NCL ThinClient Testbed

1 Introduction

The ThinClient testbed consists of five IBM Netfinity 4500R's: two packet monitors, one network simulator, a thinclient server and an application server. In addition, any number of client machines may be plugged in as needed. Figure 1 shows the current layout of the testbed.

The five testbed machines are running Debian Linux. Windows 2000 Server has been installed on the thin server and application server as of this writing.

Each of the Netfinity machines are set up to do packet forwarding. The nodes of the network without direct access to the Internet are routed through the network simulator via ip masquerading, which is running iptables. We are using 192.168.x.y addresses in the testbed. All of the consoles of the Netfinity machines are connected to a KVM switch. To select a specific console, press control then escape, which will bring up an On-Screen Display (OSD) menu. Press up and down arrows to select the console of interest and press return.

A testbed-wide account, *testbed* has been created, which should be used at all times to set up and run the tests. It's been set up with appropiate permissions where necessary. All the test-related data should be collected under its home directory.

The packet capturing software we are using is Ethereal 0.9.13, `http://www.ethereal.com`

The software being used as the network simulator is Nistnet 2.0.12, `http://snad.ncsl.nist.gov/itg/nistnet/`.

2 Platforms

Currently available thin client platforms are: X11, VNC, Citrix (ICA), Windows Terminal Services / Remote Desktop Protocol(RDP), and Tarantella. THINC and Sun Ray will be available at a future date.

The following are details about each platform, and instructions on setting up and running them.

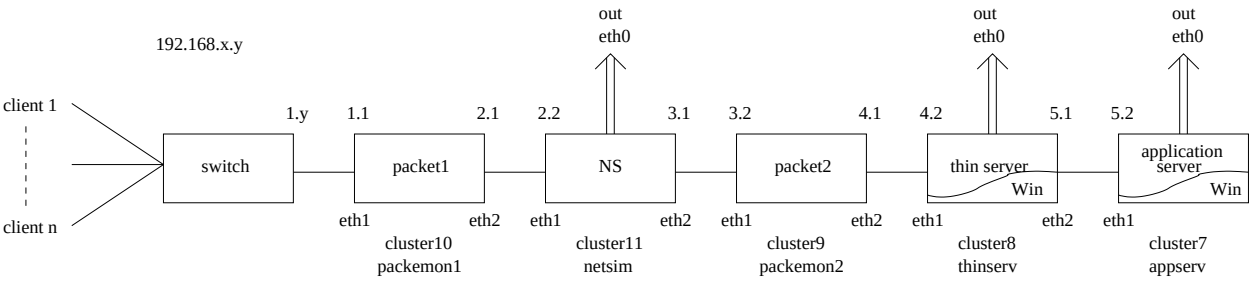

Figure 1: Testbed Layout

2.1 X11

We are using XFree86 4.3.0 (`http://www.xfree86.org`), the most popular implementation of X11. By default, X communicates on port `6000 + [display number]`. Normally, the display number is chosen automatically by X upon startup, based on its configuration and additional servers running in the computer. However, for the tests we have configured the thinclient to always use display 0, which makes the port that needs to be filtered with Ethereal, be `6000`.
To make sure the display is correct, in the client run:

```
testbed@thinclient1:~$ echo $DISPLAY
:0.0
```

If something different from `:0.0` is printed, the settings are incorrect and need to be fixed.

- To set up X for a benchmark session do the following:

  1. At the thinclient run:

     ```
     testbed@thinclient1:~$ xhost +thinserv
     thinserv being added to access control list
     ```

  2. From the thinclient connect to the thinserver using ssh:

     ```
     testbed@thinclient1:~$ ssh thinserv
     ```

  3. At the thinserver run:

     ```
     testbed@cluster08:~$ export DISPLAY=thinclient:0.0
     ```

  At this point, the session is ready to begin the benchmark (see section 4). One additional detail to take into consideration is that the benchmark program (e.g. Mozilla for the Web Benchmark) needs to be run from the same shell/session/terminal/console that you executed this last command. To make sure you are in the correct place, execute

  ```
  testbed@cluster08:~$ echo $DISPLAY
  thinclient:0.0
  ```

  If the output doesn't look right, execute the command above and check again.

- To shutdown the X session do the following:

  1. In the server, close all the benchmark programs which are using the thinclient as the display server. If you want to revert to your old settings, you may want to run:

     ```
     testbed@cluster08:~$ export DISPLAY=:0.0
     ```

     or something to that effect, although the easiest way would be to exit (close the terminal, logout of the console) and enter again (open another terminal, log back in). This will revert your settings to their default values.

  2. In the client, run

     ```
     testbed@thinclient1:~$ xhost -thinserv
     thinserv being removed from access control list
     ```

2.2 VNC

We are using RealVNC 3.3.7 (`http://www.realvnc.com`). By default, VNC uses port `5900 + [display number]`. Similar to X, the display number is chosen automatically by VNC upon startup, but for the tests we will force it to use display 1. Therefore, the Ethereal filter should be set up for port `5801`.

- To start a VNC session, do the following:

  1. From the thinclient connect to the thinserver using ssh:

     ```
     testbed@thinclient1:~$ ssh thinserv
     ```

  2. At the thinserver run:

     ```
     testbed@cluster08:~$ vncserver -depth 24 :1

     New 'X' desktop is cluster08:1

     Starting applications specified in /home/testbed/.vnc/xstartup
     Log file is /home/testbed/.vnc/cluster08:1.log
     ```

     At this point the vncserver is running on display 1, listening on port 5801, and running at color depth 24.

  3. At the thinclient run:

     ```
     testbed@thinclient1:~$ vncviewer thinserv:1
     ```

     After some informational output, it should ask for a password, which is the same as testbed's password, and then a window should open with the contents of the server's desktop. At this point the session is running, and everything is ready to begin the benchmark (see section 4).

- To shutdown a VNC session do the following:

  1. At the thinclient, close the vncviewer window.
  2. At the thinserver, run:

     ```
     testbed@cluster08:~$ vncserver -kill :1
     Killing Xvnc process ID 22475
     ```

     The process ID may change, but the output should look similar. If it complains that it can't find the process, the server may not be running, or you may have to kill it by hand.

2.3 Tarantella

2.4 Sun Ray

2.5 Citrix

We are using Citrix Metaframe XP 1.0 (`http://www.citrix.com`). The client is running Windows XP Professional, and the server is running Windows 2000 Server. By default it uses port 1494.

- To start a Citrix session, do the following:

  1. Start the thinclient in windows, and login as the testbed user.
  2. Start the thinserver in windows. There's no need to login, the service should be started automatically.

3. In the thinclient desktop, you'll see a shortcut which says "Citrix Program Neighborhoood". Double-click on it.
4. In the window that opened, double-click on the icon that says "thinserver".
5. A window should appear asking you to login. Use the testbed account to login and your thinserver desktop should appear.

At this point, the session is ready to begin the benchmark (see section 4).

- To end a Citrix session, go to "`Start->Shutdown->Log Off testbed`" in the thinclient window. The window should close automatically. Close all windows, log out from the thinclient and reboot the thinserver and thinclient to Linux.

2.6 Terminal Services

We are using Microsoft Terminal Services, running a Windows 2000 Server thinserver, and a Windows XP Professional thinclient. By default it uses port 3389. We use the terms RDP and Terminal Services interchangeably.

- To start a RDP session, do the following:

  1. Start the thinclient in windows, and login as the testbed user.
  2. Start the thinserver in windows. There's no need to login, the service should be started automatically.
  3. In the thinclient desktop, you'll see a shortcut which says "Remote Desktop Connection". Double-click on it.
  4. A window should appear asking you to login. Use the testbed account to login and your thinserver desktop should appear.

  At this point, the session is ready to begin the benchmark (see section 4).

- To end a Terminal Services session, go to "`Start->Shutdown->Log Off testbed`" in the client window. The window should close automatically. Close all windows, log out from the thinclient and reboot the thinserver and thinclient to Linux.

2.7 THINC

3 Network Simulation and Data Capturing

As mentioned before, the two programs being used for simulation and data capturing are *Nistnet* and *Ethereal*. For nistnet we have written some scripts to automatically configure the network for certain network characteristics. It is expected that as more tests are run, additional scripts will become available. The scripts can be found inside `~testbed/nistnet` in the network simulator, and should have the following naming scheme:

`nistnet_<bandwidth>_<delay>_<drop>`

For data capturing, we use filters which have the following form:

`tcp port <port> and host thinserv and host thinclient`

You should replace with the appropiate port as mentioned in section 2. Some of the standard ports may have been already saved as Ethereal filters. Click on the "Filters" button, and choose one from the list. If there isn't one for the platform you want, and this platform will be used again in the future, make sure to save the corresponding filter with the appropiate name. You'll thank yourself in the future.

To save space and facilitate the processing of the captured data, the sniffer should be set up so that it only captures the first 68 bytes of each packet. To do this, in the Capture dialog set "`Limit each packet to`" to the appropiate value. In addition, make sure the option "`Update list of packets in real time`" is not set.

When capturing has finished, save the data in libpcap format in the appropiate directory with a .pkt extension (e.g. `~testbed/data/<platform>/<name_describing_test-date>.pkt`). The data is ready to be analyzed as described in section 5.

3.1 Network Configurations

The following are the network configurations which have been tested:

- No additional latency (*LAN test*): 100Mbits, 10Mbits, 1.5Mbits, 768Kbits, 128Kbits
- Additional latency of 33ms (*I2 test*): 100Mbits, 10Mbits, 1.5Mbits, 768Kbits, 128Kbits

4 Benchmarks

This section contains the instructions to run all the different benchmarks.

4.1 Web Benchmark

The browser being used in the web benchmark is Mozilla 1.4 (`http://www.mozilla.org`). The URL to access the Web Benchmark is `http://appserv/ibench/`, which can be accessed through the home button in Mozilla.

In general, the steps to follow to run a test are the following.

1. Launch the thin client session, following the appropiate instructions found in section 2.
2. Launch mozilla from inside the thin client session, and clear its memory and disk cache. Make sure it fills the thinclient screen completely, and that the Navigation and Personal toolbars are hidden (uncheck in `View->Show/Hide`). Only the Menu and Status bar should be visible.
3. Set up packemon1 to filter for the appropriate protocol.
4. Set up netsim to the appropriate network characteristics
5. If using packet loss, then set up packemon2 to filter for appropriate protocol.
6. If benchmark you want to run needs parameters set, fill in the correct values for the number of cycles and the appropriate delay value.
7. Place mouse directly over the appropriate link or button.
8. Start packet captures.
9. Click appropriate link or button in mozilla's home page, according to the benchmark you want to run. Quickly move the mouse out of the display area of the browser to the toolbar.
10. After test is finished stop packet captures and save the captured data.
11. Remove network simulation settings (`nistnet_down`)

12. Shutdown thin client and server software, and log out from all the machines. To log out from the client machine, press Control-Alt-Backspace.

5 Packet Capture Analysis

The packet capture analysis script (analyze_pkt.pl) is located in `/usr/local/bin` in the packet capturing machines. To process the packet captures, you will need to determine latency of the benchmark using `analyze_pkt.pl`. The usage information is as follows, and will be printed any time the script is run without proper options and parameters:

```
Usage: analyze_pkt.pl [options] <filename>
Options:
-o (Print packet info to .txt file)
-b <break> (Default: 2)
-d <delay> (Default: 6)
-t <size threshold> (Default: 0)
-e <size endthreshold> (Default: 0)
-k <min packets in a page> (Default: 2)
-m <size minpage> (Default: 3)
-s <server IP addr> (Default: 192.168.4.2)
-c <client IP addr> (Default: 192.168.1.10)
-E (Only end page on server-to-client packet)

-p (Print detailed results)
-i (Print stray packet info)
-x (Print for importation into Excel)
-w (Warn for unusual results)
-g (Turn debugging messages on)
```

The default parameters of the following three options are usually fine for most tests.

`<break>` time is the maximum amount of time allowed within a page.
`<delay>` is the the amount of time between pages
`<size threshold>` is the minimum size of a packet that indicates the start of a new page.
This is an example run of the `analyze_pkt.pl` script:

```
analyze_pkt.pl -s 192.168.4.2 -c 192.168.1.10 citrix2k_lp_i2_d6_dflt_r1.txt

Average packet size: 854.294447150511
Number of pages detected: 110
Overall Results (includes results, but not blank or dropped)
----------------
Latency:   62.41
Data (s/c): 3404869
Data (c/s): 88512
Pkts (s/c): 2645
Pkts (c/s): 1383

Run 1 Results
---------------
Latency:   33.68
Data (s/c): 1723787
```

```
Data (c/s): 45888
Pkts (s/c): 1351
Pkts (c/s): 717

Run 2 Results
---------------
Latency:   28.21
Data (s/c): 1677009
Data (c/s): 42432
Pkts (s/c): 1288
Pkts (c/s): 663
```

The number of pages detected on the iBench test should be 55. Using the -w option will cause the script to print warnings about abnormal results. The `Latency` and the `Data (s/c)` from the Overall Results are those that are most likely to be of interest.

If a plain text file with packet information is needed, type:

```
$perl analyze_pkt.pl -o <capture_file>
```

This will create a file called ¡capture_file¿.txt (rather than .pkt) in the current directory. The format of the tab delimited text file will be:

```
<pkt num> <src ip> <src port> <dest ip> <dest port> <size> <timestamp> <delta time from last pkt> <prot
```