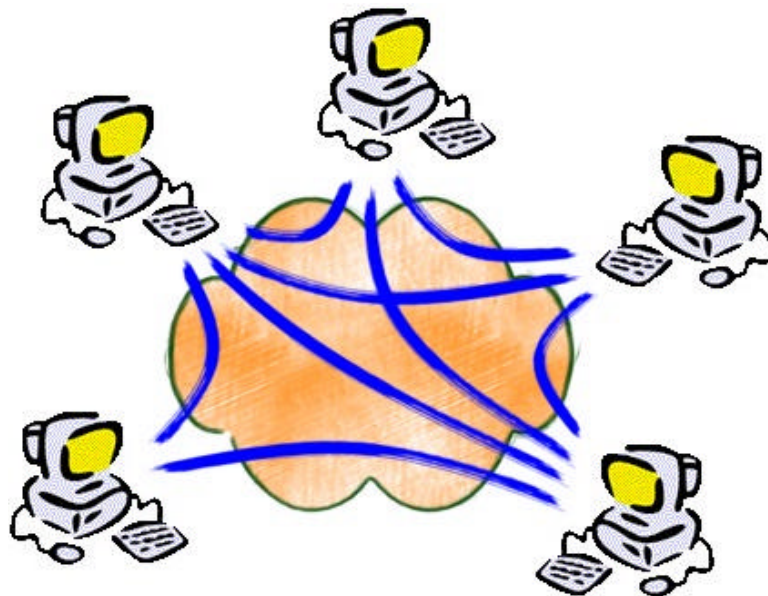


HyperCast 2.0

User Manual

Installation, Configuration, and Demo Applications



HyperCast Team
Department of Computer Science
University of Virginia

Email: hypercast@cs.virginia.edu
Web: <http://www.cs.virginia.edu/~hypercast>

January 2002

1	INTRODUCTION.....	2
2	GETTING STARTED	2
2.1	REQUIRED RESOURCES	2
2.2	DOWNLOADING AND INSTALLING HYPERCAST 2.0.....	2
2.3	RUNNING A DEMO PROGRAM : A DISTRIBUTED WHITEBOARD.....	4
3	CONFIGURING HYPERCAST 2.0 OVERLAY SOCKETS.....	5
3.1	A PRIMER ON HYPERCAST ATTRIBUTES	5
3.2	CONFIGURATION OF ANY HYPERCAST 2.0 OVERLAY NETWORK.....	6
3.3	CONFIGURATION FOR HYPERCUBE (HC) OVERLAY NETWORKS.....	7
3.4	CONFIGURATION OF DELAUNAY TRIANGULATION (DT) OVERLAY NETWORKS.....	8
4	CONFIGURING AN OVERLAY SERVER.....	9
5	RUNNING A “HELLOWORLD” PROGRAM.....	12
6	DEMO APPLICATIONS.....	13
6.1	MFTP (MULTICAST FILE TRANSFER PROTOCOL)	13
6.2	MEDIASTREAMER AND MEDIA RECEIVER.....	14
6.3	DISTRIBUTED WHITEBOARD.....	16
7	RUNNING MEASUREMENT EXPERIMENTS	17
7.1	STARTING RUNCONTROL.....	17
7.2	STARTING RUNSERVERS.....	18
7.3	RUNCONTROL COMMANDS.....	18
7.4	REMOTE TERMINAL FROM RUNCONTROL TO RUNSERVERS.....	19
8	APPENDIX: CONFIGURATION FILE “HYPERCAST.PROP”	20

1 Introduction

This User Manual contains a brief description for installing and configuring the HyperCast 2.0 software. The manual describes how to configure the software for the two overlay network topologies that have been implemented, i.e.,

- a logical hypercube where a rendezvous of nodes is done via IP multicast, and
- a Delaunay triangulation where a rendezvous is done with a server.

The user manual also describes how to run three simple demo applications, which are included in HyperCast 2.0. These applications are:

- a many-to-many file transfer program (MFTP);
- a media streaming application; and
- a distributed whiteboard.

In addition, we describe a set of tools for testing and monitoring overlay networks created by HyperCast 2.0.

2 Getting Started

2.1 Required Resources

HyperCast 2.0 is written in Sun Microsystems' Java™ programming language and can be used on any platform on which the Java Development Kit (JDK) version 1.2 or higher is installed. Information on downloading the JDK can be obtained from <http://java.sun.com>.

The storage requirements of the uncompressed HyperCast 2.0 software are less than 2MB.

2.2 Downloading and Installing HyperCast 2.0

Downloading

- The HyperCast 2.0 software is available from <http://www.cs.virginia.edu/hypercast/download.html>.
- Download one of the following two compressed archives. The files in the archives are identical:
 - hypercast2.0_release.zip (best for Windows platforms)
 - hypercast2.0_release.tar.gz (best for Unix platforms)

Installing on Windows:

1. Create a directory, e.g. "hypercast2.0", and download "hypercast2.0_release.zip" into this directory.
2. Extract "hypercast2.0_release.zip" into this directory with a package such as WinZip.

Installing on Unix :

1. Create a directory, e.g. "hypercast2.0", and download "hypercast2.0_release.tar.gz" into this directory.
2. Create a terminal (Unix shell or Windows command prompt) and change to the created directory.
3. Extract "hypercast2.0_release.tar.gz" with the following command:

```
gunzip -c hypercast2.0_release.tar.gz | tar -xvf
```

The archives "hypercast2.0_release.zip" and "hypercast2.0_release.tar.gz" are identical and contain the following files:

Table 1. Files contained in the downloaded archive.

hypercast2.0_release.jar	a jar file that contains all Java classes of the HyperCast 2.0 software
parser.jar	a jar file that contains the Java classes to process XML Documents
jaxp.jar	Java API for XML Parsing
hypercast.prop	The configuration file for HyperCast overlay sockets. The content of the configuration file is listed in the Appendix of this manual
User-manual.pdf	This User manual in PDF format
readme.txt	Readme file with instructions set up hypercast2.0
hypercast.prop.default	A backup copy of the configuration file "hypercast.prop".
HelloWorld_CallBack.java HelloWorld_NoCallBack.java	Source code of sample programs

2.3 Running a Demo Program: A Distributed Whiteboard

The following steps describe how to run a simple demo application that uses the HyperCast 2.0 software. The demo application is a distributed Whiteboard program. The topology of the overlay network is a Delaunay triangulation. Each Whiteboard application is started with the settings in the configuration file “hypercast.prop”. The content of the configuration file is included in the Appendix of this manual.

Instruction to run the demo program:

1. Open **three** terminal windows on the same host. In each terminal, change to the directory where the HyperCast software is located (e.g., “hypercast2.0”).
2. In the first terminal, start a so-called “DT server” with the command:

```
java -classpath hypercast2.0_release.jar
    edu.virginia.cs.mng.hypercast.DT.DT_Server
```

The command displays the IP address and port number of the DT server, e.g.,
 “DT Server Starts Successfully at 128.143.137.17/8081”

Note: The DT server helps new applications to rendezvous with an existing overlay network. The DT server is not involved in the exchange of application data.

3. In the second and third terminal, start the Whiteboard application with the following command:

```
java -classpath hypercast2.0_release.jar
    edu.virginia.cs.mng.hypercast.demo.Whiteboard
```

You will see an interface similar to Figure 1. Select a shape or other object, and draw in the canvas.

Note: To run the whiteboard on different hosts, perform the following after Step 2:

- Edit the configuration file “hypercast.prop” and modify the line
`NodeAdptUDPServer.UdpServer0 = 127.0.0.1/8081`
 Change the IP address and port number of the DT server entry, e.g.,
`NodeAdptUDPServer.UdpServer0 = 128.143.137.17/8081`
 , where “128.143.137.17/8081” is the value displayed by the DT server.

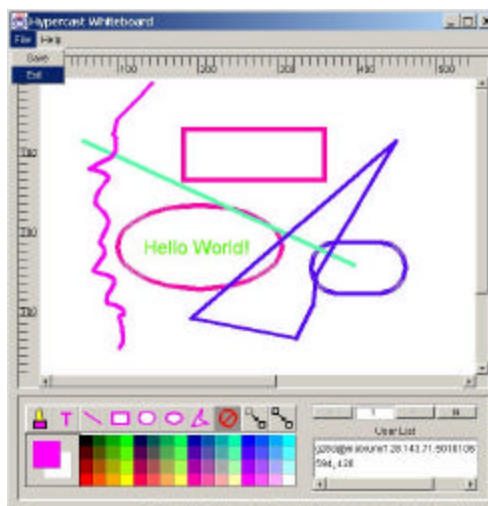


Figure 1. Whiteboard GUI.

3 Configuring HyperCast 2.0 Overlay Sockets

Each overlay socket in HyperCast 2.0 can be configured to run different overlay network topologies and different protocols for data transfer. The configuration of an HyperCast 2.0 overlay socket involves setting parameters, called attributes, in a configuration file. The default name of the configuration file is "hypercast.prop". The configuration file is read by an application when a new overlay socket is created. This section discusses the most important configuration options.

Note: The configuration of HyperCast overlay sockets can be simplified with the help of an overlay server. The use of overlay servers is discussed in Section 4. For now, we assume that no overlay server is used.

3.1 A Primer on HyperCast Attributes¹

The configuration of HyperCast is done by modifying attributes in the configuration file. HyperCast distinguishes two types of attributes: key attributes and configurable attributes. Key attributes are 'important attributes' of an overlay network, in the sense that applications that have the same values assigned to the key attributes should be able to communicate. The list of key attributes is part of the configuration of an overlay socket and is specified in the attribute "KeyAttributes". The attributes "overlay ID" and "KeyAttributes" are always key attributes, and are not included in "KeyAttributes".

The most important attribute of an overlay network is the overlay ID, which uniquely identifies a specific overlay network. Thus, the overlay ID can be used as a key to determine the other attributes of an overlay network. The HyperCast 2.0 protocol does not enforce a specific convention for selecting overlay IDs. Good choices for overlay IDs are discussed in Chapter 4 of the "HyperCast 2.0 Design Documents".

Note that many attributes have a default value. If an attribute is expected in the configuration file, but is missing, then the default value is selected. Default settings of attributes are explained in Appendix C of the HyperCast 2.0 Design Documents.

The following attributes select the overlay network topology and the protocol for data transfer. The attributes should be set as key attributes:

- **Selection of the overlay protocol (Attribute: "Node"):** The choices are the HC protocol ("Node=HC2-0"), which builds a logical hypercube overlay network, and the DT protocol ("Node=DT2-0"), which builds a Delaunay triangulation overlay network. The choice of the overlay protocol requires that additional attributes be set. For example, the HC protocol uses IP multicast for the rendezvous-process of new applications. Hence, an IP multicast address in one of the attributes.
- **Selection of the protocol for data transfer (Attribute: "SocketAdapter"):** This selection determines how application data is transferred between overlay sockets. The choices for the protocol are TCP and UDP. If the socket adapter uses TCP for data transfer, the configuration file has an entry "SocketAdapter=TCP". For UDP, the entry is "SocketAdapter=UDP".

¹ Refer to the Chapter "Overlay Network Management" of the "HyperCast 2.0 Design Documents" for an in-depth discussion of attributes.

3.2 Configuration of any HyperCast 2.0 Overlay Network

To set up a new HyperCast 2.0 overlay network, the following attributes are common to all overlay network topologies:

```
OverlayID      = <some unique name>
Socket         = HCAST2-0
SocketAdapter  = ScktAdptTCP (or ScktAdptUDP)
```

These attributes determine the overlay ID, the version of the overlay socket, and the type of socket adapter. The version number must be "HCAST2-0". The values of SocketAdapter must be "ScktAdptTCP" for data transfer with TCP, or "ScktAdptUDP" for data transfer with UDP.

There are two additional attributes for a log and an error messages. The default value for these attributes is "stderr", which displays log and error messages on the monitor. If the values are set to a file name, the messages are written to a file.

```
LogFileName = <file name>
ErrorFileName = <file name>
```

The following subsection explains sample configurations of the HC and DT protocols.

3.3 Configuration for Hypercube (HC) Overlay Networks

Hypercube (HC) overlay networks use IP multicast transmissions for the rendezvous process for adding new nodes to a hypercube and for preventing partitions. Therefore, the configuration file of an overlay socket that uses the hypercube protocol needs to contain an IP multicast address.

Note: IP Multicast must be available on the hosts that run this demo. Unless IP Multicast routing is available, the demo only runs on the local subnetwork. Refer to Section 3 of this manual, for other configuration options.

The attributes in the provided configuration file “hypercast.prop” are set to IP multicast address and port number 224.228.19.78/9472.

1. **Acquire an IP multicast address and port number:** To create a new HC overlay network, the application must acquire an IP multicast address and a port number. In the following we assume that the address and port number is 224.228.19.78/9472. This is the value in the “hypercast.prop” file.
2. **Set the configuration file:** The following attributes should be set when configuring an overlay socket for an HC overlay network.

```
KeyAttributes = Socket,SocketAdapter,Node,NodeAdapter,
               NodeAdptUDPMulticast.UDPMulticastAddress
Node          = HC2-0
NodeAdapter   = NodeAdptUDPMulticast
NodeAdptUDPMulticast.UDPMulticastAddress
               = <Multicast IP address/port number>
```

Comments:

- For hypercube overlay networks, it makes sense to use the acquired multicast IP address as the Overlay ID, e.g., OverlayID = 224.228.19.78/9472.
- There are several attributes that describe transport addresses, with an IP address and a port number. Here, the port number can be separated by a “/” or a “:”. So, both 224.228.19.78/9472 and 224.228.19.78:9472 are valid.
- The choice of the KeyAttributes is not enforced. However, the shown list of key attributes is recommend.
- Setting “Node=HC2-0” selects the HC protocol as the overlay protocol.²
- Choosing the HC protocol requires to select “NodeAdapter=NodeAdptUDPMulticast” and to set “NodeAdptUDPMulticast.UDPMulticastAddress” to an IP address and port number.
- When “Node=HC2-0”, the default value of “NodeAdapter” is “NodeAdptUDPMulticast”. Hence, if no value for “NodeAdapter” is set in the configuration file, the default value is chosen.

3. **Start applications:** Applications can be started immediately when the configuration file is set.

² Refer to the Chapter “Overlay Protocols” of the “HyperCast 2.0 Design Documents” for an in-depth discussion of attributes.

3.4 Configuration of Delaunay Triangulation (DT) Overlay Networks

The Delaunay triangulation (DT) overlay network of HyperCast 2.0 uses a server program, called DT server, for performing rendezvous functions. Therefore, when the DT protocol is used, one needs to start a DT server.

Note: Even though DT overlay networks require a DT server, they are the preferred overlay network topology, mainly, since IP multicast is not required. Also, DT overlay networks perform well in very large overlay networks.

1. **Set the configuration file:** The following are the changes to the provided configuration file “hypercast.prop” to run the DT protocol.

```
KeyAttributes = Socket,SocketAdapter,Node,NodeAdapter,
               NodeAdptUDPServer.UdpServer0
Node           = DT2-0
NodeAdapter   = NodeAdptUDPServer
NodeAdptUDPServer.UdpServer0
               = <IP address/port number of DT server>
```

Comments:

- “Node=DT2-0” selects the DT protocol as the overlay protocol. Choosing the DT protocol requires to select “NodeAdapter=NodeAdptUDPServer” and to set “NodeAdptUDPServer.UdpServer0” to the IP address and port number of the DT server.
 - When “Node=DT2-0”, the default value of “NodeAdapter” is “NodeAdptUDPServer”.
 - When 128.143.71.50:8081 is the IP address and port number of the DT server, then the setting should be NodeAdptUDPServer.UdpServer0=128.143.71.50:8081.
2. **Start a DT server:** The DT protocol performs the rendezvous process with the help of the DT server. For each overlay ID, there is one dedicated application, which acts as DT server. The DT server must be started before any other application in the overlay. A DT server is started with the command:

```
java -classpath hypercast2.0_release.jar
     edu.virginia.cs.mng.hypercast.DT.DT_Server
```

Recall that the IP address and the port number are stored in the configuration file in attribute “NodeAdptUDPServer.UdpServer0”. Since the IP address of the DT server must be the local address, the DT server ignores the IP address of the DT server in the configuration file.

3. **Start applications:** Once the DT server is running, applications can be started. Applications that cannot reach the DT server are not integrated in the overlay network.

4 Configuring an Overlay Server

An overlay server is a server that can help with the management of overlay IDs and overlay attributes, and helps with the configuration of application that use HyperCast overlay socket. For example, if an application does not know the attributes for an overlay network, the overlay server can provide these attributes, and ensure that the attributes used by all applications are compatible.

A HyperCast overlay server can be used (1) to create overlay IDs, (1) to query if an overlay network with a given Overlay ID exists or not, (2) to download the attributes of an overlay network.³ The overlay server is implemented as a minimal HTTP server. Refer to the chapter ‘Overlay Network Management’ of the ‘HyperCast 2.0 Design Documents’ for an in-depth discussion of overlay servers.

Any HyperCast 2.0 overlay network can be run with or without an overlay server. If no overlay server is used, all configuration information for an overlay socket is read from the local configuration file with default name ‘hypercast.prop’. Hence, the application users must ensure that the configuration files are compatible. (If all applications use the same configuration file, compatibility of the applications is always ensured.)

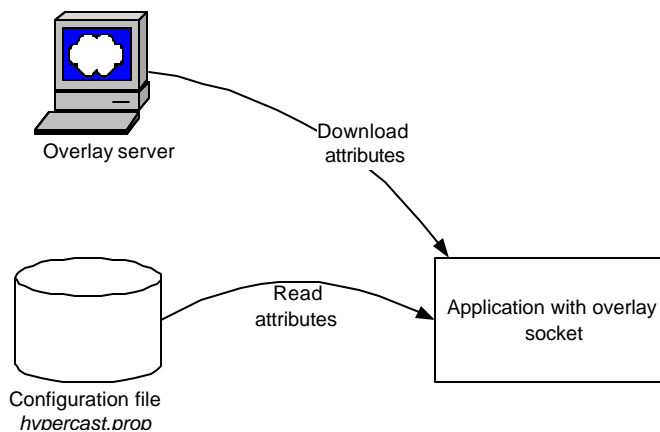


Figure 2. Applications read configuration parameters for an overlay socket from a configuration file and/or download configuration parameters from an overlay server. The configuration parameters are called attributes. The application reads attributes from the configuration file "hypercast.prop". If one of the attributes specifies an overlay server ("OverlayServer=HTTP"), the application downloads attributes from the overlay server. The downloaded attributes overwrite the values of the configuration file.

When an application starts an overlay socket it performs the following steps (see Figure 2): (1) It reads the attributes in the configuration file, and (2) if the configuration file specifies an overlay server, accesses the overlay server and send its key attributes to the server. The server distinguishes a few cases:

- a. If the configuration file of the application contains no Overlay ID, the server assumes that a new overlay network must be created and generates a new Overlay ID. The created overlay ID consists of a string, which contains, the IP address and port number of the server, and a timestamp. The Overlay ID creates a record that associates the key attributes submitted by the

³ In HyperCast 2.0, the overlay server only stores key attributes of an overlay network.

application. The created Overlay ID (together with the key attributes) is returned to the application.

- b. If the configuration file contains an Overlay ID and the overlay server has **no** record for this Overlay ID, the server assumes that a new overlay network must be created. Here, the overlay server creates a new record that associates the submitted Overlay ID with the submitted key attributes, and returns the Overlay ID with the key attributes to the application
- c. If the configuration file contains an Overlay ID and the overlay server **has** a record for this Overlay ID, the server returns the Overlay ID and the key attributes stored at the server to the application. Any download key attributes from the overlay server take precedence over values in the configuration file.

Manual creation of Overlay IDs with overlay server. A new overlay ID can be created manually by accessing the overlay server with a standard web browser, e.g., Netscape. Suppose the overlay server is set up as shown above. Then, access the overlay server at URL <http://128.143.137.16:8080>. The overlay server displays all known Overlay Ids. By typing a Overlay ID, say "newOverlay", and pushing the submit button, and new Overlay ID is created. The web interface of the overlay server is shown in Figure 3.

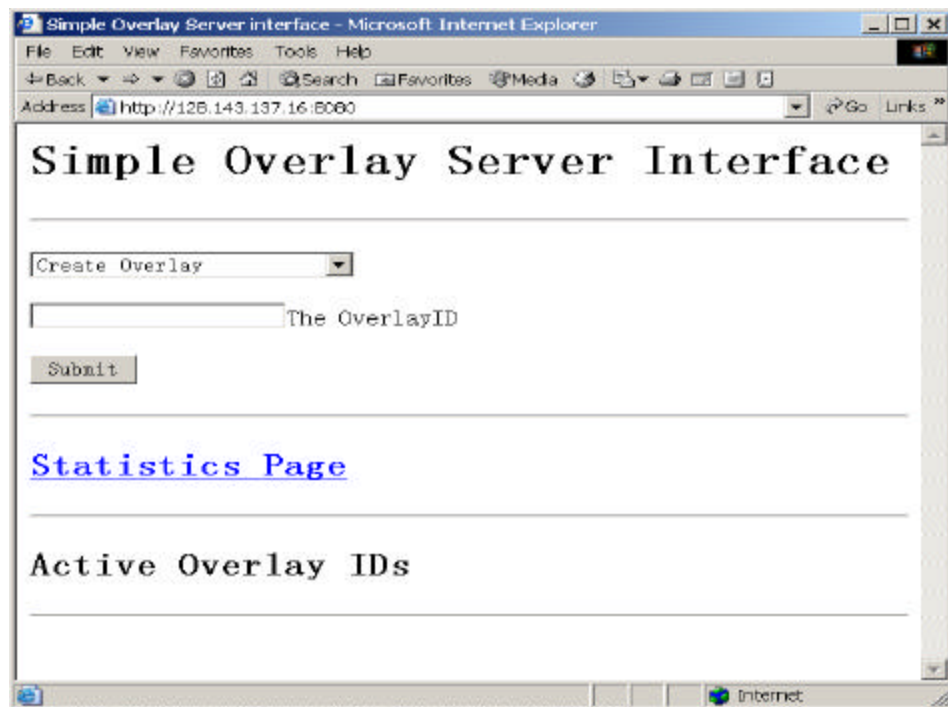


Figure 3. Website of the overlay server.

The following are instructions to use the overlay server.

1. **Start an overlay server:** An overlay server can be started on any host where the HyperCast software is installed. The command:

```
java -classpath hypercast2.0_release.jar
      edu.virginia.cs.mng.hypercast.HTTP_Server 8080
```

starts an overlay server at port number 8080.

2. **Set the configuration file:** Whether an overlay socket uses or does not use an overlay server is specified in the configuration file. For example, if an application uses an overlay server at IP address 128.143.137.16 and port 8080, the configuration file must contain the following entries:

```
OverlayServer = HTTP
HTTP.HttpServer0=http://128.143.137.16:8080/Overlays
If no overlay server is used, the line reads
```

```
OverlayServer =
```

, or the line is commented out as follows

```
#OverlayServer = HTTP
```

, or the line is missing.

3. **Start applications:** Once the overlay server is running, applications can be started. (Note: For DT overlay networks, a DT server must be started. The overlay server and DT server are independent entities.)
-

5 Running a “HelloWorld” Program

The software distribution contains the Java code of a simple application, called *HelloWorld*, which uses the overlay socket of HyperCast. The *HelloWorld* application creates an overlay socket that joins an overlay network. After joining the overlay, the applications multicasts the string “Hello World” (only once) in a message to the overlay network. All applications that receive the message display the content of the message on the screen. After the string is sent, the program waits in an infinite loop for “Hello World” strings sent from other members of the overlay network.

The downloaded software has two Java files with two versions of the HelloWorld program:

- HelloWorld_CallBack.java
- HelloWorld_NoCallBack.java

The difference between the two programs is explained in Part II of the design documents (see: <http://www.cs.virginia.edu/hypercast/documentation.html#Programming>).

The following explains how to compile and execute the HelloWorld program:

1. Create a terminal and change to the directory, where the HyperCast software is located.
2. Compile the Java program with

```
javac -classpath hypercast2.0_release.jar HelloWorld_CallBack.java
```

3. Within a short amount of time, execute the Java program in multiple terminals or multiple hosts. (If necessary, copy the compiled program HelloWorld_CallBack.class to remote hosts.) The program is executed with the command (In Unix, replace “;” by “:.”)

```
java -classpath hypercast2.0_release.jar ; .  
HelloWorld_CallBack
```

6 Demo Applications

The following three applications, MFTP, Media Streamer, and Whiteboard are part of the software distribution. The applications are not sophisticated and limited in features. All applications can run with the HC and DT overlay protocols.

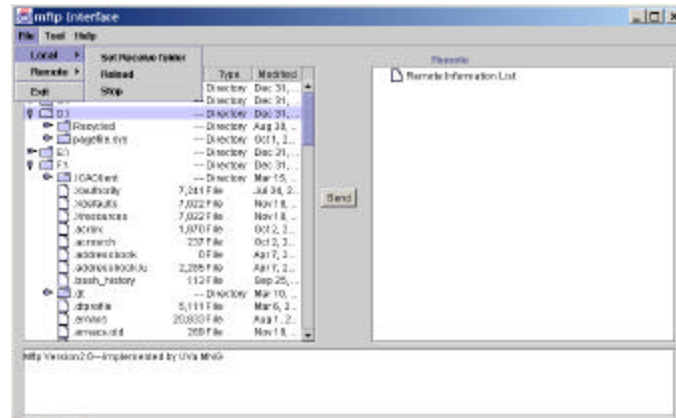


Figure 4. Graphical user interface (GUI) of the MFTP application.

6.1 MFTP (Multicast File Transfer Protocol)

This is a multicast file transfer protocol that can be run with or without a graphical user interface (GUI). The configuration files “hypercast.prop” must be set so that the overlay sockets use TCP for data transfer between neighbors in the overlay network, i.e., `SocketAdapter=ScktAdaptTCP`.

To start an MFTP client without a GUI (in Windows or Unix), type:

```
java -classpath hypercast2.0_release.jar
    edu.virginia.cs.mng.hypercast.demo.MFTP_Text send
```

To start an MFTP client with a GUI in Windows, type:

```
java -classpath hypercast2.0_release.jar
    edu.virginia.cs.mng.hypercast.demo.MNGmftp windows
```

To start an MFTP client with a GUI in Unix, type:

```
java -classpath hypercast2.0_release.jar
    edu.virginia.cs.mng.hypercast.demo.MNGmftp unix
```

Notes:

- The GUI is shown in Figure 4. The use of the GUI should be self-explanatory. The application provides a minimal help menu.
- By default, the MFTP program creates a directory called “MFTP” in the current working directory. This folder holds the files received from other MFTP applications connected to the overlay network. The directory store received files can be specified as follows:
 - If MFTP is started without a GUI, type the command “setdir directoryname”.
 - If MFTP is started with a GUI, you can set the receiving folder by selecting the “Set Receive Folder” sub-menu under the “File” menu.

6.2 MediaStreamer and MediaReceiver

This is a simple media streaming application. The MediaStreamer and MediaReceiver use UDP for data transfer between neighbors in the overlay network. The configuration files “hypercast.prop” must be set to `SocketAdapter=ScktAdptUDP`.

The MediaStreamer sends media files at a specified bit rate to a set of MediaReceivers. The MediaReceiver sends the received data to a local UDP port for playback. The MediaReceiver does not provide a media player. Therefore, some media player, e.g., MpegTV (available at <http://www.mpegTV.com>), is needed which can play media streams that are received from a UDP port.

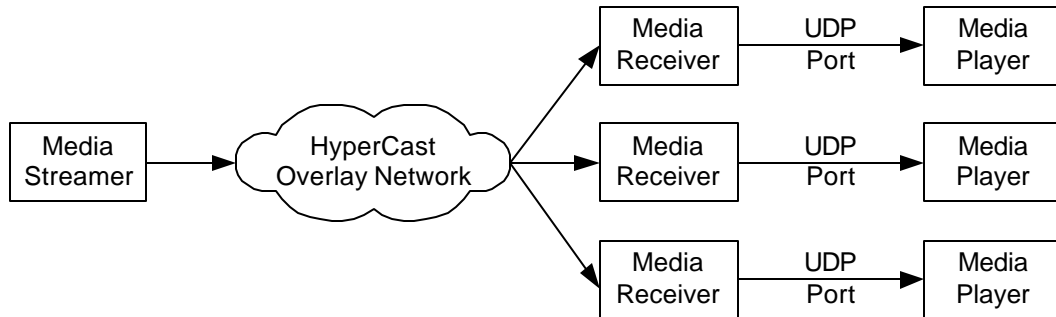


Figure 5. MediaStreamer and MediaReceiver applications. The media receiver forwards the data to a local UDP port. The Media Player is not part of the demo application. The Media Player can be any network-capable media player.

Starting a MediaReceiver

Usage:

```
java -classpath hypercast2.0_release.jar
      edu.virginia.cs.mng.hypercast.demo.MediaReceiver [#port number]
```

where `port number` is the port on which data is received (default: 6666).

Example : The command

```
java -classpath hypercast2.0_release.jar
      edu.virginia.cs.mng.hypercast.demo.MediaReceiver 9000
```

starts a MediaReceiver which receives an incoming media stream at UDP port 9000. If the playback is done by a tool such as MpegTV, the tool should be configured to receive data on port 9000.

Starting the MediaStreamer

Usage:

```
java -classpath hypercast2.0_release.jar
      edu.virginia.cs.mng.hypercast.demo.MediaStreamer
      [wait_time_to_start] [filename1] [#bitrate] [#numloop]
      [filename2] [#bitrate] [#numloop]
```

with:

- `wait_time_to_start` – Delay (in seconds) before the Media Streamer starts to transmit data (default: 30 seconds).
- `filename1`, `filename2`– Names of the media files.

-
- #bitrate – Bitrate in kbps. The default are 1500 kbps for MPEG-1 video (*.mpg) , and 40 kbps for MP3 music (*.mp3).
 - loop number – The number of times a file is transmitted by the MediaStreamer (default: 1).

Example:

- The command

```
java -classpath hypercast2.0_release.jar
     edu.virginia.cs.mng.hypercast.demo.MediaStreamer
     20 mymedia.mpg 1000 1
```

waits for 20 seconds before transmitting the file mymedia.mpg at 1000 kbps for a single time.

- The command

```
java -classpath hypercast2.0_release.jar
     MediaStreamer 60 mymedia1.mpg 1000 1 mymedia2.mpg 140 2
```

6.3 Distributed Whiteboard

A distributed whiteboard is a drawing utility that allows several users to share a drawing space. The distributed whiteboard demo application is shown in Figure 1. A user can draw items on the whiteboard, remove items, copy or move items to another location in the whiteboard, etc. If the application is run on several hosts, each drawing done on any whiteboard is displayed on all other whiteboards.

The configuration file "hypercast.prop" is set as described in Section 3. The distributed whiteboard runs with the DT protocol (Node=DT2-0) or the HC protocol (Node=DT2-0). The socket adapter can be set to TCP (SocketAdapter = ScktAdptTCP) or to UDP (SocketAdapter = ScktAdptUDP).

Start Whiteboard :

```
java -classpath hypercast2.0_release.jar
     edu.virginia.cs.mng.hypercast.demo.Whiteboard
```

Usage:

The whiteboard main window (see Figure 1) consists of several areas, which includes a drawing area, a Tool Chooser, a Color Chooser, a Pager and a User List.

The Tool Chooser is used to choose a tool for drawing. A tool is selected by clicking on the icon that represents the tool; then all following operations are done with this selected tool. The available drawing tools are BRUSH (🖌️), TEXT (📄), LINE (📏), RECTANGLE (📐), ROUND-RECTANGLE (📐), OVAL (📐) and POLYGON (📐). When REMOVE (🗑️) is selected clicking on an item in the drawing canvas removes the item. When MOVE (👉) is selected, a drag-and-drop mouse operations can be used to move an item. COPY (📄) is similar to MOVE, but it creates a new copy of the selected item.

The Color Chooser is used to select a color for drawing.

The Pager is used to create a new page and switch between pages. The whiteboard can contain several pages. Drawing on one page will not affect other pages. The current page number is shown in the textbox of the Pager. To create a new page, click "New Page" button. To switch between pages, click "<" or ">" buttons.

The User List shows all current users in the overlay. Each whiteboard is associated with a user, who is identified as "username@host+timestamp".

The **File/Save** menu saves the content of the canvas as a JPEG image.

7 Running Measurement Experiments

To run measurement experiments with HyperCast, there are two programs called RunControl (RC2) and RunServer (RS2). A host that runs a RunServer creates a specified number of overlay sockets. The RunControl program is a monitoring program, which communicates with the RunServer programs at different hosts to collect measurement data from the overlay sockets and to modify configuration parameters (see Figure 6).

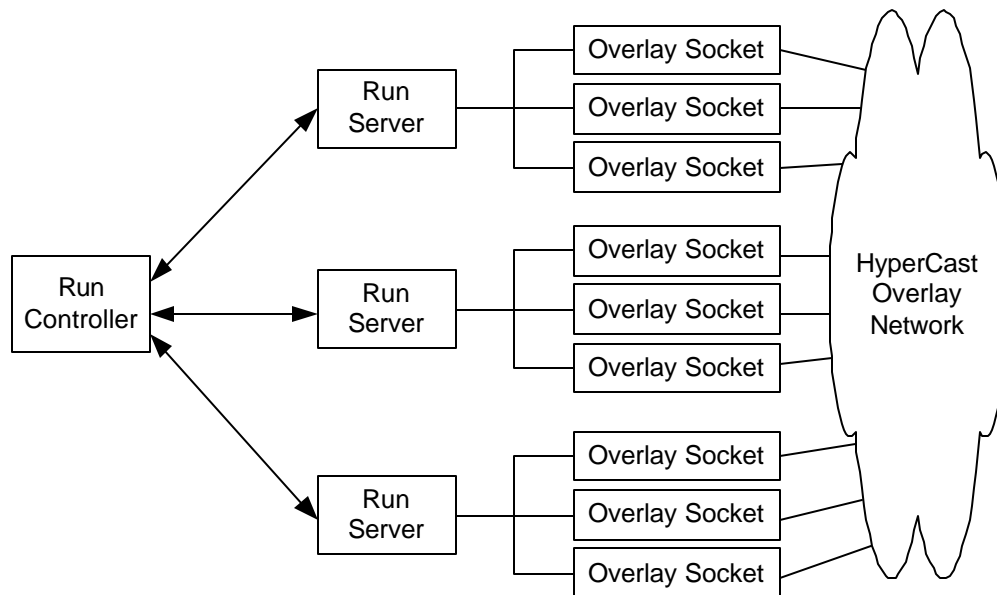


Figure 6. A RunServer can create several overlay sockets. The Runcontrol remotely monitors these sockets by communicating with the RunServers that started the overlay sockets.

7.1 Starting RunControl

The RunControl program is a console for the control of measurement experiments. The RunControl provides a command line interface that allows a user to control and measure data from experiments. The user can change the parameters of an experiment and collect data from the overlay sockets of RunServers that are connected to the RunControl. There are different versions of the RunControl program for experiments with Delaunay triangulation and hypercube overlay networks.

The RunControl program for hypercube overlay networks can be started as follows (on Unix systems, the “;” is replaced by “:”):

```
java -classpath hypercast2.0_release.jar;parser.jar;jaxp.jar
edu.virginia.cs.mng.hypercast.HC.HC_RC2 noserver 1500
```

The RunControl program for Delaunay triangulation overlay networks can be started as follows (on Unix systems, the “;” is replaced by “:”):

```
java -classpath hypercast2.0_release.jar;parser.jar;jaxp.jar
edu.virginia.cs.mng.hypercast.DT.DT_RC2 noserver 1500
```

In the preceding commands, "1500" is the port number through which the RunControl initially communicates with the RunServers. (The `noserver` argument indicates that RunControl does not use a server to automatically discover RunServers.) When the RunControl is started, it prints a line similar to the following:

```
Created noserver Socket at address: 128.143.137.16/1500
```

The displayed line gives the IP address and port number of the socket through which the RunServers can establish communication with RunControl. The user must initialize any RunServers that wish to use RunControl with this IP address and port number.

This line indicates the IP address and port number of the socket that is used for communication between RunControl and RunServers. When starting a new RunServer, you must provide this socket address.

Eventually, the RunControl program displays the following command prompt:

```
>
```

In Section 7.3, we discuss the commands that can be typed at the command prompt. The commands start a measurement experiment, collect data, and modify parameters of an experiment in execution.

7.2 Starting RunServers

A RunServer is started at each host that is involved in a measurement experiment. The commands to start a RunServer is identical for hypercube and Delaunay triangulation overlay network experiments, and is as follows (on Unix systems, the ";" is replaced by " "):

```
java -classpath hypercast2.0_release.jar;parser.jar;jaxp.jar
edu.virginia.cs.mng.hypercast.testing_and_monitoring.RS2
" " 128.143.137.16/1500 10
```

The command has three parameters. The first parameter is a string, which describes a set of attributes that override the attributes in the "hypercast.prop" file; An empty string ("") indicates that there are no changes. The second parameter ("128.143.137.16/1500") is the address of RunControl. The third parameter is the maximum number of overlay nodes that the RunServer will create.

7.3 RunControl Commands

We next discuss commands that can be typed at the command prompt of RunControl. The commands `help` and `help help` give a complete list of available commands. A simple experiment might use the following set of commands.

- `available`
Tells the number of available nodes running on all RunServers.
 - `create_experiment 10`
Tells one or more RunServers to start a total of 10 overlay sockets.
 - `start_transcript foo.xml`
Indicates that all messages from the RunServers will be written to in XML format to file "foo.xml".
 - `slow_start_experiment 5`
Tells all RunServers to generate new overlay sockets at a rate no greater than 5 overlay sockets per second.
-

-
- `wait_until_stable`
Forces RunControl to wait until the overlay network has stabilized. The command outputs the total time required by the overlay network to become stable.
 - `get_bw_results`
This command displays, for each overlay socket, the number of packets and bytes that were sent or received.
 - `stop_transcript`
This command closes the transcript file, e.g., "foo.xml".
 - `stop_experiment`
This command terminates the overlay sockets of the RunServers and terminates the RunServers.

7.4 Remote Terminal from RunControl to RunServers

The RunControl program can create a remote terminal (similar to telnet) to a RunServer. The remote terminal can be used to query or set the RunServer's local properties. RunControl can only run one remote terminal at a time. The ability to establish a remote terminal must be enabled by typing `start_telnet_service [port number]` at RunControl's command line prompt.

The following is a subset of the commands that can be used in a remote terminal session. The complete set of commands can be viewed by typing `help` at the RunControl's command prompt.

- `experiment_status`
Prints the current status of the experiment.
 - `start_new_sockets [NumberofSockets]`
Adds `NumberofSockets` new overlay sockets to the experiment.
-

8 Appendix: configuration file “hypercast.prop”

```
# This is the HyperCast Configuration File
#
# (c) University of Virginia 2001

#####
# LOG FILE:

# LogFileName =
LogFileName = stderr
# LogFileName = stdout
# LogFileName = hypercast.log

#####
# ERROR FILE:

# ErrorFileName =
ErrorFileName = stderr
# ErrorFileName = stdout
# ErrorFileName = hypercast.err

#####
# Overlay Server:

OverlayServer =
#OverlayServer = HTTP

HTTP.HTTPServer0=http://128.143.71.50:8080/Overlays

#####
# Overlay:

OverlayID = testOverlay

KeyAttributes = Socket,SocketAdapter,Node,NodeAdapter,
NodeAdptUDPServer.UdpServer0

#####
# SOCKET:

Socket = HCAST2-0

HCAST2-0.TTL = 255
HCAST2-0.ReceiveBufferSize = 200
HCAST2-0.ReadTimeout = 0

#####
# SOCKET ADAPTER:

SocketAdapter = ScktAdptTCP
#SocketAdapter = ScktAdptUDP

ScktAdptTCP.MaximumPacketLength = 16384
ScktAdptUDP.MaximumPacketLength = 16384
ScktAdptTCP.Timeout = 10000
ScktAdptTCP.MaxIdleTime = 2000
ScktAdptUDP.MessageBufferSize = 100
```

```
#####
```

```
# NODE:
```

```
Node = DT2-0
#Node = HC2-0
```

```
#DT2-0.Coords = 12345,67890
#DT2-0.Coords = USE_IP
DT2-0.Coords = RANDOM10000
DT2-0.TimeoutTime = 10000
DT2-0.SlowHeartbeatTime = 2000
DT2-0.FastHeartbeatTime = 250
DT2-0.ServerHeartbeat = 250
DT2-0.TryBackupServerTime = 10000
HC2-0.SleepTime = 400
HC2-0.MaxAge = 5
HC2-0.MaxMissingNeighbor = 10
HC2-0.MaxSuppressJoinBeacon = 3
```

```
#####
```

```
# NODE ADAPTER:
```

```
#
```

```
# The type of node adapter is implied by the
# selection of the Node
```

```
NodeAdapter= NodeAdptUDPServer
# NodeAdapter= NodeAdptUDPMulticast
```

```
NodeAdptUDPServer.MaximumPacketLength = 8192
NodeAdptUDPMulticast.MaximumPacketLength = 8192
NodeAdptUDPServer.MessageBufferSize = 18
NodeAdptUDPMulticast.MessageBufferSize = 18
```

```
NodeAdptUDPServer.UdpServer0 = 127.0.0.1:8081
NodeAdptUDPServer.MaxTransmissionTime = 1000
NodeAdptUDPMulticast.UDPMulticastAddress = 224.228.19.78/9472
```
