for the Golden Image are beginning to emerge. In the case of Tmote Sky platform, writes are only allowed if the USB port is connected to a desktop computer. Using the hardware write-protection, the Golden Image can only be installed through a physical connection. The purpose of the Golden Image is to have a mechanism for bringing the node into a recoverable state. Thus, the Golden Image provides minimal functionality to support Deluge, allowing the node to wireless reprogramming.

TOSBoot installs the Golden Image when the count of repeated resets exceeds a specified threshold. For example, an error in reading the program binary could occur while writing to program flash. On such an error, the node is reset and another attempt is made to program. When multiple attempts are made, the Golden Image is loaded. The reset count also allows a human to install the Golden Image by resetting the node multiple times consecutively, which we call the *Rollback Gesture*. The Rollback Gesture is especially useful in cases where a buggy application no longer accepts radio communication. The user can input the Rollback Gesture and inject a new program without the need of a physical connection to the desktop.

5.3 Theory of Operation

Ease of use is a primary design goal, trying to make management of network programming as simple and seamless as possible. A simple command-line java toolchain is used for all operations. The java toolchain may be used to ping the network and retrieve information about what images are in the network, inject new images, and inject commands to program the network with a new image. One important aspect of Deluge is that the network is viewed in aggregate as a single entity. When using the java toolchain to retrieve information, no information about which node replied is given. The assumption is that any node should provide information about the network's *eventual* state.

To ease management, each program image is tagged at compile time with metadata including: program name, compile time, user ID that compiled the program, the hostname the compilation occurred at, size of the image, and the amount of the image injected into the network. Thus metadata is included with the program image when it is propagated. Much of the metadata is useful just for identifying where the image came from. Some useful aspects are checking which code base the image was compiled against or who was responsible for the compilation.

The metadata also provides a convenient mechanism to uniquely identify program images. One use is automatic resume of partially injected images. The java toolchain will automatically detect an attempt to resume the injection of an identical image and set the version number accordingly. This is greatly useful when there are multiple injection points in the network. Users need not concern themselves with managing version numbers and ensuring that the version numbers are equal across the entire network.

In order to minimize operator error, user confirmation is required when requesting an expensive operation, such as injecting a new image that may overwrite a current one. One drawback of a simple command-line utility is its ease of issuing commands unintentionally. In recent deployments, we have witnessed these operator errors and can cause significant energy and time expense. When requesting an expensive operation, the java toolchain will provide information relevant to the context. In the case of injecting an image, information for both the image for injection and the image being replaced is displayed.

Deluge does not require any persistent state on the desktop PC. All necessary state is kept in the network. Synchronization occurs on each execution of the java toolchain and allows an administrator to switch desktop PCs without copying persistent state.

Deluge works equally well no matter which node is connected to the desktop PC. A program image can be physically injected into any Deluge node within the network. Furthermore, the Deluge node need not be connected to the rest of the wireless network during injection. With an inaccessible deployment, a node with a new injection can be dropped into the network and dissemination will begin. Additionally, the java toolchain works just as well over a TOSBase node, a bridge that forwards packets between the desktop PC and the wireless network. The java toolchain will automatically detect whether a Deluge or TOSBase node is connected and adjust its actions accordingly.

Finally, Deluge exports a very simple interface to extend its functionality. For example, Deluge can be enabled or disabled to control which nodes participate in the dissemination process. Additionally, nodes can decide which program image to use and when, thus allowing for heterogeneous networks where nodes execute different binaries. A detailed presentation of the API is given in Section 4.

6 Frequently Asked Questions

This section lists a set of common mistakes that users should avoid:

1. **I injected the reboot command, my node reboots, but does not reprogram itself to the new image.**

   Look carefully at the LEDs displayed by TOSBoot. If TOSBoot blinks the red LED three times before counting down, the system voltage is too low to safely reprogram the node. Replace your batteries and try again.

2. **Every time I program my node with a serial connection, it begins rebooting into a version previously injected with Deluge.**

   Deluge is an epidemic protocol. It ensures reliability and reachability by periodically polling its neighborhood to see if any nodes are inconsistent and brings them up-to-date if necessary. The best way to install program images on Deluge nodes is through the Deluge protocol itself.

3. **`TOS_LOCAL_ADDRESS` and `TOS_GROUP_ID` are not restored appropriately.**

   Important mote state including `TOS_LOCAL_ADDRESS` and `TOS_GROUP_ID` are restored when `NetProg.init()` is called. It is important to remember that `NetProg.init()` should be called before using any of these values. To ensure this behavior, wire `DelugeC.StdControl` or `NetProgC.StdControl` to `Main.StdControl` and never reference `TOS_LOCAL_ADDRESS` and `TOS_GROUP_ID` in `StdControl.init()` of any module.