D-ITG V. 2.6.1d Manual*

Stefano Avallone, Alessio Botta,
Alberto Dainotti, Walter de Donato, and Antonio Pescapé
University of Naples Federico II

May 2, 2008

*home page: http://www.grid.unina.it/software/ITG

Contents

1 ITGSend

1.1 Synopsis

In case of using a script file to generate multiple flows, type:

ITGSend ⟨script_file⟩ [**–l** [⟨logfile⟩]] [**–L** [⟨log_server_addr⟩] [⟨protocol_type⟩]] [**–x** [⟨receiver_logfile⟩]] [**–X** [⟨log_server_addr⟩] [⟨protocol_type⟩]]

If you want to remotely control the sender, launch it in daemon mode:

ITGSend –Q [**–l** [⟨logfile⟩]] [**–L** [⟨log_server_addr⟩] [⟨protocol_type⟩]] [**–x** [⟨receiver_logfile⟩]] [**–X** [⟨log_server_addr⟩] [⟨protocol_type⟩]]

Otherwise if you want to generate a single flow:

ITGSend [**–m** ⟨msr_type⟩] [**–a** ⟨destination_address⟩] [**–rp** ⟨destination_port⟩] [**–sp** ⟨source_port⟩] [**–T** ⟨protocol_type⟩] [**–f** ⟨TTL⟩] [**–b** ⟨DS byte⟩] [**–rk** ⟨receiver_serial_iface⟩] [**–sk** ⟨sender_serial_iface⟩] [**–D**] [**–P**] [**–s** ⟨seed⟩] [**–t** ⟨duration⟩] [**–d** ⟨gen_delay⟩] [**–p** ⟨payload_log_type⟩] [**–j** ⟨enable_idt_recovery⟩] [**–l** [⟨logfile⟩]] [**–L** [⟨log_server_addr⟩] [⟨protocol_type⟩]] [**–x** [⟨receiver_logfile⟩]] [**–X** [⟨log_server_addr⟩] [⟨protocol_type⟩]] [[**–C** ⟨pkts_per_s⟩ | **–U** ⟨min_pkts_per_s⟩⟨max_pkts_per_s⟩ | **–E** ⟨average_pkts_per_s⟩ | **–V** ⟨shape⟩⟨scale⟩ | **–Y** ⟨shape⟩⟨scale⟩ | **–N** ⟨mean⟩⟨std_dev⟩ | **–O** ⟨average_pkts_per_s⟩ | **–G** ⟨shape⟩⟨scale⟩] [**–c** ⟨pkt_size⟩ | **–u** ⟨min_pkt_size⟩⟨max_pkt_size⟩ | **–e** ⟨average_pkt_size⟩ | **–v** ⟨shape⟩⟨scale⟩ | **–y** ⟨shape⟩⟨scale⟩ | **–n** ⟨mean⟩⟨std_dev⟩ | **–o** ⟨average_pkt_size⟩ | **–g** ⟨shape⟩⟨scale⟩]] | [**Telnet** | **DNS** | **CSa** | **CSi** | **Quake3** | **VoIP** [**–x** ⟨codec_type⟩] [**–h** ⟨protocol_type⟩] [**–VAD**]]

NOTE: launching ITGSend in background requires to redirect stdin to /dev/null

1.2 Description

Sender Component of the D-ITG Platform.
The script mode enables ITGSend to simultaneously generate several flows. Each flow is managed by a single thread, with a separate thread acting as a master and coordinating the other threads. To generate n flows, the script file has to contain n lines, each of which is used to specify the characteristics of one flow. Each line can contain all the options illustrated in Section 1.3, but those regarding the logging process (–l, –L, –X, –x). Such options can be specified at the command line and refer to all the flows.

1.3 Options

Flow options:

–m ⟨msr_type⟩	Set the type of meter. Two values are allowed: owdm (one way delay meter) and rttm (round trip time meter). Default is owdm. D-ITG does not provide any sort of synchronization among senders and receivers. In order to correctly measure packet One Way Delay (OWD), the clocks of sender and receiver must be synchronized by other means. Otherwise, we suggest to use the Round Trip Time (RTT) meter;
–a ⟨destination_address⟩	Set the destination address. Default is localhost;
–rp ⟨destination_port⟩	Set the destination port. Default is 8999;
–sp ⟨source_port⟩	Set the source port. If this option is not specified, the source port is set by the operating system;
–T ⟨protocol_type⟩	Set the protocol type. Valid values are UDP, TCP, ICMP, SCTP, DCCP. Default is UDP. If you choose ICMP you must specify the type of message. Root privileges are needed under Linux;
–f ⟨TTL⟩	Set the time to live (TTL). The value is interpreted as a decimal number, or as an hexadecimal number if the prefix "0x" is used. The range is [0, 255];
–b ⟨DS byte⟩	Set the DS byte for QoS tests. The value is interpreted as a decimal number, or as an hexadecimal number if the prefix "0x" is used. The range is [0, 255]. Default is 0 (Note: this option is disabled under Windows 2000 and XP, according to the "Microsoft Knowledge Base Article - 248611" *http://support.microsoft.com/Default.aspx?scid=kb;EN-US;q248611*; under Linux, root privileges are needed to set the DS byte to a value greater than 160);
–rk ⟨receiver_serial_iface⟩	Instructs the receiver to raise a signal on the specified serial interface every time a packet is received. Typical values are COM1, COM2, etc. under Windows and ttys0, ttys1, etc. under Linux;
–sk ⟨sender_serial_iface⟩	Raises a signal on the specified serial interface every time a packet is sent;
–s ⟨seed⟩	Set the seed for the random number generator. By default a random value is taken;
–D	Disable Nagle Algorithm;
–P	Enable high thread priority (only Windows platform);
–t ⟨duration⟩	Set the generation duration. It is expressed in milliseconds. Default is 10000 ms;
–d ⟨gen_delay⟩	Set the generation delay. It is expressed in milliseconds. Default is 0.

–p ⟨payload_log_type⟩	Set the type of information sent in the payload of each packet. Valid values are: 0 no information is sent in the payload packet; 1 only the sequence numbers are sent in the payload packet; 2 standard informations are sent in the payload packet. Default value is 2.
–j ⟨enable_idt_recovery⟩	Enable (1) or disable (0) the strategy used to guarantee the mean bitrate. Default value is 0.

Log options:

–l [⟨logfile⟩]	Generate the log file. If the meter type is OWDM and this option is omitted, ITGSend does not generate the log file. If the meter type is RTTM and this option is omitted, ITGSend generates a log file with the default log file name. The default log file name is /tmp/ITGSend.log under Linux and ITGSend.log under Windows;
–x [⟨receiver_logfile⟩]	Generate the log file at the receiver side. The default log file name is /tmp/ITGRecv.log under Linux and ITGRecv.log under Windows;
–L [⟨log_server_addr⟩] [⟨protocol_type⟩]	Remote log file. The first parameter is the log server IP address (default is localhost); the second parameter is the protocol used to rule the communication between the sender and the log server. Valid values are UDP and TCP (default is UDP). The log file name is specified by the –l option;
–X [⟨log_server_addr⟩] [⟨protocol_type⟩]	This option enables ITGRecv to remotely configure a log server. The first parameter is the log server IP address (default is localhost); the second parameter is the protocol used to rule the communication between the receiver and the log server. Valid values are UDP and TCP (default is UDP). The log file name is specified by the –x option;

Inter-departure time options:

–C ⟨pkts_per_s⟩	Constant inter-departure time (IDT)
–U ⟨min_pkts_per_s⟩ ⟨max_pkts_per_s⟩	Uniformly distributed IDT
–E ⟨average_pkts_per_s⟩	Exponentially distributed IDT
–V ⟨shape⟩ ⟨scale⟩	Pareto distributed IDT
–Y ⟨shape⟩ ⟨scale⟩	Cauchy distributed IDT
–N ⟨mean⟩ ⟨std_dev⟩	Normal distributed IDT
–O ⟨average_pkts_per_s⟩	Poisson distributed IDT
–G ⟨shape⟩ ⟨scale⟩	Gamma distributed IDT

NOTE: The IDT random variable provides the inter-departure time expressed in milliseconds. For the sake of simplicity, in case of Constant, Uniform, Exponential and Poisson variables, each parameter, say it x, is specified as a packet rate value. It is then converted to a time interval value ($x \rightarrow \frac{1000}{x}$). If no option

is specified, a constant IDT with 1000 packets per second is assumed.

Packet size options:

–c ⟨pkts_size⟩	Constant payload size.
–u ⟨min_pkts_size⟩ ⟨max_pkts_size⟩	Uniformly distributed payload size
–e ⟨average_pkts_size⟩	Exponentially distributed payload size
–v ⟨shape⟩ ⟨scale⟩	Pareto distributed payload size
–y ⟨shape⟩ ⟨scale⟩	Cauchy distributed payload size
–n ⟨mean⟩ ⟨std_dev⟩	Normal distributed payload size
–o ⟨average_pkts_size⟩	Poisson distributed payload size
–g ⟨shape⟩ ⟨scale⟩	Gamma distributed payload size

NOTE: If no option is specified, a constant payload size of 512 bytes is assumed.

Transport Layer protocols:

TCP Generates traffic using Transmission Control Protocol.

UDP Generates traffic using User Datagram Protocol.

DCCP Generates traffic using the Datagram Congestion Control Protocol, a message-oriented protocol like UDP with some new features. DCCP implements not only congestion control and congestion control negotiation, but also reliable connection setup, teardown, and feature negotiation. No options are required.

SCTP Generates traffic using the Stream Control Transmission Protocol. At the moment no special features of this protocol have been implemented yet. Soon multi-streaming can be managed. Two options are required: the first is an identificator of the session whom it belongs to and the second is the max number of streams of the session. NOTE: All streams belonging to the same session have to be specified with the same values for all the two options.

Application Layer protocols:

DNS Generate traffic with DNS traffic characteristics. No option is required. NOTE: DNS traffic generation works with both UDP and TCP transport layer protocols. Different settings will be ignored.

Telnet Generate traffic with Telnet traffic characteristics. No option is required. NOTE: Telnet traffic generation only works with TCP transport layer protocol. Different settings will be ignored.

VoIP Generate traffic with VoIP traffic characteristics. NOTE: VoIP traffic generation only works with UDP transport layer protocol. Different settings will

be ignored. VoIP options are:

–x ⟨codec_type⟩	Set the Codec type. VALUES:
	G.711.1 for G.711 codec with 1 sample per pkt (default)
	G.711.2 for G.711 codec with 2 samples per pkt
	G.723.1 for G.723.1 codec
	G.729.2 for G.729 codec with 2 samples per pkt
	G.729.3 for G.729 codec with 3 samples per pkt
–h ⟨protocol_type⟩	Set the protocol type. VALUES:
	RTP for Real Time Protocol (default)
	CRTP for Real Time Protocol with header compression
–VAD	Set the Voice Activity Detection (it is off by default).

CSa Generate traffic with Counter Strike traffic characteristics related the active phase of the game. No option is required. NOTE: CSa traffic generation only works with UDP Transport Layer protocol. Different settings will be ignored [1].

CSi Generate traffic with Counter Strike traffic characteristics related the inactive phase of the game. No option is required. NOTE: CSi traffic generation only works with UDP Transport Layer protocol. Different settings will be ignored [1].

Quake3 Generate traffic with Quake III Arena traffic characteristics. No option is required. NOTE: Quake traffic generation only works with UDP Transport Layer protocol. Different settings will be ignored [2].

NOTE: If you specify an application layer protocol then you cannot specify any inter-departure time or packet size option. The other options illustrated above are allowed. If you want to specify an application layer protocol you must indicate it after every other option.

2 ITGRecv

2.1 Synopsis

ITGRecv $\left[\right.$ **–l** $\left[\langle\text{logfile}\rangle\right]\left.\right]$ $\left[\right.$ **–L** $\left[\langle\text{log\_server\_addr}\rangle\right]$ $\left[\langle\text{protocol\_type}\rangle\right]\left.\right]$

NOTE: launching ITGRecv in background requires to redirect stdin to /dev/null

2.2 Description

Receiver Component of the D-ITG Platform.
It can simultaneously receive flows from different senders.

2.3 Options

–l [⟨logfile⟩]	Generate the log file. If the –l option is specified at the receiver side and the –x option at the sender side (with different log file names), then the last option is ignored. If the –l option is not specified, each sender may specify a different log file name by means of the –x option;
–L [⟨log_server_addr⟩] [⟨protocol_type⟩]	The first parameter is the log server IP address (default is localhost); the second parameter is the protocol used to rule the communication between the receiver and the log server. Two values are allowed: UDP and TCP (default is UDP). If the –L option is specified at the receiver side and the –X option at the sender side, then the last option is ignored. If the –L option is not specified, each sender may specify a different log server by means of the –X option.

3 ITGLog

3.1 Synopsis

ITGLog

NOTE: launching ITGLog in background requires to redirect stdin to /dev/null

3.2 Description

Log Server of the D-ITG Platform.
ITGLog receives log information from the ITGSend sender and the ITGRecv receiver. ITGLog listens on ports dynamically allocated in the range [9003–10003].

3.3 Options

No option available.

4 ITGDec

4.1 Synopsis

ITGDec [⟨logfile⟩ [**–v** | **–i**] [**–t**] [**–s**] [**–l** ⟨text_log_file⟩] [**–o** ⟨octave_log_file⟩] [**–d** ⟨delay_interval_size⟩] [**–j** ⟨jitter_interval_size⟩] [**–b** ⟨bitrate_interval_size⟩] [**–p** ⟨packetloss_interval_size⟩] [**–f** ⟨max_flow_num⟩] [**–P**] [**–I**]] | [**–h** | **––help**]

4.2 Description

The ITGDec decoder is the utility to analyze the results of the experiments conducted by using the D-ITG generation platform. ITGDec parses the log files generated by ITGSend and ITGRecv and calculates the average values of bitrate, delay and jitter either on the whole duration of the experiment or on variable-sized time intervals. You can analyze the binary log file only on the operating system used to create that file. You can use another operating system if the log file is in text format. The *Total time* of the experiment is calculated as the difference between receiving time of last and first packet.
The displayed *Jitter* is an average value. It is calculated according to Figure 4.2.

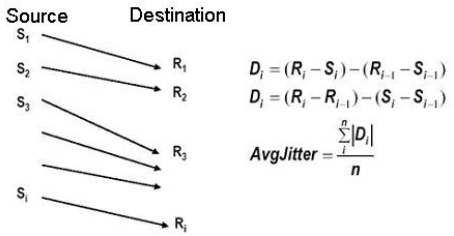

Figure 1: Jitter formula

The displayed *Delay* is calculated as the average of differences between receiving and sending times of packets. D-ITG does not provide any sort of synchronization among senders and receivers. In order to correctly measure packet One Way Delay (OWD) the clocks of sender and receiver must be synchronized by other means. Depending on the requested resolution, NTP or GPS can be used. In the case of synchronization issues we suggest to use the Round Trip Time (RTT) meter.
The displayed *Delay standard deviation* (σ) is calculated according to equation 1:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(d_i - \hat{d}\right)^2} \tag{1}$$

where N is the number of packets considered, d_i is the delay of packet i, and $\hat{d}$ is the average delay of packets.

4.3 Options

–v	Print average and total results on screen in standard visualization format.
–i	Print average and total results on screen in a different format in which each field is separated by "*" charachter.
–t	The log file in input is considered to be in text format. If not specified the log file in input is considered to be in binary format.
–s ⟨personal_string⟩	The log file in input is split in N files where N is the number of flows detected in the log file. The names of the split files are in following format: "sender ip address"-"receiver ip address"-"flow number".⟨personal_string⟩.dat . If not provided, the Default value for ⟨personal_string⟩ is "log".
–l ⟨text_log_file⟩	Produce an output log file in text format with name ⟨text_log_file⟩ .
–o ⟨octave_log_file⟩	Generate a log file named "octave_log_file" that can be imported in Octave/Matlab.
–f ⟨flow_info⟩	If ⟨flow_info⟩ is a number, only the flows with flow number less or equal to ⟨flow_info⟩ will be considered. If ⟨flow_info⟩ = t all packets will be considered as belonging to the same flow.
–d ⟨delay_interval_size⟩	Every ⟨delay_interval_size⟩ milliseconds the average of transmission time of packets is calculated and printed in a file called "delay.dat". The structure of this file is as follows: a first column with time reference is present and is followed by different columns containing results calculated for each flow. A final column with aggregate result is also printed. In the first row "sender ip address"-"receiver ip address"-"flow number" is also printed in the columns related to single flows results. This is the reference for single flows information.

-b ⟨bitrate_interval_size⟩	Every ⟨bitrate_interval_size⟩ milliseconds the average of instantaneous bit rate of packets is calculated and printed in a file called “bitrate.dat”. The structure of this file is as follows: a first column with time reference is present and is followed by different columns containing results calculated for each flow. A final column with aggregate result is also printed. In the first row “sender ip address”-“receiver ip address”-“flow number” is also printed in the columns related to single flows results. This is the reference for single flows information.
-j ⟨jitter_interval_size⟩	Every ⟨jitter_interval_size⟩ milliseconds the average of instantaneous jitter of packets is calculated and printed in a file called “jitter.dat”. The structure of this file is as follows: a first column with time reference is present and is followed by different columns containing results calculated for each flow. A final column with aggregate result is also printed. In the first row “sender ip address”-“receiver ip address”-“flow number” is also printed in the columns related to single flows results. This is the reference for single flows information.
-p ⟨packet_loss_interval_size⟩	Every ⟨packet_loss_interval_size⟩ milliseconds the average packet loss is calculated and printed in a file called “packetloss.dat”. The structure of this file is as follows: a first column with time reference is present and is followed by different columns containing results calculated for each flow. A final column with aggregate result is also printed. In the first row “sender ip address”-“receiver ip address”-“flow number” is also printed in the columns related to single flows results. This is the reference for single flows information.
-P	Print on screen the size of each packet.
-I	Print on screen the inter departure time between each packet.

4.4 Compatibility with previous versions

Version 2.4 of ITGDec is still capable of processing log files of D-ITG previous versions. To have a version of ITGDec that is compatible with 2.3 version (or previous) log files it is necessary to compile the source file (ITGDecod.cpp) by using a pre-compiler option. This option is `V23` and it has to be passed to the compiler with the directive `-D`.
Therefore, to compile ITGDec to support this feature it is necessary to modify the `Makefile` present in the `src` directory and to add the `-DV23` option. Alternatively it is possible to compile ITGDec alone, using the following syntax on the command line: “**g++ ITGDecod.cpp -DV23 -lm -o ITGDec**”.

5 ITGplot

5.1 Synopsis

octave ITGplot ⟨input_file⟩ [⟨flow_set⟩]

5.2 Description

ITGplot is an Octave (*http://www.octave.org*) script that enables to draw plots starting from the data contained in delay.dat, bitrate.dat, jitter.dat or packet-loss.dat (see Section 4.3). The plot is saved (in the EPS format) in a file having the same name as the input file and the .eps extension. It is possible to save the plots in other formats by changing the `graphicformat` string in ITGplot. The available formats are those provided by gnuplot (run gnuplot and type 'set term' to see the list of the available terminal types). It is also possible to give a title to the plot by setting the environment variable DITG_PLOT_TITLE.

5.3 Options

⟨input_file⟩	The file containing the data to be plotted. It may be one of the .dat files produced by ITGDec;
⟨flow_set⟩	The subset of flows to be plotted, expressed in the Octave notation. Thus, 2:4 causes the second, the third and the fourth flows to be plotted, while "[1 3 5]" causes the first, the third and the fifth flows to be plotted (remember: double quotes are needed to enclose values containing blanks). If not specified, all the flows are plotted.

5.3.1 Notes for Linux users

It is possible to make ITGplot an executable Octave program by exploiting the '#!' script mechanism. Follow these steps:

1. make ITGplot an executable file (`chmod +x ITGplot`)
2. locate your Octave interpreter (`which octave`)
3. write such location in the first line of ITGplot (e.g. `#! /usr/bin/octave -qf`)

Then, you can directly execute ITGplot instead of passing it as argument to octave (e.g. `./ITGplot bitrate.dat 1:4`).
If you want to set a title for the plot, you can use the `env` command:

```
env DITG_PLOT_TITLE="A wonderful plot" ITGplot bitrate.dat "[1 4 6]"
```

5.3.2 Notes for Windows users

First, since ITGplot uses gnuplot to draw plots it is necessary to specify the path to the gnuplot executable (pgnuplot.exe), which should be "C:\Program Files\GNU Octave *version*\bin". The path Octave will search for programs to run can be specified in three different ways:

1. using the `--exec-path` command line option:
   `octave --exec-path "C:\Program Files\GNU Octave version\bin" ITGplot bitrate.dat`
2. setting the OCTAVE_EXEC_PATH environment variable:
   `set OCTAVE_EXEC_PATH="C:\Program Files\GNU Octave version\bin"`
3. defining the EXEC_PATH variable in the startup file ("C:\Program Files\GNU Octave *version*\opt\octave\share\octave\site\m\startup\octaverc"):
   EXEC_PATH="C:\\Program Files\\GNU Octave *version*\\bin"

Clearly, if the method described either in 2 or 3 is used there is no need to use the `--exec-path` command line option.
If you want to set a title for the plot, you can type:

```
set DITG_PLOT_TITLE="A wonderful plot"
```

before executing ITGplot.

6 ITGapi

6.1 Description

ITGapi is a C++ API that enables to remotely control traffic generation. For this purpose, after having launched ITGSend in daemon mode (ITGSend –Q) on one or more traffic source nodes, it is possible to use the following function to remotely coordinate the traffic generation from different sender:

```
int DITGsend(char *sender, char *message);
```

sender is the IP address of ITGSend and message is the string you would type at command line (except the name of the ITGSend executable file). Returns 0 in case of success, -1 otherwise. ITGSend, when used in daemon mode, sends messages back to the application that issued the generation of the traffic flow. Two types of messages are used, one to acknowledge the start of the generation process and the other to signal its end. The manager application is able to catch those messages by using the function:

```
int catchManagerMsg(char **senderIP, char **msg);
```

the return value is -1 in case no message arrived (the function is non blocking), 1 to indicate the start of the flow and 2 to indicate the end of the flow; senderIP is a pointer to a string containing the IP address of the sender that sent

the message and msg is a pointer to a string containing the command that the sender received.

These prototypes are declared in ITGapi.h ITGManager.cpp is an example of application to remotely control the generation of traffic. To compile it, compile first ITGapi.cpp:

g++ ITGManager.cpp ITGapi.o -o ITGManager

7 Example

7.1 Example 1

Single UDP flow with costant inter-departure time between packets and costant packets size

```
1.start the receiver on the destination host (say it B):
./ITGRecv
2.start the sender on the source host (say it A):
./ITGSend -a B -sp 9400 -rp 9500 -C 100 -c 500 -t 20000
-x recv_log_file
The resulting flow from A to B has the following
charateristic:
the sender port is 9400
the destination port is 9500
100 packets per second are sent (with constant
inter-departure time between packets)
the size of each packet is equal to 500 bytes
the duration of the generation experiment is 20 seconds
(20000 milliseconds)
at receiver side ITGRecv creates log file recv_log_file
```

7.2 Example 2

Single TCP flow with costant inter-departure time between packets and uniformly distributed packet size between 500 and 1000 bytes with local sender/receiver log

1. `start receiver on the destination host (10.0.0.3)`
   `[donato@catarella tmp]$ ./ITGRecv -l recv_log_file`
2. `start the sender on the source host [donato@otto`
   `donato]$ ./ITGSend -a 10.0.0.3 -rp 9501 -C 1000 -u`
   `500 1000 -l send_log_file`
3. `close the ITGRecv by pressing Ctrl-C`

4. decode the receiver log file on the destination host: [donato@catarella tmp]$ ./ITGDec recv_log_file

```
---------------------------------------------------------
Flow number:  1
From 10.0.0.4:34771
To 10.0.0.3:9501
---------------------------------------------------------
Total time = 10.001837 s
Total packets = 10000
Minimum delay = 3633.445701 s
Maximum delay = 3633.464808 s
Average delay = 3633.449749 s
Average jitter = 0.000706 s
Delay standard deviation = 0.001364 s
Bytes received = 7498028
Average bitrate = 5997.320692 Kbit/s
Average packet rate = 999.816334 pkt/s
Packets dropped = 0 ( 0 %)
---------------------------------------------------------
**************** TOTAL RESULTS ******************
Number of flows = 1
Total time = 10.001837 s
Total packets = 10000
Minimum delay = 3633.445701 s
Maximum delay = 3633.464808 s
Average delay = 3633.449749 s
Average jitter = 0.000706 s
Delay standard deviation = 0.036939 s
Bytes received = 7498028
Average bitrate = 5997.320692 Kbit/s
Average packet rate = 999.816334 pkt/s
Packets dropped = 0 ( 0 %)
Error lines = 0
```

5. decode the sender log file on the source host: [donato@otto donato]$ ./ITGDec send_log_file

```
---------------------------------------------------------
Flow number:  1
From 10.0.0.4:34771
To 10.0.0.3:9501
---------------------------------------------------------
Total time = 9.999002 s
Total packets = 10000
Minimum delay = 0.000000 s
Maximum delay = 0.000000 s
```

```
Average delay = 0.000000 s
Average jitter = 0.000000 s
Delay standard deviation = 0.000000 s
Bytes received = 7498028
Average bitrate = 5999.021102 Kbit/s
Average packet rate = 1000.099810 pkt/s
Packets dropped = 0 ( 0 %)
-------------------------------------------------------
*************** TOTAL RESULTS ******************
Number of flows = 1
Total time = 9.999002 s
Total packets = 10000
Minimum delay = 0.000000 s
Maximum delay = 0.000000 s
Average delay = 0.000000 s
Average jitter = 0.000000 s
Delay standard deviation = 0.000000 s
Bytes received = 7498028
Average bitrate = 5999.021102 Kbit/s
Average packet rate = 1000.099810 pkt/s
Packets dropped = 0 ( 0 %)
Error lines = 0
-------------------------------------------------------
```

7.3 Example 3

Single TCP flow with costant inter-departure time between packets and uniformly distributed packet size between 500 and 1000 bytes with remote sender/receiver log

1. start the log server on the log host:
   `[donato@catarella tmp]$ ITGLog`
2. start the receiver on the destination host:
   `[donato@catarella tmp] ITGRecv`
3. start the sender on the source host: `[donato@otto donato]$ ITGSend -a 10.0.0.3 -rp 9501 -C 1000 -u 500 1000 -l send_log_file -L 10.0.0.3 UDP -X 10.0.0.3 UDP -x recv_log_file`
4. close the receiver by pressing Ctrl-C
5. close the log server by pressing Ctrl-C
6. decode the receiver log file on the log host:
   `[donato@catarella tmp]$ ITGDec recv_log_file`

```
-------------------------------------------------------
```

```
Flow number:  1
From 10.0.0.4:34772
To 10.0.0.3:9501
------------------------------------------------------------
Total time = 9.993970 s
Total packets = 9997
Minimum delay = 3633.432089 s
Maximum delay = 3633.504881 s
Average delay = 3633.436616 s
Average jitter = 0.000795 s
Delay standard deviation = 0.004083 s
Bytes received = 7495843
Average bitrate = 6000.292576 Kbit/s
Average packet rate = 1000.303183 pkt/s
Packets dropped = 2 ( 0 %)
------------------------------------------------------------
**************** TOTAL RESULTS ******************
Number of flows = 1
Total time = 9.993970 s
Total packets = 9997
Minimum delay = 3633.432089 s
Maximum delay = 3633.504881 s
Average delay = 3633.436616 s
Average jitter = 0.000795 s
Delay standard deviation = 0.063898 s
Bytes received = 7495843
Average bitrate = 6000.292576 Kbit/s
Average packet rate = 1000.303183 pkt/s
Packets dropped = 2 ( 0 %)
Error lines = 0
------------------------------------------------------------
```

7. decode the sender log file on the log host:

```
[donato@otto donato]$ ITGDec send_log_file
------------------------------------------------------------
Flow number:  1
From 10.0.0.4:34772
To 10.0.0.3:9501
------------------------------------------------------------
Total time = 9.999001 s
Total packets = 10000
Minimum delay = 0.000000 s
Maximum delay = 0.000000 s
Average delay = 0.000000 s
Average jitter = 0.000000 s
Delay standard deviation = 0.000000 s
```

```
Bytes received = 7498028
Average bitrate = 5999.021702 Kbit/s
Average packet rate = 1000.099910 pkt/s
Packets dropped = 0 ( 0 %)
-------------------------------------------------------------
*************** TOTAL RESULTS *****************
Number of flows = 1
Total time = 9.999001 s
Total packets = 10000
Minimum delay = 0.000000 s
Maximum delay = 0.000000 s
Average delay = 0.000000 s
Average jitter = 0.000000 s
Delay standard deviation = 0.000000 s
Bytes received = 7498028
Average bitrate = 5999.021702 Kbit/s
Average packet rate = 1000.099910 pkt/s
Packets dropped = 0 ( 0%)
Error lines = 0
```

7.4 Example 4

If you want to simultaneously generate more than one flow, you have to prepare a script file like those shown in the following examples. Three UDP flows with different constant bit rate and remote log:

1. start the log server on the log host: [donato@otto tmp]$ ITGLog
2. start the receiver on the destination host: [donato@catarella tmp] ITGRecv
3. create the script file: [donato@otto donato]$ cat script_file
   ```
   -a 10.0.0.3 -rp 10001 -C 1000 -c 512 -T UDP
   -a 10.0.0.3 -rp 10002 -C 2000 -c 512 -T UDP
   -a 10.0.0.3 -rp 10003 -C 3000 -c 512 -T UDP
   ```
4. start the sender: [donato@otto donato]$ ITGSend script_file -l send_log_file -L 10.0.0.4 UDP -X 10.0.0.4 UDP -x recv_log_file
5. close the receiver by pressing Ctrl-C:
6. close the log server by pressing Ctrl-C:
7. decode the receiver log file on the log host: [donato@otto donato]$ ITGDec recv_log_file
   ```
   -------------------------------------------------------------
   Flow number: 3
   ```

```
From 10.0.0.4:34775
To 10.0.0.3:10003
------------------------------------------------------------
Total time = 10.016555 s
Total packets = 6098
Minimum delay = 3633.409810 s
Maximum delay = 3634.259565 s
Average delay = 3633.507249 s
Average jitter = 0.002100 s
Delay standard deviation = 0.156419 s
Bytes received = 3122176
Average bitrate = 2493.612624 Kbit/s
Average packet rate = 608.792145 pkt/s
Packets dropped = 22216 ( 78
------------------------------------------------------------
------------------------------------------------------------
Flow number:  1
From 10.0.0.4:34773
To 10.0.0.3:10001
------------------------------------------------------------
Total time = 9.638360 s
Total packets = 2269
Minimum delay = 3633.402899 s
Maximum delay = 3634.260524 s
Average delay = 3633.461365 s
Average jitter = 0.003114 s
Delay standard deviation = 0.135945 s
Bytes received = 1161728
Average bitrate = 964.253670 Kbit/s
Average packet rate = 235.413494 pkt/s
Packets dropped = 4274 ( 65 %)
------------------------------------------------------------
------------------------------------------------------------
Flow number:  2
From 10.0.0.4:34774
To 10.0.0.3:10002
------------------------------------------------------------
Total time = 10.000351 s
Total packets = 3136
Minimum delay = 3633.407982 s
Maximum delay = 3634.455203 s
Average delay = 3633.514464 s
Average jitter = 0.002740 s
Delay standard deviation = 0.221725 s
Bytes received = 1605632
Average bitrate = 1284.460515 Kbit/s
```

```
Average packet rate = 313.588993 pkt/s
Packets dropped = 16864 ( 84 %)
------------------------------------------------------------
**************** TOTAL RESULTS ******************
Number of flows = 3
Total time = 10.038005 s
Total packets = 11503
Minimum delay = 3633.402899 s
Maximum delay = 3634.455203 s
Average delay = 3633.500165 s
Average jitter = 0.003291 s
Delay standard deviation = 0.417552 s
Bytes received = 5889536
Average bitrate = 4693.790051 Kbit/s
Average packet rate = 1145.944837 pkt/s
Packets dropped = 43354 ( 79 %)
Error lines = 0
```

8. decode the sender log file on the log host:

```
[donato@otto donato]$ ITGDec send_log_file
------------------------------------------------------------
Flow number:  3
From 10.0.0.4:34775
To 10.0.0.3:10003
------------------------------------------------------------
Total time = 9.997255 s
Total packets = 28480
Minimum delay = 0.000000 s
Maximum delay = 0.000000 s
Average delay = 0.000000 s
Average jitter = 0.000000 s
Delay standard deviation = 0.000000 s
Bytes received = 14581760
Average bitrate = 11668.611034 Kbit/s
Average packet rate = 2848.781991 pkt/s
Packets dropped = 0 ( 0 %)
------------------------------------------------------------
------------------------------------------------------------
Flow number:  1
From 10.0.0.4:34773
To 10.0.0.3:10001
------------------------------------------------------------
Total time = 9.603001 s
Total packets = 9604
Minimum delay = 0.000000 s
Maximum delay = 0.000000 s
```

```
Average delay = 0.000000 s
Average jitter = 0.000000 s
Delay standard deviation = 0.000000 s
Bytes received = 4917248
Average bitrate = 4096.426107 Kbit/s
Average packet rate = 1000.104030 pkt/s
Packets dropped = 0 ( 0 %)
-------------------------------------------------------------
-------------------------------------------------------------
Flow number:  2
From 10.0.0.4:34774
To 10.0.0.3:10002
-------------------------------------------------------------
Total time = 9.999501 s
Total packets = 20000
Minimum delay = 0.000000 s
Maximum delay = 0.000000 s
Average delay = 0.000000 s
Average jitter = 0.000000 s
Delay standard deviation = 0.000000 s
Bytes received = 10240000
Average bitrate = 8192.408801 Kbit/s
Average packet rate = 2000.099805 pkt/s
Packets dropped = 0 ( 0 %)
-------------------------------------------------------------
*************** TOTAL RESULTS ******************
Number of flows = 3
Total time = 10.013192 s
Total packets = 58084
Minimum delay = 0.000000 s
Maximum delay = 0.000000 s
Average delay = 0.000000 s
Average jitter = 0.000000 s
Delay standard deviation = 0.000000 s
Bytes received = 29739008
Average bitrate = 23759.862390 Kbit/s
Average packet rate = 5800.747654 pkt/s
Packets dropped = 0 ( 0 %)
Error lines = 0
```

7.5 Example 5

VoIP, Telnet and DNS flows towards two distinct destinations

1. start the receiver on the first destination host:

[donato@catarella donato]$ ITGRecv -l recv1_log_file

2. start the receiver on the second destination host:
   [donato@otto donato]$ ITGRecv -l recv2_log_file
3. create the script file [donato@otto donato]$ cat
   script_file
   -a 10.0.0.3 -rp 10001 VoIP -x G.711.2 -h RTP -VAD
   -a 10.0.0.4 -rp 10002 Telnet
   -a 10.0.0.4 -rp 10003 DNS
4. start the sender on the source host: [donato@otto
   donato]$ ITGSend script_file -l sender_log_file
5. close the first receiver by pressing Ctrl-C
6. close the second receiver by pressing Ctrl-C
7. decode the sender log file: [donato@otto donato]$
   ITGDec sender_log_file

```
------------------------------------------------------------
Flow number:  2
From 10.0.0.4:33029
To 10.0.0.4:10002
------------------------------------------------------------
Total time = 9.998991 s
Total packets = 1139
Minimum delay = 0.000000 s
Maximum delay = 0.000000 s
Average delay = 0.000000 s
Average jitter = 0.000000 s
Delay standard deviation = 0.000000 s
Bytes received = 2482
Average bitrate = 1.985800 Kbit/s
Average packet rate = 113.911494 pkt/s
Packets dropped = 0 ( 0 %)
------------------------------------------------------------
------------------------------------------------------------
Flow number:  1
From 10.0.0.4:34776
To 10.0.0.3:10001
------------------------------------------------------------
Total time = 9.980002 s
Total packets = 500
Minimum delay = 0.000000 s
Maximum delay = 0.000000 s
Average delay = 0.000000 s
Average jitter = 0.000000 s
Delay standard deviation = 0.000000 s
Bytes received = 56000
```

```
Average bitrate = 44.889771 Kbit/s
Average packet rate = 50.100190 pkt/s
Packets dropped = 0 ( 0 %)
------------------------------------------------------------
------------------------------------------------------------
Flow number:  3
From 10.0.0.4:34775
To 10.0.0.4:10003
------------------------------------------------------------
Total time = 8.928575 s
Total packets = 6
Minimum delay = 0.000000 s
Maximum delay = 0.000000 s
Average delay = 0.000000 s
Average jitter = 0.000000 s
Delay standard deviation = 0.000000 s
Bytes received = 1507
Average bitrate = 1.350271 Kbit/s
Average packet rate = 0.672000 pkt/s
Packets dropped = 0 ( 0 %)
------------------------------------------------------------
**************** TOTAL RESULTS ******************
Number of flows = 3
Total time = 10.027982 s
Total packets = 1645
Minimum delay = 0.000000 s
Maximum delay = 0.000000 s
Average delay = 0.000000 s
Average jitter = 0.000000 s
Delay standard deviation = 0.000000 s
Bytes received = 59989
Average bitrate = 47.857286 Kbit/s
Average packet rate = 164.040981 pkt/s
Packets dropped = 0 ( 0 %)
Error lines = 0
------------------------------------------------------------
```

8. decode the first receiver log file:

```
[donato@catarella src]$ ITGDec recv1_log_file
------------------------------------------------------------
Flow number:  1
From 10.0.0.4:34776
To 10.0.0.3:10001
------------------------------------------------------------
Total time = 9.980004 s
Total packets = 500
```

```
Minimum delay = 3633.375466 s
Maximum delay = 3633.384447 s
Average delay = 3633.376101 s
Average jitter = 0.000138 s
Delay standard deviation = 0.000259 s
Bytes received = 56000
Average bitrate = 44.889762 Kbit/s
Average packet rate = 50.100180 pkt/s
Packets dropped = 0 ( 0 %)
---------------------------------------------------------
*************** TOTAL RESULTS ******************
Number of flows = 1
Total time = 9.980004 s
Total packets = 500
Minimum delay = 3633.375466 s
Maximum delay = 3633.384447 s
Average delay = 3633.376101 s
Average jitter = 0.000138 s
Delay standard deviation = 0.016080 s
Bytes received = 56000
Average bitrate = 44.889762 Kbit/s
Average packet rate = 50.100180 pkt/s
Packets dropped = 0 ( 0 %)
Error lines = 0
```

9. decode the second receiver log file: [donato@otto donato]$ ITGDec recv2_log_file

```
---------------------------------------------------------
Flow number: 2
From 10.0.0.4:33029
To 10.0.0.4:10002
---------------------------------------------------------
Total time = 9.998989 s
Total packets = 1139
Minimum delay = 0.000019 s
Maximum delay = 0.000934 s
Average delay = 0.000034 s
Average jitter = 0.000014 s
Delay standard deviation = 0.000056 s
Bytes received = 2482
Average bitrate = 1.985801 Kbit/s
Average packet rate = 113.911516 pkt/s
Packets dropped = 0 ( 0 %)
---------------------------------------------------------
---------------------------------------------------------
Flow number: 3
```

```
From 10.0.0.4:34775
To 10.0.0.4:10003
------------------------------------------------------------
Total time = 8.928556 s
Total packets = 6
Minimum delay = 0.000023 s
Maximum delay = 0.000042 s
Average delay = 0.000028 s
Average jitter = 0.000005 s
Delay standard deviation = 0.000006 s
Bytes received = 1507
Average bitrate = 1.350274 Kbit/s
Average packet rate = 0.672001 pkt/s
Packets dropped = 0 ( 0 %)
------------------------------------------------------------
**************** TOTAL RESULTS ******************
Number of flows = 2
Total time = 10.023268 s
Total packets = 1145
Minimum delay = 0.000019 s
Maximum delay = 0.000934 s
Average delay = 0.000034 s
Average jitter = 0.000014 s
Delay standard deviation = 0.007472 s
Bytes received = 3989
Average bitrate = 3.183792 Kbit/s
Average packet rate = 114.234200 pkt/s
Packets dropped = 0 ( 0 %)
Error lines = 0
------------------------------------------------------------
```

7.6 Example 6

Single SCTP flow, with association Id 3 and max outband stream 1, with costant inter-departure time between packets and costant packet size with local sender log

1. start receiver on the destination host (143.225.229.135): [ercolino@localhost bin]$ ./ITGRecv
2. start the sender on the source host: [ercolino@localhost bin]$ ./ITGSend -a 143.225.229.135 -m RTTM -T SCTP 3 1 -rp 9030 -l send_log_file
3. close the ITGRecv by pressing Ctrl-C

4. decode the sender log file on the source host:
   [ercolino@localhost bin]$ ./ITGDec send_log_file

```
/------------------------------------------------------------
Flow number:  1
From 143.225.229.135:32772
To 143.225.229.135:9030
------------------------------------------------------------
Total time = 9.998896 s
Total packets = 10000
Minimum delay = 0.000061 s
Maximum delay = 0.000277 s
Average delay = 0.000079 s
Average jitter = 0.000026 s
Delay standard deviation = 0.000017 s
Bytes received = 5120000
Average bitrate = 4096.452248 Kbit/s
Average packet rate = 1000.110412 pkt/s
Packets dropped = 0 (0.00 %)
------------------------------------------------------------
**************** TOTAL RESULTS *****************
Number of flows = 1
Total time = 9.998896 s
Total packets = 10000
Minimum delay = 0.000061 s
Maximum delay = 0.000277 s
Average delay = 0.000079 s
Average jitter = 0.000026 s
Delay standard deviation = 0.000017 s
Bytes received = 5120000
Average bitrate = 4096.452248 Kbit/s
Average packet rate = 1000.110412 pkt/s
Packets dropped = 0 (0.00 %)
Error lines = 0
------------------------------------------------------------
```

7.7 Example 7

Single DCCP flow with costant inter-departure time between packets and costant packet size with local sender log

1. start receiver on the destination host (143.225.229.135): [ercolino@localhost bin]$ ./ITGRecv
2. start the sender on the source host:
   [ercolino@localhost bin]$ ./ITGSend -a

143.225.229.135 -m RTTM -T DCCP -rp 9030 -l
send_log_file

3. close the ITGRecv by pressing Ctrl-C
4. decode the sender log file on the source host:
   [ercolino@localhost bin]$ ./ITGDec send_log_file

```
/------------------------------------------------------------
Flow number:  1
From 143.225.229.135:47426
To 143.225.229.135:9030
------------------------------------------------------------
Total time = 9.998912 s
Total packets = 10000
Minimum delay = 0.000053 s
Maximum delay = 0.000682 s
Average delay = 0.000101 s
Average jitter = 0.000004 s
Delay standard deviation = 0.000010 s
Bytes received = 5120000
Average bitrate = 4096.445693 Kbit/s
Average packet rate = 1000.108812 pkt/s
Packets dropped = 0 (0.00 %)
------------------------------------------------------------
**************** TOTAL RESULTS ******************
Number of flows = 1
Total time = 9.998912 s
Total packets = 10000
Minimum delay = 0.000053 s
Maximum delay = 0.000682 s
Average delay = 0.000101 s
Average jitter = 0.000004 s
Delay standard deviation = 0.000010 s
Bytes received = 5120000
Average bitrate = 4096.445693 Kbit/s
Average packet rate = 1000.108812 pkt/s
Packets dropped = 0 (0.00 %)
Error lines = 0
------------------------------------------------------------
```

References

[1] A. Dainotti, A. Botta, A. Pescapé, G. Ventre, *"Searching for Invariants in Network Games Traffic"* , Poster at ACM Co-Next 2006 Student Workshop. 2-pages abstract published in Co-Next '06 Proceedings .

[2] T. Lang, P. Branch, G. J. Armitage: *A synthetic traffic model for Quake3*. Advances in Computer Entertainment Technology 2004: 233-238