MBone Analysis Tool

Sneha K. Kasera*
kasera@cs.umass.edu

Mike Rizkalla
mrizkalla@tasc.com

Preliminary Draft

December 5, 1998

1 Introduction

Ever since its deployment, MBone has served as a test-bed for multicast applications and protocols. For the purpose of designing multicast applications, protocols and networks it is important to understand the dynamic behavior, especially loss and delay, on the Mbone. It is also important to understand the behavior of sources that individually or collectively generate the application packets. With this view, we have developed a measurement tool that would allow us to study the loss and delay behavior in any IP multicast network including MBone, and the source characteristics of the multicast applications that use this network.

Our measurement tool (called *mbat – MBone Analysis Tool*) is capable of simultaneously monitoring a multicast session (collecting relevant data) at several end hosts on the MBone. A pre-processing program which formats the collected binary data so as to make it ready for data analysis is also provided.

In this document, we first present a high level description of our data collection methodology and then describe the list of commands that are available for collecting and pre–processing data. We also provide some data collection examples. We then provide instructions for obtaining and installing *mbat*. Finally, we discuss some of the enhancements and modifications that can be made in *mbat*.

2 Data Collection Methodology

In this section we present a high level description of our measurement tool. The data collection phase consists of a control program and several copies of a data collection program. The control program running on an end host (control node) controls the data collection

*Corresponding author.

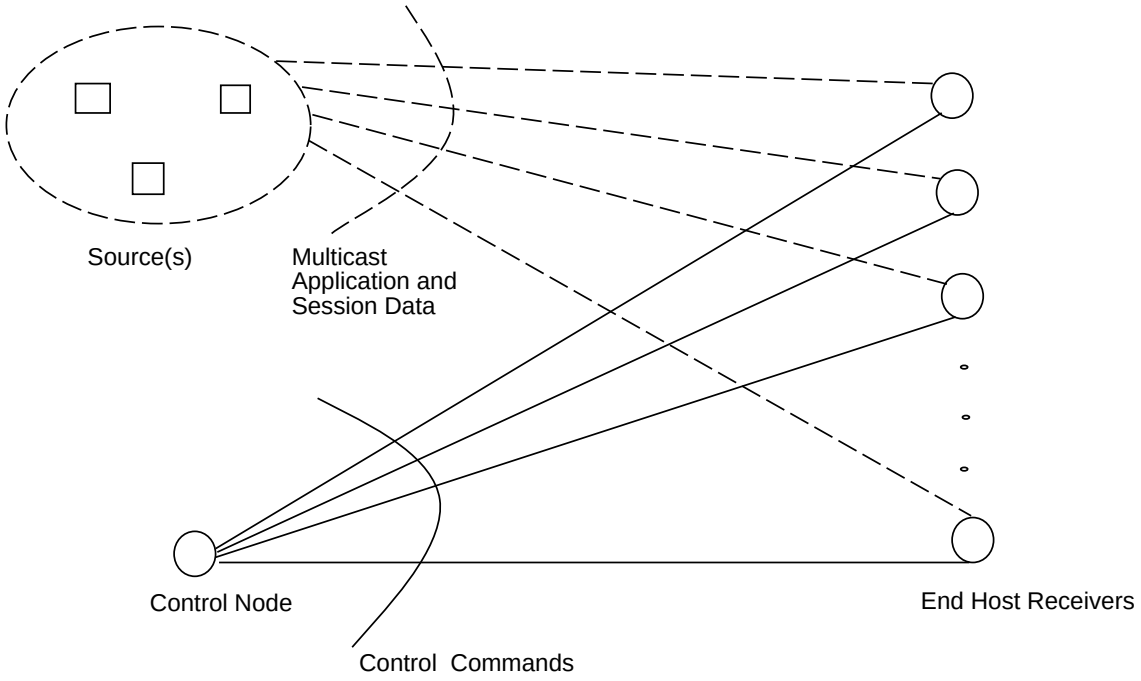


Figure 1: Data Collection Phase

programs running on end host receivers with the help of several control commands. These control commands are "reliably" multicast to the end host receivers. Reliable multicast of control commands is achieved through a simple positive acknowledgement-based scheme. The control node keeps the IP address of all the end host receivers that are collecting data. If it does not receive a positive acknowledgement in response to a command from any of the receivers, it retransmits the command. It continues to retransmit the command until all receivers send acknowledgements or until a certain maximum number (currently set to 20) of retransmissions are attempted. If the control node does not receive an acknowledgement from one or more receivers even after the maximum number of retransmissions are sent, it prints out the identity (IP addresses) of the unresponsive receivers at the controller. Figure 1 exhibits the data collection phase.

A **start** (or **stop**) command is used by the control node to ask the end host receivers to start (or stop) *listening* to a multicast session on the Mbone. When end receivers *listen* to a multicast session they record the necessary information, from the received packets, that is required (and all that can be obtained) to study the loss, delay and source behavior. For example, when end receivers listen to a multicast session that is using *vic* [3] they record the RTP headers [1] of the application packets. An RTP header provides the source identification, the packet sequence number, the source timestamp etc.. The length of the packet and the time at which it was received and the IP address of the source of the packet is also recorded.

Our tool is RTP capable and can record applications such as *vic, vat and nv.* In addition to recording information on the data packets our tool also records information on the low

rate RTCP [1] packets.

In *mbat*, the time at which a packet is received (receive timestamp) could be recorded in one of the two different ways. In the first approach (the normal mode) the receive timestamp is recorded when the packet is received by the application, i.e. is the data collection program, on the receive socket. In the second approach (the superuser mode), using the *tcpdump* [2] program, the receive timestamp is recorded as soon as the packet is received at the kernel from the network interface hardware. The *tcpdump* program communicates the receive timestamp together with the data packet to the data collection program. This second approach of obtaining receive timestamps is useful when we want to measure network delay without considering the protocol processing and other delays at the hosts. The problem with this approach is that the use of *tcpdump* requires superuser privileges at the receiving hosts and hence is likely to be used sparingly.

It is worth pointing out here that even though *mbat* has been primarily developed for recording multicast sessions, it could be used to record unicast sessions too.

Another important feature of *mbat* is that it allows (with the help of *ntpdate* program) the end receivers to collect timestamps from NTP servers. The local time at which the NTP server time is obtained is also recorded. These timestamps could be used to facilitate the study of the delay behavior even in the presence of clock offsets and clock drifts. It is important to note that the use of *ntpdate* in *mbat* does not in any way affect the end receiver clocks.

Once the necessary information has been recorded at several end hosts it could be collected at a desired node using a single **ftpdata** command from the control node. The collected data is in binary form and could be pre–processed using the **preproc** program to obtain all the required information in an ASCII format in appropriate files. These files could then be used for analysis.

mbat could collect data from multiple sessions of same application or different applications running at the same time. It uses the IP address and port to distinguish one session from another.

The important features of *mbat* are summarized as follows:

- Captures multicast/unicast packets on the MBone/Internet/Intranet.
- Records information necessary to study loss and delay behavior of the network and also to study the source characteristics of applications. The recorded information includes the identity of the sender, length and sequence number of the application packets, send and receive timestamps.
- Supports a *post facto* synchronization of receiver clocks with the help of NTP timestamps.
- Supports simultaneous collection of data from multiple sessions of the same application or different applications.

3 User Commands

We now describe the commands that are available for data collection using *mbat*. There are different procedures to be followed depending on whether a host is a control node or a receiver. It should be noted that there can be only one control node in the system[1]. However, a single host can be both a control node (controller) and a receiver at the same time by running the data collection component two times with different parameters.

NOTE: For the purpose of reliably multicasting the control commands a file named *recvr-file* containing the IP addresses of all the end host receivers has to be created. This file is required at the control node if control commands are multicast to a groups of receivers and should be created and/or edited to add or remove the IP addresses of end-receivers. One IP address should be written per line in *recvr-file*. If control commands are unicast to a single receiver using a unicast IP address then this file is not required.

- Running the data collection component:

  The following command has to be executed at the shell prompt on the control node and all the receivers. At the control node it is started with the *-c* argument and at the receivers it is started with the *-r* argument.

  mbat -c|-r [-m aaa.bbb.ccc.ddd][-p port][-t ttl][-T][-N][-F|-S aaa.bbb.ccc.ddd]

-r	:	Act like a receiver
-c	:	Act like a controller
-m	:	Use the given multicast address
-p	:	Use the given multicast port
-t	:	Use the given time to live
-F	:	Use the given NTP server address only
-S	:	Use the given NTP server address
-T	:	Record data using tcpdump
-N	:	Record NTP timestamps

  The multicast address and port are used for multicasting commands from the control node to the receivers. One could use a unicast address instead of a multicast address for sending control commands if there is only one receiver.
  Examples:
  $ mbat -c -m 224.2.138.39 -p 44465
  $ mbat -r -m 224.2.138.38 -p 44465 -T
  $ mbat -r -m 128.119.40.58 -p 44430
  $ mbat -r -T
  $ mbat -r -N

[1] *mbat's* controller is stateless. This implies that the controller can be potentially started at a different node after each command, if there is a problem at the node where the controller was started previously.

The default values of the address, port number and time to live are 224.2.138.38, 44415 and 1 respectively. The default mode of data collection is the normal mode (i.e., *non-tcpdump* data collection). NTP timestamps are not collected by default and one should use -*N* to let a receiver start collecting them. The default NTP server's IP address is 128.96.60.5. The -*S* option is used to specify an NTP server IP address.

The following commands are to be used by the control node for controlling the data collection at the end host receivers. These commands are entered at the *Command>* prompt of the controller program. These commands are unicast/multicast to the receiver(s).

1. Start recording a multicast application session:

start addr port id type

addr	:	IP address ('dot' numerical format)
port	:	Session port
id	:	Experiment id
type	:	Experiment type (vat|vic|rtpvat[2]|nv)

The *address* and *port* number are the ones used by the multicast application to send out data. A receiver on receiving this command starts recording the necessary information from the data received on this address and port. In addition, a receiver also starts listening to session data that is received on the same multicast address and a port number [3] equal to *port* + 1.

Examples:

Command> start 224.2.0.1 23456 0 rtpvat

Command> start 128.119.40.229 43434 0 rtpvat

When the *start* command is used to record data sent to a unicast address two scenarios are possible. If an end receiver is "listening" to the data on one process in addition to recording the data on another process (for example if we wish to record *vat* data unicast from a sender to a receiver and if the receiver is also using the *vat* application to listen to the voice data in addition to recording it) then *mbat* has to be started with the -*T* option (*tcpdump* mode). This is because the two processes, with different sockets, cannot bind to the same port. When *tcpdump* is used for recording we do not need to open a socket and bind the socket to the destination port. However, if the end receiver is only interested in recording and not "listening" to the data then there is no necessity to start *mbat* with the -*T* option.

2. Stop recording a multicast application session:

[2] *rtpvat* stands for the newer version of *vat* that uses the RTP protocol.

[3] *mbat* assumes that session data is always sent on a port number that is one more than the port number on which the actual application data is sent.

stop addr port

addr	:	IP address ('dot' numerical format)
port	:	Session port

The *address* and *port* number are the ones used by the multicast application to send out data. A receiver on receiving this command stops recording the multicast application. In addition, a receiver also stops recording any session data that is received on the same multicast address and a port number equal to $port + 1$.
Example:
Command> stop 224.2.0.1 23456

3. Gathering the collected data at a desired host:

ftpdata addr port site dir [user] [password]

addr	:	IP address ('dot' numerical format)
port	:	Session port
site	:	Ftp site location
dir	:	Directory path
user	:	User name - defaults to anonymous
password	:	User password - defaults to root@

The *ftpdata* command is used to inform the end-receivers to transfer the files in which the multicast application data and session data corresponding to *addr* and *port* has been recorded. The files are transferred to the host *site* in the directory *dir*. The default user name is "anonymous" and the default password is "root@site." Please note that the password is not encrypted and also it does get printed on the control node's monitor.
Example:
Command> ftpdata 224.2.0.1 23456 gaia.cs.umass.edu /usr/mmedia/kasera

4. Gathering the NTP data at a desired host:

ftpsyncfile addr port site dir [user] [password]

site	:	Ftp site location
dir	:	Directory path
user	:	User name - defaults to anonymous
password	:	User password - defaults to root@

This command is same as *ftpdata* except that it is used for transferring the files containing the NTP timestamps collected at the end-receivers and not for application or session data files.
Example:
Command> ftpsyncfile gaia.cs.umass.edu /usr/mmedia/kasera

5. Help on a command:

   help cmd
   cmd : command name

   This command could be used to obtain help on any of the above commands.
   Example:
   Command> help start

6. Quitting the control program:

   quit

The difference between the *exit* and *quit* commands is that *exit* is used for sending a message to the end receiver(s) to kill the data collection processes and *quit* is used for locally quitting the control program (usually after *exit*).

Using the above commands data is collected in files whose names are constructed as follows. The first eight characters of the filename are the first eight characters of the receiver's hostname. This is followed by the string describing the type of application that is being recorded (i.e., *vic*|*vat*|*rtpvat*|*nv*) and the data collection mode (1 if *tcpdump* is used and 0 otherwise). This is followed by a string that contains the date and time at which the data recording was started. Finally the application port number is concatenated at the end. For example for *rtpvat* data collected on host *gonzo* using *tcpdump* on 5^{th} December 1996 at 17:05 hours GMT on port 53500 the filename is *gonzo-rtpvat*1-05*Dec*96-17:05.73*d*4[4].

The collected data is in the binary form. To convert it into a formatted ASCII form for easy manipulation we have developed a simple pre–processing program. The following command has to be used to pre–process the data.

- Pre-processing collected data:

  preproc -i inputfile [**-o outputfile**]
  -i : Input Filename
  -o : Output Filename

  Example:
  $ preproc -i sahir.cs-vic0-29Oct96-18:08.8628 -o out1

The input and output filenames should specify the entire absolute path if the default directory is not the current directory. The first line of the formatted ASCII output file contains the application type, the mode of data collection (Mode is 1 if *tcpdump* is used otherwise Mode is 0), the type of data (whether application data or session data), the application's multicast IP address, the application port number and the receiver's IP address in this order.

[4] The extension 73*d*4 is nothing but the number 53500 expressed in hexadecimal.

Each subsequent line contains per packet information such as the IP address of the sender, time in seconds and microseconds (since midnight if *tcpdump* is used and since midnight January 1, 1970 otherwise) when the packet was received, length of the packet, sequence number of the packet, source id (application specific), source timestamp and flags in the packet header in that order. Some of these fields might be absent for some applications or session data. Each missing field is assigned a value zero.

4 Data Collection Examples

In this section we present some examples to illustrate how our *mbat* could be used for data collection.

Figure 2 exhibits the commands used during the recording of data multicast from *vat* and *vic* sources at three receivers. Figure 3 exhibits the commands used during the recording of data unicast from a*vat* source at a receiver (with IP address 128.119.40.58). Here the only process interested in the *vat* data is the recording process. Figure 4 exhibits the commands used during the recording of data unicast from a*vat* source at a receiver (with IP address 128.119.40.58). Here the *vat* data is being received by the *vat* application running at the receiver. Hence data is recorded using *tcpdump*. For this reason *mbat* has been started with the -*T* option at the receiver.

5 Installation

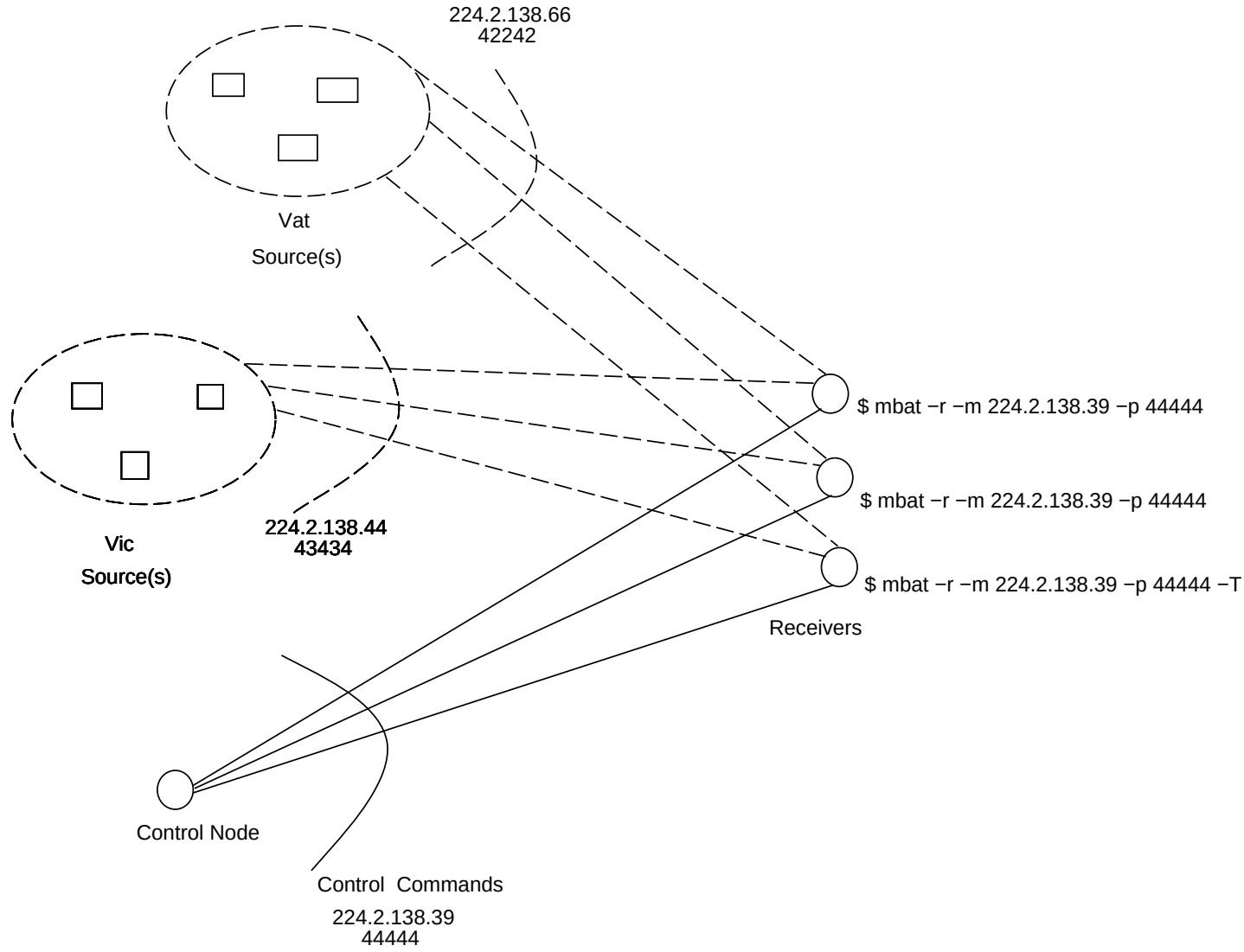
The source and the binary files of *mbat* could be obtained by anonymous ftp to the host *gaia.cs.umass.edu* from the directory pub/mist. Currently, the binaries are available for SunOS 4.X, SunOS 5.X, OSF, Irix6.X and Linux 2.X platforms. The following steps need to be taken for installing *mbat* binaries.

1. Download the proper release for your platform

2. Untar and uncompress the release by running the following command:
   $ zcat mbat.Sun5-4.tar.Z|tar xvf -
   This will create a sub-directory called *mist*. A subdirectory called bin is created within mist. The bin directory contains the binaries *mbat* and *preproc*.

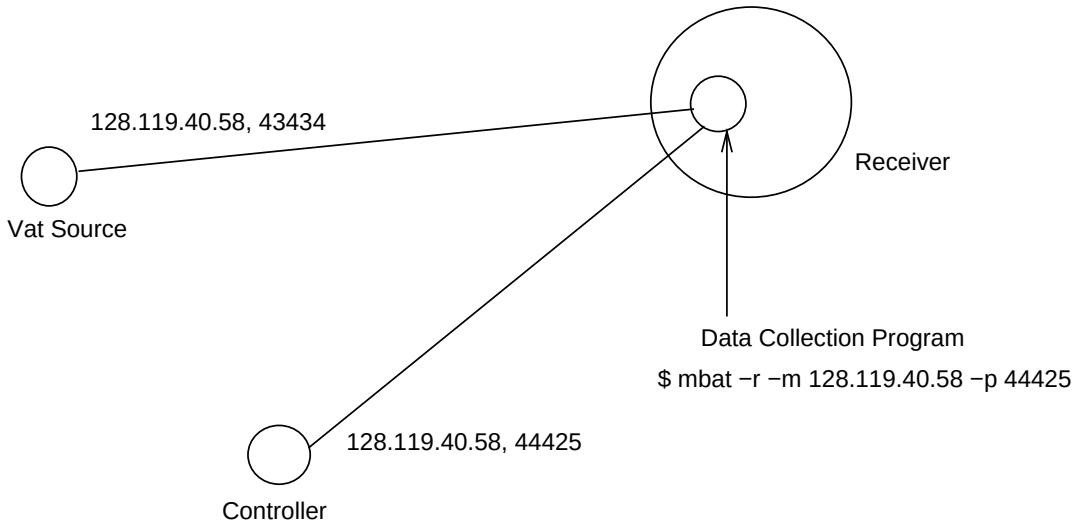
3. Set the proper environment variable by running the following commands (assuming that you have untarred the release in /usr/local/):
   For csh or tcsh users:

   - **$ setenv MIST_HOME /usr/local/mist**
   - **$ setenv PATH ${PATH}:${MIST_HOME}/bin**



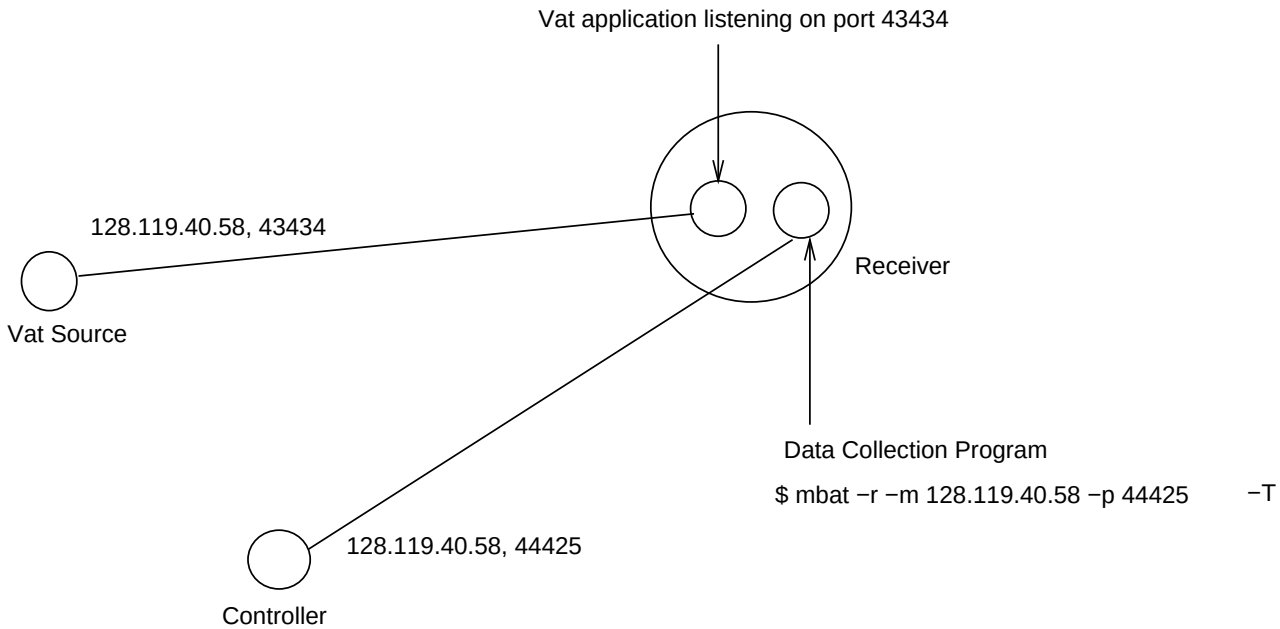

```
$ mbat −c −m 224.2.138.39 −p 44444
Command> start 224.2.138.44 43434 0 vic
Command> start 224.2.138.66 42242 0 vat
Command> stop 224.2.138.66 42242
Command> stop 224.2.138.44 43434
                    .
```

Figure 2: Recording of Multicast Data at Multiple Receivers




```
$ mbat −c −m 128.119.40.58 −p 44425
Command> start 128.119.40.58 43434  0 vat
Command> stop 128.119.40.58 43434
```

Figure 3: Recording of Unicast Data without any "listener"




```
$ mbat −c −m 128.119.40.58 −p 44425
Command> start 128.119.40.58 43434  0 vat
Command> stop 128.119.40.58 43434
```

Figure 4: Recording of Unicast Data in the presence of a "listener"

For ksh users:

- **$ export MIST_HOME=/usr/local/mist**
- **$ export PATH=${PATH}:${MIST_HOME}/bin**

Once the installation is complete the commands described in section 3 could be used to collect and pre–process data. If control commands will be multicast to a group of receivers then one should not forget to create a file called "recvr-file," at the control node, and add IP addresses of each end-receiver, one IP address per line.

The source files (in $C++$) are available under the name mbat.src.tar.Z. They can be installed in the same way as above but now the bin directory does not contain any binary files. In addition to bin, subdirectories dc and preproc are created inside mist. In the dc (data collection) and preproc (pre-processing) directories, all the source files as well as sample *Makefiles* for different platforms are present. Some of these *Makefiles* might require minor changes depending on the local configuration of the platform in which the source files are to be compiled.

For the purpose of using *tcpdump* and *ntpdate* programs from within our tool one could obtain them from http://www-nrg.ee.lbl.gov and http://www.eecis.udel.edu/ñtp respectively. The binaries of these programs should be made available in the bin directory that contains *mbat* executable or the $PATH$ variable should be set to include the path to the directory containing the binaries.

6 Enhancements and Modifications

Currently *mbat* is not available on Windows platform. Hence one important future work would be to port *mbat* on Windows. We also plan to build a GUI for *mbat*. In the current *mbat* we have implemented a simple positive acknowledgment based reliable multicast protocol for transmission of control commands. One could try different protocols if the number of end-receivers becomes very large. Our code is written in a modular fashion so as to allow an easy replacement of the existing multicast protocol. Also, we could easily plug–in new applications in our code.

References

[1] H. Schulzrinne, S. Casner, R. Fredrick and V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications*. Internet draft, November 1995.

[2] S. McCanne, C. Leres and V. Jacobson, *tcpdump*. Lawrence Berkeley Laboratory.

[3] S. McCanne and V. Jacobson, *vic: A Flexible Framework for Packet Video*. In Proceedings of ACM Multimedia Conference, 1995.