Mtools: a one-way-delay and round-trip-time meter

S. AVALLONE[+], M. ESPOSITO[+*], A. PESCAPÈ[+*], S. P. ROMANO[+], G. VENTRE[+*]

[*]ITEM - Laboratorio Nazionale CINI per l'Informatica e la Telematica Multimediali
Via Diocleziano, 328
80125 – Napoli
ITALY

[+]DIS, Dipartimento di Informatica e Sistemistica
Università degli Studi Federico II di Napoli
Via Claudio, 21
80125 – Napoli
ITALY

Abstract: - In this work we present a collection of tools (named Mtools) for analyzing network performance through the measurement of the one-way-delay and the round-trip-time of packets that traverse the network. Mtools has many useful features that will be shown in the following. Mtools is currently available at *http://www.grid.unina.it/grid/mtools*.

Key-Words: - Measurement Environment, Traffic Generator and Simulation Techniques Tools, One-way-delay, Round-trip-time, Packet loss.

1 Introduction

Very often it is important to test the network behaviour when it is subjected to different loads, e.g. for measuring its capacity of serving incoming traffic. We also might want to verify the accuracy of an analytic model which estimates throughput and packet transmission time making some assumptions on the traffic source. This raises the need to simulate that particular traffic source. Moreover, we may want to compare different configurations of the same network, e.g. to compute the benefits introduced by strategies such as DiffServ, MPLS, DiffServ over MPLS, etc. In such situation it is useful to repeat many times the same realization of the packet generation (random) process. As we can see below, in all of these cases Mtools can help us.

Mtools currently runs under Linux or FreeBSD operating systems (we encourage everyone to port Mtools on other platforms). It is made up of two instruments:

- One-way-delay Meter (OWDM)
- Round-trip-time Meter (RTTM)

each of which is constituted by two utilities, a sender and a receiver. OWDM and RTTM have many common features, which we will describe below, before delving into the peculiarities of each one in the following sections. The transport protocol used by senders is UDP. It is possible to model both packets inter-departure and packets size as a random process. Currently, three choices have been implemented: constant, uniformly distributed and exponentially distributed. However, thanks to the Robert Davies' random number generator library (included in our distribution), it is very simple to add new random distributions, so as to simulate very complex traffic sources. A useful feature of Mtools is the possibility of specifying the seed value for the packets inter-departure and packets size random processes; in this way, it is possible to repeat exactly a particular realization of these random processes. This feature, as we said above, is interesting in several circumstances. If the seed is not specified, its value is random, that is it is a function of the time of day.

Moreover senders can be used both in command line mode (generating only one flow) and in script mode (generating multiple flows). The usage in these modes is described in detail in the following sections.

To collect statistics it is necessary to store some information in the sent packets. UDP payload of sent packets, in fact, contains the number of the flow the packet belongs to, a sequence number and the time it was sent (expressed in seconds and microseconds since midnight, January 1, 1970). The remaining bytes of the payload are filled with random numbers.

Both senders and receivers can create a log file containing the information stored in transmitted packets. The log file format is the same as that used by MGEN (available at *http://manimac.itd.nrl.navy.mil/MGEN*), so it is possible to use its powerful utilities to analyze log files and obtain statistics. Among these utilities, we mention here *mcalc* (multi-calculator), which outputs statistics (packet and data rate, number of dropped packets, average transmission delay, jitter) on a per-flow and summary basis, and *ez*, which provides delay and rate plots.

In the following sections we describe in detail the syntax of OWDM and RTTM, both in command line mode and in script mode.

2 One-way-delay Meter

One-way-delay Meter (OWDM) enables to send UDP packets to a specific host (at a given port) and measure the transmission time of each packet. As we have already said, OWDM is made up of a sender and a receiver. Sender can be used both in command line mode and in script mode; in the last case, the syntax is:

owdmSend <script_file> [<log_file>]

where script_file is the file that contains the specification of the flows to be generated (a flow for each line, except blank lines and lines beginning with '#'). Each flow is assigned a flow identifier equal to the number of the line of the script file in which it is specified. The syntax of each line is the same as that used in command line mode (described later on). If log_file is specified, OWDM sender records packets transmission times in a file named log_file. This file can be then analyzed by MGEN's utilities to produce, for example, the bit rate graph of generated traffic (in section 4 we will show some examples). This is a very useful tool, since it allows us to retrieve information not only about received traffic but also about transmitted traffic.

In command line mode you can generate only one flow, whose properties are defined by the list of options you type. It is possible to specify, besides destination IP address and port, DS field of packets, probability distribution of packets inter-departure and packets size processes, seed value, generation duration, initial delay and the name of the file in which OWDM sender records packets transmission times (this option, -l, is available only in command line mode). Available options are listed and explained in the online project page. If no options are specified, an help screen will be printed.

OWDM receiver (owdmRecv) listens for and records packets arrived at the ports specified with the –p option. It can be used only in command line mode and has few options. Note that OWDM receiver always produces a log file and, if it is invoked without options, it will monitor port 8999 and write log info in "/tmp/owdmRecv.log".

3 Round-trip-time Meter

Round-trip-time Meter (RTTM) enables to transmit UDP packets to a specific host (at a given port), which sends them back to the sender. It is possible to log both packets arrived at the receiver and packets returned at the sender. RTTM is made up of a sender and a receiver, which differ slightly in the usage from those of OWDM. RTTM sender can be used both in command line mode and in script mode; in the last case, the syntax is:

rttmSend <script_file> [<log_file>]

where script_file is the file that contains the specifications of the flows to be generated (same format as an OWDM script). Round-trip-time of returned packets is stored in a log file, whose name is log_file, if it is specified, or the default value "/tmp/rttmSend.log". In command line mode you can generate only one flow, whose properties are defined by the list of options you specify. Available options are the same as those of OWDM sender; the only difference regards the –l option (valid only in command line mode): if it is not specified, the default value "/tmp/rttmSend.log" is used, otherwise the log file name to be used must be specified. RTTM receiver (rttmRecv) listens for packets arrived at specified ports and sends them back to the sender. It optionally logs received packets, so it is possible to get also one-way delay. RTTM receiver can be used only in command line mode and its options are the same as those of OWDM receiver. The only difference regards the –l option: if it is not present packets arrived at the receiver are not logged, otherwise they are logged in the specified file or, if no file name is specified, in the default file "/tmp/rttmRecv.log".

4 Software architecture

We begin by describing the software architecture related to OWDM; with regard to RTTM, we will stress only the differences. Both OWDM and RTTM are written in C++ language. As we said above, OWDM sender can be used in script mode by providing a script file which contains the specification of the flows to be generated in the same format as that would be used in command line mode. Multiple flows are handled in this way: the flow specified in the first line is handled by the parent process; for each other line a child process is created to handle the corresponding flow. This allows us to write the remaining code as we had only one flow, and this obviously makes data structures and code simpler. Then each process parses its own line and recognized tokens are analyzed by the option parser, which checks for their correctness and sets the appropriate variables. On success, a datagram socket is opened; then, if necessary, the initial delay is awaited and the socket option for setting DS field is set. After calculating the ending time of the packet generation process, a *do* cycle is repeated until this time is reached. Two operations are performed in a cycle: the transmission of a packet and the wait for the inter-departure time.The first operation is constituted by the generation of a random value for the packet size (using the Next method of the class that implements a random variable in the Newran random number generator library), the storage in the packet of the information used to collect statistics, the effective transmission of the packet and, if a log is required, the storage of the above information in a file. We note that the last operation can cause problems when multiple processes try to write into the file at the same time; this problem has been avoided by making the log file a line buffered stream (that is, buffered output is flushed

when newline is encountered) and writing into the file by means of a unique fprintf call for each packet (the printed string ends with newline). Inter-departure time (obtained from a Next method call) is awaited by means of a *select* function call; however, since Linux and FreeBSD support of real-time applications is not very efficient (due to their scheduling mechanism and the inevitable timer granularity), it was necessary to use a trick. That is, a variable records the time elapsed since the last packet was sent; when the inter-departure time must be awaited, this variable is updated. If its value is less than inter-departure time the remaining time is awaited, otherwise the inter-departure time is subtracted from the value of this variable and no time is awaited. This trick guarantees the required bit rate, as shown by the example of the next section.

OWDM receiver is simpler and does not create any child process. It opens a datagram socket for each port on which it will listen for packets, then it enters in an infinite cycle (*while(1)*) in which a *select* function call blocks the program until input is ready on a socket descriptor (a packet is arrived) or on the standard input. When a packet arrives, the information it carries is written into the log file; when Enter key is pressed, the program terminates.

The architecture of RTTM sender is very similar to that of OWDM sender; the only difference regards the *do* cycle, which is replaced by an infinite *while* cycle. Inside this cycle, a boolean variable is used to indicate that the end of the packet generation process has been reached. This time, besides the transmission of a packet and the wait for the inter-departure time (performed until the above boolean variable is false) the cycle contains a non-blocking check for arrived packets, realized with a *select* function call with a null timer. When a packet arrives, the information it carries is written into the log file; when Enter key is pressed, the process terminates.

RTTM receiver is even closer to OWDM receiver; the unique difference is in the case a packet arrives: besides to write log information, it sends the packet back to the sender.

5 Usage example

We now show an example of how to use our tools and MGEN's utilities to analyze log files. We used OWDM and the following script:

-a helios –p 5000 –C 100 –c 958 –t 2000
-a helios –p 5001 –E 200 –c 958 –d 500 –t 2000

This script tells the sender to generate two flows addressed to the host named helios, one at port 5000 and the other at port 5001. The payload size of all packets is 958, that is,

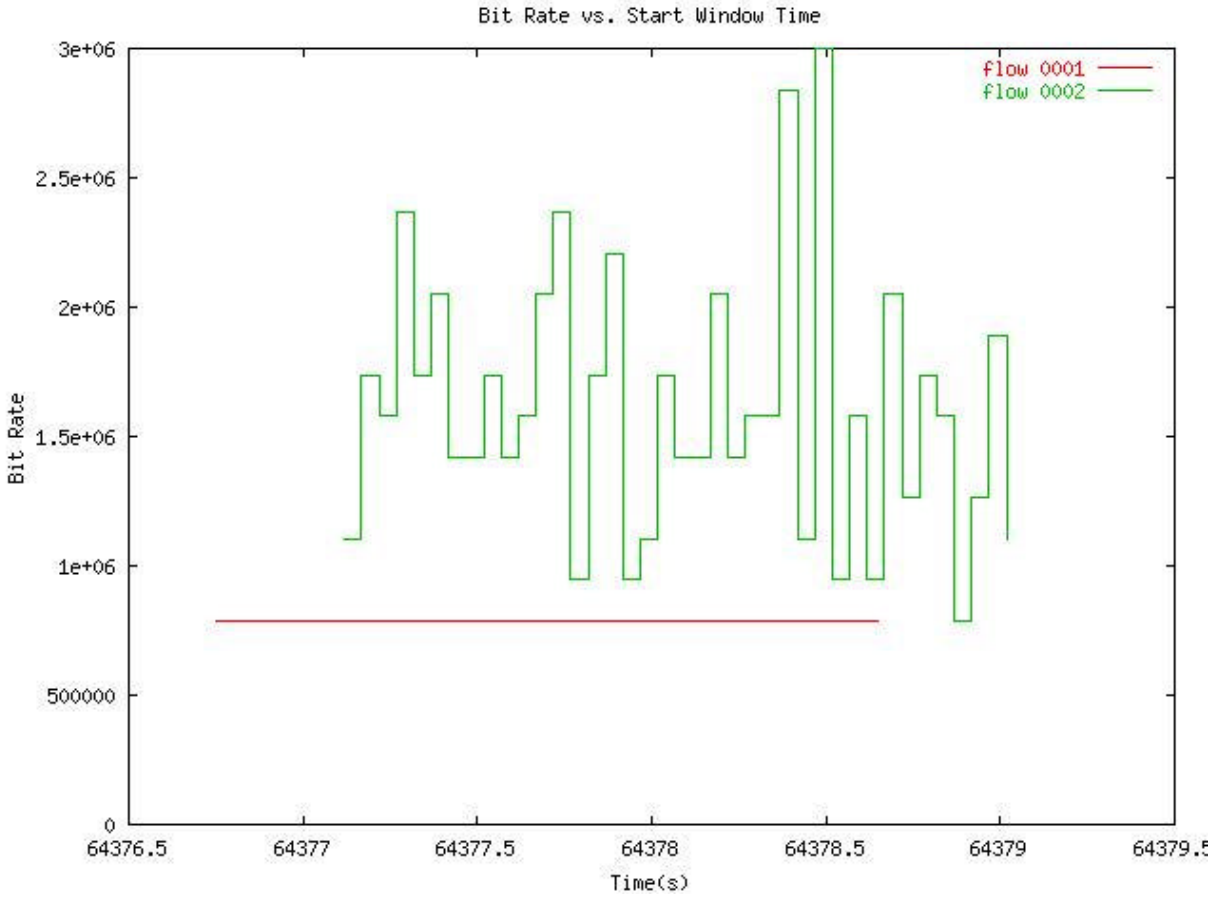

Figure 1: bit rate plot

considering Ethernet, IP and UDP headers, their full size is 1000 bytes. The first flow lasts 2 seconds and packet generation process is a constant process, characterized by the transmission of 100 packets per second (this means a bit rate of 0.8 Mbps); the second flow starts after half a second and lasts two seconds; packet generation process is a gaussian process, characterized by the transmission of 200 packets per second on average (this means a bit rate of 1.6 Mbps). OWDM sender logs sent packet in a file that we process using *ez*, in order to obtain the bit rate graph of generated flows (we set the temporal window on which the bit rate is calculated equal to 50 milliseconds). The resulting plot is shown in Figure 1. As we can see, the required specifications have been met.

6 Conclusion

The idea of creating a new traffic generator arose from the lacks of existing ones (MGEN, Rude/Crude, etc.), emerged when we used them to analyze the different behaviour of the network when some strategies that provide QoS are implemented. The features we needed were the possibility of simulating more complex traffic sources, repeating many times exactly the same traffic pattern (not only its mean values) and getting information not only about received packets but also about transmitted packets. So we decided to build a traffic generator which satisfies these requirements and also includes a tool for measuring the round trip time. Mtools is easily extensible to support other features; as new needs arise, we are ready to add them to our program. At the moment, anyway, we believe that Mtools owns such features that it proves useful in several circumstances (e.g. those mentioned in section 1).

References:

[1] Almes, G., et al. “A One-way Delay Metric for IPPM”, RFC 2679, 1999.

[2] Almes, G., et al. “A Round-trip Delay Metric for IPPM”, RFC 2681, 1999.

[3] Naval Research Laboratory (NRL), The MGEN Toolset on http://manimac.itd.nrl.navy.mil/MGEN/

[4] Davies, R., Newran02A – a random number generator library on http://webnz.com/robert/nr02doc.htm