Chapter 5.0 The x-kernel configuration.

This chapter discusses on the issues of installation of the x-kernel on a Linux based operating system. The x-kernel goes through a series of installation steps before it can be used. This x-kernel requires to be installed, configured and built on a non-graphical environment. All of the installation steps for the x-kernel uses the Unix based command interface scripting. In order to execute the x-kernel simulator, the x-kernel needs to be installed first. The installation process for both the x-kernel and it simulator is described in the following topics.

5.1 The x-kernel installation steps

The x-kernel is downloaded from ftp://ftp.cs.Arizona.edu/xkernel/xkernel.tar.Z. (7.1MegaBytes(MB) compressed, 16.8MB uncompressed). The latest release of the *x*-kernel (version 3.3.1) is running either as a user-level program or as a network simulator, both on top of Unix. The following Unix platform is supported: Intel Pentium running Linux (Red Hat 5.0 / 2.0.32 Kernel)

Installation steps:

1. Create directory */usr/xkernel* and copy the compressed tar file from ftp://ftp.cs.arizona.edu/xkernel/xkernel.tar.Z into this directory. Then type:
   cd /usr/xkernel
   uncompress xkernel.tar.Z
   tar xf xkernel.tar
   This will create several subdirectories in */usr/xkernel*.

2. Create a directory in which an instance of the x-kernel will be built. This is called the *build* directory, and in general, each different configuration of the x-kernel has its own *build* directory.

cd /usr/xkernel/user level/build
mkdir linux

3. Copy configuration files into the *build* directory.
   cd /usr/xkernel/user level/build/linux
   cp ../Template/Makefile.linux Makefile
   cp ../Template/graph.comp .
   mkdir client server
   cp ../Template/rom.client client/rom
   cp ../Template/rom.server server/rom
   chmod 664 Makefile graph.comp client/rom server/rom

4. Edit the "*XRT* =" line of the *Makefile* in the *build* directory to reflect the root of the x-kernel source tree. Assuming your tree is at */usr/xkernel*, this means changing
   XRT = ../../..
   to
   XRT = /usr/xkernel

5. Edit the last line of the *graph.comp* file into the *build* directory to reflect the root of x-kernel tree. Assuming the tree is at */usr/kernel*, this means changing
   prottbl /cs/x33/etc/prottbl.std;
   to
   prottbl=/usr/xkernel/etc/prottbl.std;

6. Edit the two *rom* files to reflect the addresses of the two machines on which the x-kernel will be running. This will involve editing the Real IP address field of the two lines that begin with *arp(address resolution protocol)*. Specifically, edit *client/rom* by changing 192.12.69.186 in the first *arp* line to the IP address of machine A and by changing 192.12.69.35 in the second *arp* line to the IP address of machine B . Edit *server/rom* in exactly the same way. A and B can be the same machine, in this case, the same IP address is used in the *rom* file for both machines.

7. Put */usr/xkernel/bin/linux-x86* and */usr/xkernel/bin* in search path. They should appear before */bin* and */usr/bin* in order to pick up *GNU*(gnu's not Unix) *make* before the standard Unix *make*.

8. To build libraries necessary for the x-kernel.
   cd /usr/xkernel/user level/build/linux
   make system

9. To build the x-kernel.
   cd /usr/xkernel/user level/build/linux
   make compose
   make depend
   make

10. The x-kernel is ready to be executed. This will involve being logged onto both machines A and B. This is easiest to do by using a windows based interface and opening a separate terminal (shell) window on both A and B . First, on machine A type:
    cd /usr/xkernel/user level/build/linux/server
    ../xkernel -s
    This starts a version of the x-kernel on machine A that will act as a server for this run. Second, on machine B type:
    cd /usr/xkernel/user level/build/solaris/client
    ../xkernel -c128.1.2.3
    This starts a version of the x-kernel on machine B that will act as a client for this run. When the run is over, the x-kernel can be stopped by typing Control-C on both A and B.

The run involves the client establishing a TCP connection to the server, with a sequence of different-sized messages then sent back and forth between the client and the server. The time it takes to exchange 100 messages of each size between the client and the

server will be printed out. An example shown below shows the output on the client will consist of a sequence of line:

Protocol: TCP
Time: Fri Jun 6 16:16:11 2001
Host: root.localhost
Participant: client
Round Trips: 100
Message Length (bytes): 1
Times (sec):
select: Interrupted system call
0.061818
0.060986
0.063218

This means it took 0.061818 seconds (or 61ms) to send 100, 1-byte messages between the client and server machines, for an average of .61ms per round-trip.

5.2 The x-kernel simulator installation steps

1. Create a directory in which to will build an instance of *x-sim*. This is called as *build* directory and in general, each different configuration of the x-kernel and *x-sim* has its own *build* directory.

cd /usr/xkernel/simulator/build
mkdir linux

2. Copy configuration files into the *build* directory.

cd /usr/xkernel/simulator/build/linux
cp ../Template/Makefile.linux Makefile
cp ../Template/example/ .*
mkdir db
*chmod 664 **

3. Edit the "*\XRT* =" line of the *Makefile* into *build* directory to reflect the root of the x-kernel source tree. Assuming the tree is at */usr/xkernel*, this means changing *XRT* = *../../..*

to

XRT /usr/xkernel

4. Put */usr/xkernel/bin/linux* and */usr/xkernel/bin* in search path. They should appear before */bin* and */usr/bin* in order to pick up *GNU make* before the standard *Unix make*.

5. To build libraries necessary for *x-sim*.

cd /usr/xkernel/simulator/build/linux

make system

6. To build the *x-sim*.

cd /usr/xkernel/simulator/build/linux

make compose

make depend

make

7. To run the *x-sim*.

cd /usr/xkernel/simulator/build/linux

xsim