1 Initial Setup

This part applies to nims2, since that's the one that I set up from scratch. Plugged ethernet cable into port nearest the edge, and registered on the PNNL network by trying to get to google.com using Firefox. In the network control panel, set the dhcp client ID to "nims2" so PNNL's DNS server picks it up.

Setting the hostname requires changing it in /etc/hosts

```
sudo echo nims2 > /etc/hostname
```

and also editing /etc/hosts to point 127.0.1.1 at nims2:

```
127.0.0.1 localhost
127.0.1.1 nims1
```

I changed the editor from nano to vim.tiny, as follows. This affects commands like visudo and vipw, after I screwed up the /etc/sudoers file by not realizing that it had invoked nano.

```
sudo update-alternatives --config editor
There are 3 choices for the alternative editor (providing /usr/bin/editor).

Selection Path Priority Status

* 0 /bin/nano 40 auto mode
1 /bin/ed -100 manual mode
2 /bin/nano 40 manual mode
3 /usr/bin/vim.tiny 10 manual mode
```

For remote access, we also need to install the SSH server:

```
apt-get install openssh-server
```

At this point, it's basically functional on the network, and accessible as a headless server.

2 Users and Groups

I added a nims user and amaxwell user, for the NIMS executables and my user, respectively, using adduser foo (which creates home directories and sets up permissions appropriately). The nims user's password is the same as that of owner. I then set my user up with sudo access via sudo visudo, which invokes the vi editor. After adding the line

```
amaxwell ALL=(ALL) ALL
```

at the end of the file, amaxwell can execute commands as root.

I edited /etc/passwd manually to change UID and GID of nims as follows:

```
nims:x:200:200:NIMS user,,,:/home/nims:/bin/bash
```

Next, I edited /etc/group manually to change the nims group ID to 200 (this group was automatically created by adduser nims):

```
nims:x:200:
```

This signifies that **nims** is a system user, and prepares us for future usage of groups for filesystem permissions. Finally,

```
sudo chown nims:nims /home/nims
```

to fix the mess we just made with user and group ID.

3 Configuration

To set timezone:

```
dpkg-reconfigure tzdata
```

and follow the menu prompts. I set this to Los_Angeles, which is the tzdata name for Pacific time. Next, to install NTP support:

```
apt-get install system-config-date
apt-get install ntp
sudo system-config-date
```

Edit /etc/ntp.conf to include

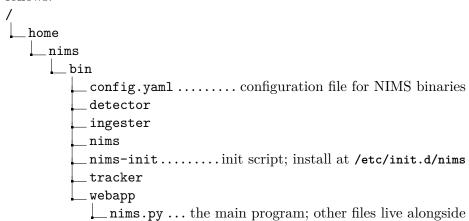
```
server time.apple.com iburst
server 130.20.248.2 iburst prefer
server 130.20.128.83 iburst prefer
```

since our network blocks outside NTP servers. Note that this will likely need to change for deployment; ideally, we'd have a GPS receiver onboard. Reboot or restart the ntp service after changing the config file.

4 Installation and Startup

The install-nims.sh script in the Subversion repository can be used to install all of the PNNL binaries and the webapp via scp (requires you to install the sshpass program on your build machine). It requires a single

argument, which is the password for the **nims** account, and installs programs as follows:



4.1 Startup

The init script must be copied manually, but it only changes if the path to the nims binary changes or the path to the webapp (nims.py) changes. To install the script,

```
cd ~nims/bin
sudo cp nims-init /etc/init.d/nims
```

will copy it to /etc/init.d and rename it; this latter step is important, in order for the service command to work as documented here.

For one-time activation on a new machine, use

```
/usr/sbin/update-rc.d nims start 99 2 3 4 5 . stop 1 0 1 6 .
```

This will set **nims** to start in runlevels 2–5 at S99, and stop in runlevels 0, 1, and 6 at K1 (last to start, first to stop in each runlevel).

For regular start/stop/restart of the nims system, you can use

```
sudo service nims [start|stop|restart]
```

from the command prompt as usual for system services. Note that the script must be run as root, but it will set each process to be owned by the nims unprivileged user.

4.2 Philosophy

Briefly, the reason for setting this up as a service is to allow for operation as an embedded system. I think it's best to run our services as unprivileged

processes, though, in order to avoid potential security issues. Since the system will be accessible remotely, we don't want webapp users or people just looking at data to accidentally reconfigure the network interface or disable the system.

This does require hardcoded paths, notably in the init script, and in the config file. It also causes some issues with permissions if you're not careful.

4.3 Logging

The nims binaries log all output to <code>/var/tmp</code> in a subfolder munged with the user's name and UID. For example, user <code>amaxwell</code> with UID 1000 has log files in <code>/var/tmp/NIMS-amaxwell-1000/log</code>. This allows multiple users to run the program without hitting permissions problems. Log files are rotated periodically, and the latest log file has a higher number appended. You can monitor these in the shell with

tail -f /var/tmp/NIMS-\$USER-\$UID/log/nims_0.log

for example.