The JHU testbed software is developed to help users install, use, maintain their testbeds easy. This software is following the HiNRG copyrights written in every source code. This page only tells you about the installation process and how to use it. For more details, please download the documentation [click] (If you have any questions or find bugs, please send an email to ljh@cs.jhu.edu)

0. Information

- Maintainer: Jong H. Lim
- Platform: NSLU2 (Cisco) for client, Any Intel-arch desktop for server
- OS: Kamikaze 7.09/8.09 (openwrt based) for NSLU2, Ubuntu Linux(kernel ver. 2.6.31) for server
- Programming Language: C for NSLU2, Python 2.6 for server

1. Development

If you want to add or chalinge.dtyrentrtestbeidgskatwikraz,exoushi09u,Idheipstiagl Kamimkazye oOf Swoinkyasurwcelh.Tpoted

2. Installation Process

2.1. Server

The server program we **scovide** is named 'MIB.py'. This is a Python scrip**screen**at no installation is nacces

```
>> screen    >> ./MIB.py    >> ctrl + a + d (to get out of the screen)    >> screen -r -d (to get in
```

2.2. Client

There are two ways to install the client program: 1) install with NSLU2 images or 2) with precompiled pa

2.2.1 Install with binary image.

One advantage of this method is that the image file includes everything you need to run the client softwa **upslug**. Please follow the instruction on the page. After the installation, all of

2.2.2 Install with pre-compiled packages

A user may not want to replace his or her OS with our image. In this case, you can install the packages that you follo **tinyos-telos-monitor_2.1-1_armeb.ipk**

```
>> scp tinyos-telos-monitor_2.1-1_armeb.ipk [your machine]   >> ssh [your NSLU2 box]   >> ip
```

The other package you need to install is the **cppbsl**. This program compiles the connected to an USB **tinyos-cppbsl_2.1-1_armeb.ip**

On top of this, you may need to install the module package if your device(mote) uses FTI **kmod-usb-serial-ftdi_2.6.26.8-ixp4xx-1_armeb.ipk**

3. How to use testbed

3.0. Conventions

The followings are the conventions defined by tos.py. Suppose you want communicate(i.e. read and wri

```
>> ./user.py network@magma.cs.jhu.edu:17003
```

In new testbed convention, the last three digits of TCP port implies TOS_NODE_ID of a mote. With the e

Suppose the mote's TOS_NODE_ID you want to read data from is 3. Then, by executing the following st

network@magma.isi.jhu.edu:17001

```
While True:      p = am.read()      print p.data
```

3.2. Write a data to a mote

For writing data to a mote, a user first needs to build a packet formatted in a serial ActiveMessage(called

```
if am.write(ampacket, 238) == True:

     print 'ack received'
```

With this command executed, tos.py sends ampacket to a mote 3 with AM ID 238, and waits for an ackn

3.3. Recompiling motes

To be able to recompile motes, you need to prepare two things: burn script(burn.all) and all file. The two

3.4. Remote resetting motes

As testbed motes are all around our building, manual reset would be painful. The reset.all script (which i

3.5. all file and scripts

We maintain two scripts (burn.all and reset.all) and all (MAC addresses and TOS_NODE_ID mapping fil