

MSHN Framework Prototype Components Users' Manual

Heterogeneous Computing Laboratory
Department of Computer Science
Naval Postgraduate School
Monterey, CA

November, 1998

United States Government Notice* :

"The Government retains Government Purpose Rights in the software, which means the rights to -

(i) Use, modify, reproduce, release, perform, display, or disclose technical data within the Government without restriction;

and

(ii) Release or disclose technical data outside the Government and authorize persons to whom release or disclosure has been made to use, modify, reproduce, release perform, display, or disclose that data for United States Government purposes."

* 48 Code of Federal Regulations 252.227-7013, 10-1-97 Edition

1.	INTRODUCTION.....	1
2.	OBTAINING THE SOURCE.....	1
3.	CONFIGURING THE INSTALLATION FILE: MSHN.INI.....	2
3.1.	CONFIGURING PATH NAMES (CLIENT_ROOT, VIS_ROOT, DOC_ROOT, CGI_ROOT):.....	3
3.2.	CONFIGURING BUILD TYPE (CLIENT_TYPE):.....	3
3.3.	CONFIGURING DEMO HOSTNAME (DEMO_HOST):.....	4
3.4.	CONFIGURING HOSTNAMES (HOST1 TO HOST5):.....	4
3.5.	CONFIGURING APPLICATION NAMES/FORMATS (APP1 TO APP5, FORM1, FORM2):.....	5
4.	BUILDING AND INSTALLING THE COMPONENTS.....	5
5.	SETTING UP THE WEB SERVER.....	7
6.	RUNNING THE FRAMEWORK COMPONENTS.....	7
6.1.	STEP 1A. USING A SIMULATED NETWORK AND MACHINES:.....	7
6.2.	STEP 1B. USING ACTUAL MACHINES:.....	8
6.3.	STEP 2. REGISTERING THE COMPONENTS AND STARTING THE ORBIX DAEMON:.....	8
6.4.	STEP 3. STARTING THE MSHN FRAMEWORK COMPONENTS:.....	9
7.	RUNNING THE VISUALIZER.....	11
8.	RESTARTING THE DEMONSTRATION.....	13

1. INTRODUCTION

The MSHN (Management System for Heterogeneous Networks) framework is a resource management system designed to schedule and passively monitor a collection of distributed and heterogeneous resources and applications in shared military environments so as to deliver optimal end-to-end QoS.

This document describes how to obtain, install and run three parts of the MSHN framework prototype. The three parts consist of (1) the CORBA-enabled framework prototype components themselves, (2) the visualizer, and (3) sample html and cgi-bin scripts for running the www-based demo. The procedures and code described herein were developed on a Sun Sparc 20 running Solaris 2.5. Everything has subsequently been successfully ported to Solaris 2.6. The framework components were built using Iona's Orbix 2.3c multithreaded. The visualizer code requires tcl/tk 8.0p with the BLT extensions. In order to use the www-based demo, a web server with cgi-bin capability is required. The demo was developed using Apache 1.2.6 but has also been successfully run on Apache 1.3.1.

There are essentially four steps described in this document: (1) Obtaining the source, (2) configuring it for your system, (3) building the framework components and installing the files appropriately, and (4) running the framework demo. Note: Screen output, code and commands to be typed at a terminal are denoted by `courier` font.

2. OBTAINING THE SOURCE

The current source file ("mshn_x.x.x.tar.gz", where x.x.x represents the major, minor and patch release versions, respectively) can be obtained from the MSHN website

(<http://www.mshn.org/downloads/framework/>), or by contacting webmaster@mshn.org.

Once obtained, place the “mshn_x.x.x.tar.gz” file in a temporary directory with at least 21 megabytes of space. Unzip and unarchive this file by typing something like the following:

```
gunzip mshn_x.x.x.tar.gz
```

```
tar -xvf mshn_x.x.x.tar
```

This will create a mshn_x.x.x/ directory structure with three subdirectories: framework/, visualizer/, and web_demo/

3. CONFIGURING THE INSTALLATION FILE: mshn.ini

Change to the mshn_x.x.x/ directory (cd mshn_x.x.x). Before starting, consult the README file for any amendments to this document. These instructions are intended for someone with experience installing Unix software. The files can be installed system-wide (if you have admin privileges) or within your own account. A familiarity with Iona Orbix and web server administration is necessary.

Now, using a text editor, set each configuration variable in the file “mshn.ini” appropriately for your system. A description of each variable is presented in the next 5 sections. When you finish editing the mshn.ini file you may want to make a hard copy of it for later reference.

3.1. Configuring path names (CLIENT_ROOT, VIS_ROOT, DOC_ROOT, CGI_ROOT):

CLIENT_ROOT is the path name where you want to install the binaries and supporting files for the framework components. VIS_ROOT is the path name where you want to install the tcl/tk scripts and supporting files for the visualizer components. DOC_ROOT is the path name where you want to install the .html and .mov files required for the web-based demo. CGI_ROOT is the path name where you want to install the Common Gateway Interface scripts required for the web-based demo. You must have “write” privileges on each directory you specify. If you chose to use the defaults, you must have root privileges.

3.2. Configuring build type (CLIENT_TYPE):

There are two modes in which the primary MSHN client (primaryclient) can be compiled. The first, “BYPING”, enables pinging of the individual demo machines. This method is used when there are real machines on the demo network. The second, “BYFILE”, enables simple reading from a machine status file. This method is used for a simulated network, where machine status is controlled via the file machStatus, which is created during installation. See **RUNNING THE FRAMEWORK COMPONENTS**, below for more information. Decide on a method that will be used and ensure that the corresponding “client_type” parameter is uncommented in the mshn.ini file.

3.3. Configuring demo hostname (DEMO_HOST):

This is the name of the machine on which the demo components, visualizers and web server will run during the demo. It can be one of HOST1 – HOST5, but it is not necessary for the demo to run.

3.4. Configuring hostnames (HOST1 to HOST5):

If you chose to use the “BYFILE” compile switch the demo will run regardless of your network environment (eg. On a stand-alone machine). HOST1-HOST5 can be simulated system names (they do not need to be names of actual hosts, although you should probably make one of the host names the machine on which you’re running the demo). HOST1 is the name of the machine on which the application at the lower format will appear to run (the less powerful computer). HOST2 is the name of the machine on which the application at the higher format will appear to run (the “super-computer,” which is initially off-line during the demo and is subsequently brought on-line).

If you use the “BYPING” compile switch, the host names you use need to be actual network aliases, or fully-qualified domain names, reachable from the machine on which the demo runs. No MSHN software will run on any of the machines other than the machine where the framework and visualizer components are installed (which must be the same machine) and where the web browser runs (which can be any machine with access to the web server). Also, when using the “BYPING” switch, HOST2 (the machine representing the “supercomputer”) should be able to be taken off line easily (e.g., by unplugging the network card).

3.5. Configuring application names/formats (APP1 to APP5, FORM1, FORM2):

The application names are somewhat arbitrary. The `mshn.ini` file contains some generic names corresponding to the types of applications that the MSHN project is targeting, but can be changed to suite the particular demonstration environment. The application that is simulated by this demo is defined as `APP1`. In order to change the name of the simulated application, change the value of `APP1` in the `mshn.ini` file.

`FORM1` and `FORM2` are the respective lower and higher format levels for the application you are using in the demonstration. Change them to suite the application defined to be `APP1`.

You can also change the simulated application's visual display output by replacing "`mshn_0.3.2/web_demo/html/app1.mov`" (which represents `APP1, FORM1`) and "`mshn_0.3.2/web_demo/html/app2.mov`" (which represents `APP1, FORM2`) with your own quicktime videos. `APP2-APP5` are arbitrary application names. Note: the only place `APP2-APP5` are seen during the demo is in the first page of the web-base demo (where you are asked to choose an application), and in the RRD instantiation of the visualizer. See **RUNNING THE FRAMEWORK COMPONENTS**, below for more information.

Finally, in the files `framework/src/Makefile` and `framework/src/platform.mk`, make sure that the paths and CORBA parameters are correct for your system.

4. BUILDING AND INSTALLING THE COMPONENTS

To build and install all three parts of this package first make sure you have write privileges for all paths indicated in the `mshn.ini` file. Generally, if the chosen directories

are within your home directory, permissions will not be a problem. Next, run the C Shell script, “install.sh” (`./install.sh`) in the main `mshn_x.x.x/` directory. This will, in turn, run three other install scripts, one in each of the primary directories (See below for details). Check the display for errors while this script runs. To selectively install one of the components, change to the appropriate directory and run `./install.sh` from that directory (not recommended). Note: in order to make application and machine name changes for future demonstrations, simply make your corrections to the `mshn.ini` file and rerun “install.sh” from the `mshn_x.x.x/` directory.

Running the install script will do the following, assuming “`mshn.ini`” is correctly configured:

1. Create files required by the framework components, build the executables and install them in the appropriate directory (indicated by `CLIENT_ROOT` in `mshn.ini`).
2. Create a configuration file for the visualizer and install the visualizer components in the appropriate directory (indicated by `VIS_ROOT` in `mshn.ini`).
3. Create all HTML files, image files and cgi files required by the web-based demo and install them in their appropriate directories (indicated by `DOC_ROOT` and `CGI_ROOT` in the `mshn.ini` file).

Depending on your system, this may take several minutes. Check the output for errors, make any necessary corrections and rerun the install script.

5. SETTING UP THE WEB SERVER

You must allow your web server to deliver documents from the MSHN html directory (indicated as DOC_ROOT in the mshn.ini file), and execute scripts in the MSHN cgi directory (indicated as CGI_ROOT in the mshn.ini file) as if they resided in your primary cgi-bin directory. There are two methods to accomplish this: (1) reconfigure your web server's configuration to point to these directories and (2) create symbolic links between the MSHN directories and your existing web server directories.

6. RUNNING THE FRAMEWORK COMPONENTS

There are three steps to run the framework components. Use step 1a if you chose to build the installation with the "BYFILE" switch and use step 1b if you chose to build the installation with the "BYPING" switch. There is also a runtime switch, "-p", which enables the demo to be run in a "pause" mode. The pause mode affects the daemonsrvr and sasrvr components. When enabled, these components pause, waiting for user input, at critical events, allowing a more controlled demonstration. The demo is typically run first from the user's perspective, with the components running in "non-pause" mode and then again from a behind-the-scenes perspective, with the components running in pause mode in order to show the interaction between components more clearly.

6.1. Step 1a. Using a simulated network and machines:

During installation, three files were created and placed in the framework installation directory, machStatus, machine.up and machine.down. The primaryclient will read from the machStatus file. The demo program determines which simulated machines

are available to MSHN based on the hostnames in this file. The machine.down and machine.up files are used alternately for copying over the machStatus file, which will change the status of HOST2. During the demonstration, HOST2 is brought up or down by copying the appropriate file over the machStatus file.

6.2. Step 1b. Using actual machines:

First, ping the localhost (by name) to verify that the local network interface is running. Then, to ensure that there is network connectivity throughout, ping each of the machines participating in the demonstration (indicated as HOST1 – HOST5 in mshn.ini). Make adjustments as necessary to ensure that each machine can reply to a simple network ping. Also, ensure that the machine indicated as HOST2 in the mshn.ini file can, without much effort, be made unavailable to the network. For example, this can be done by simply unplugging the network card. Test this to be sure that your network unavailability method works. During the demonstration you must be able to control whether or not this machine (HOST2) will answer a ping.

6.3. Step 2. Registering the components and starting the Orbix Daemon:

Ensure that the Orbix daemon (orbixd) is running. Now, change to the directory where the framework components are installed (CLIENT_ROOT) and register the MSHN components by typing:

```
regit.sh
```

This script file will register the framework components. You should see multiple connection reports followed by four detail reports from the Orbix daemon. If this process

does not work, examine the `regit.sh` file and inspect the lines of script. Apply any fixes as necessary. Make sure your environment variables are properly set according to your Orbix installation.

6.4. Step 3. Starting the MSHN framework components:

Again, make sure your environment variables are properly set according to your Orbix installation. You will now start each of the framework components. Note: each of the components is run in the foreground of its respective xterm (don't use the trailing “&” on any of the command lines).

Open 4 additional xterm windows and change to the directory where the framework components are located (`CLIENT_ROOT`). In the first xterm window, start the daemons server by typing:

```
./daemonserver
```

Place this window in the upper left hand corner of the desktop. Optionally, a “-p” switch can be given if the demonstration is to be run in pause mode. In the second window start the resource requirements database server by typing:

```
./rrdserver
```

Place this window in the lower left corner of the desktop. In the third window, start the resource status server by typing:

```
./rssserver
```

Place this window in the upper right corner of the desktop. In the fourth window, start the scheduling advisor server by typing:

```
./saserver
```

Place this window in the lower right corner of the desktop. Optionally, a “-p” switch can be given if the demonstration is to be run in pause mode. Following the start of each CORBA object, you should see a connection message followed by a message indicating that the respective server is available to the network.

Now, each of the MSHN components as well as the Orbix daemon should be running. At this point ensure that HOST2 is unreachable on the network (ie. unplug HOST2 if the “BYPING” switch was used, or copy machine.down into machStatus if the “BYFILE” switch was used). In the original xterm window change to the framework directory (CLIENT_ROOT) and start the primaryclient program, providing to it the log file to read from (`./primaryclient demo.log`).

The file demo.log was created during installation and contains sample resource updates for each of the machines. Now, answer the question about logreading speed. Generally, this would be a “1” (real-time based on the timestamps in demo.log). If run at real-time, demo.log will give you approximately 4 hours of RSS updates. Alternatively, you may wish to run the demo faster or slower than real-time. Position this window on the desktop so that it may be accessed from behind the other windows. At this point you should see the primaryclient executing. It will be either pinging or checking the ”machStatus” file and repeatedly calling the Resource Status Server short-term update function for each of the machines on the network.

7. RUNNING THE VISUALIZER

The visualizer is designed to show what's going on "behind the scenes" with MSHN. Each MSHN framework component keeps a running log of events taking place by that component. The visualizer is used for debugging and demonstration purposes.

There are three different modes that the visualizer can run in: Raw Text, Graphical, and Communication modes. In Graphical mode, there are four views available, one for each of the MSHN components: Client, SA, RSS and RRD view. Hence, there are six different ways to display what's going on with the MSHN components.

The visualizer scripts should be run from the same machine that the framework components are running. They can be run remotely via an xhost session (preferred) or from a different virtual terminal on the same computer that is hosting the MSHN components.

xhost session:

At the command prompt type:

```
xhost hostname
```

Where hostname is the name of the computer running the MSHN components. Note: this machine can not be HOST2. Now type:

```
telnet hostname
```

When the login is complete, type the following:

```
setenv DISPLAY remotename:0.0
```

Where remotename is the name of the computer from which you are “xhost”ing. Next, change to the directory (VIS_ROOT) where you installed the visualizer components and type:

```
./startall.sh &
```

This will invoke six versions of the visualizer, one for each view.

virtual terminal:

At the command prompt, change to the directory (VIS_ROOT) where you installed the visualizer components and type:

```
./startall.sh &
```

This will invoke six versions of the visualizer, one for each view.

You can also invoke each view one at a time (see startall.sh of mshnview.tcl for the proper syntax). Finally, if your windowing environment supports tk menu widgets, you can switch between the various views at runtime by using the menus.

Any of the above methods will launch the Tcl/Tk programs that display the state windows for each of the MSHN components. For clarity, the individual windows on the screen are usually arranged as follows: the scheduler window is in the lower right corner, the client window is in the upper left corner, the resource status server window is in the upper right corner and the resource requirements database window is in the lower left corner. The communication window is usually centered at the top, between the client and resource status windows. The raw text window can be minimized or put behind the other windows during the demonstration.

Assuming the log file output from the framework components is available to the visualizer, and assuming you have started the primaryclient, you will notice resource updates occurring within the resource status server state window. Note: the visualizer receives location information from the “mshnview.ini” file for the appropriate log files to read from. The “mshnview.ini” file was created automatically when you installed the components.

8. RESTARTING THE DEMONSTRATION

In the xterm window where the “primaryclient” is running, type ^C (control-C) to stop it from running. At the command prompt type:

```
./dunkit.sh
```

This is an executable script used to stop all other MSHN components. Now copy the machine.down file to the machStatus file (if you are using the BYFILE method) or unplug the HOST2 machine (if you are using the BYPING method). Follow the steps given in section 6.4: “**Starting the MSHN framework components**”.

If the demo is to be run again but under a different system account, un-register the MSHN components by typing:

```
./unregit.sh
```

from the directory where the framework components are installed (CLIENT_ROOT).