D-ITG, Distributed Internet Traffic Generator

Antonio Pescapè, Donato Emma, Stefano Avallone, Alessio Botta, and Giorgio Ventre

{pescape@unina.it}

Dipartimento Informatica e Sistemistica
Università degli Studi di Napoli "Federico II"
Naples, Italy

Talk plan

- We called our platform
  - ✓ D-ITG: Distributed Internet Traffic Generator

- Motivations
- Platform Overview
- Platform Architecture
- Performance Analysis
  - ✓ Traffic Generator analysis
  - ✓ Traffic Generation analysis: some results of network performance evaluations performed using D-ITG (in the paper several examples are given).
- Ongoing and Future Work

Why a traffic generator ?

- A traffic generator is useful for:
  - ✓ Network analysis and performance evaluation
    - Throughput, packet loss, delay and jitter analysis of Heterogeneous Networks (Wired LAN, WLAN, GPRS, Bluetooth, …)
  - ✓ Testing device capabilities
    - PC desktop, Laptop/Notebook, Pocket PC, Advanced Mobile Phone, ...
  - ✓ Testing Quality of Service (QoS) architectures
    - Queuing disciplines
    - Traffic shapers
  - ✓ Routing algorithms analysis
  - ✓ Traffic Engineering
  - ✓ Scalability and Protocol behavior analysis

Related Work

- Rude/Crude
  - ✓ UDP only, either constant or trace-based traffic patterns, the receiver only logs packets.
- MGEN
  - ✓ UDP only, a few traffic patterns available, the receiver only logs packets.
- TG2
  - ✓ Cannot simultaneously generate multiple flows, only runs on Unix-based systems.
- Traffic
  - ✓ Does not create log files, only outputs average values, only constant inter-departure time between packets.

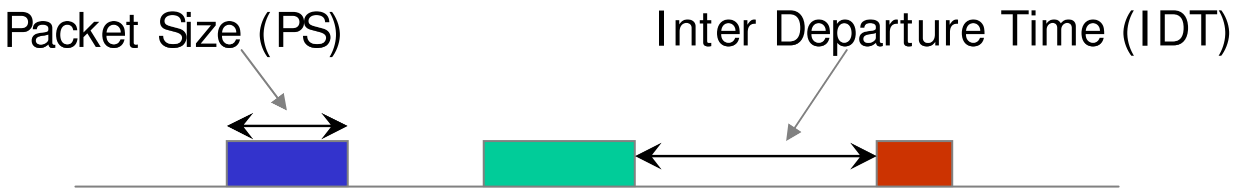
http://www.grid.unina.it/software/ITG/link.php

Why a new traffic generator?

- Improve performance
  - ✓ Increase generated bit rate
  - ✓ Increase received bit rate
  - ✓ Increase scalability
  - ✓ Increase usability
- Add new features
  - ✓ Increase supported stochastic processes
  - ✓ Introduce a Log server
  - ✓ Introduce the Daemon mode
- A multiplatform tool
  - ✓ Linux, Windows, Linux Familiar

Distributed Internet Traffic Generator

- D-ITG analyzes networks generating network traffic on a packet-by-packet basis (*packet-level traffic generator*).
- A traffic flow is specified through
  - ✓ *Packets Inter Departure Time (IDT)* - the time between the transmission of two successive packets
  - ✓ *Packet Size (PS)* - the amount of data being transferred by the packet.
- Both processes are modeled as i.i.d. series of random variables:
  - ✓ constant, uniform, exponential, pareto, normal, cauchy, etc.
  - ✓ it is possible to reproduce exactly the same stochastic experiment by choosing the *same seed* values for IDT and PS random processes.

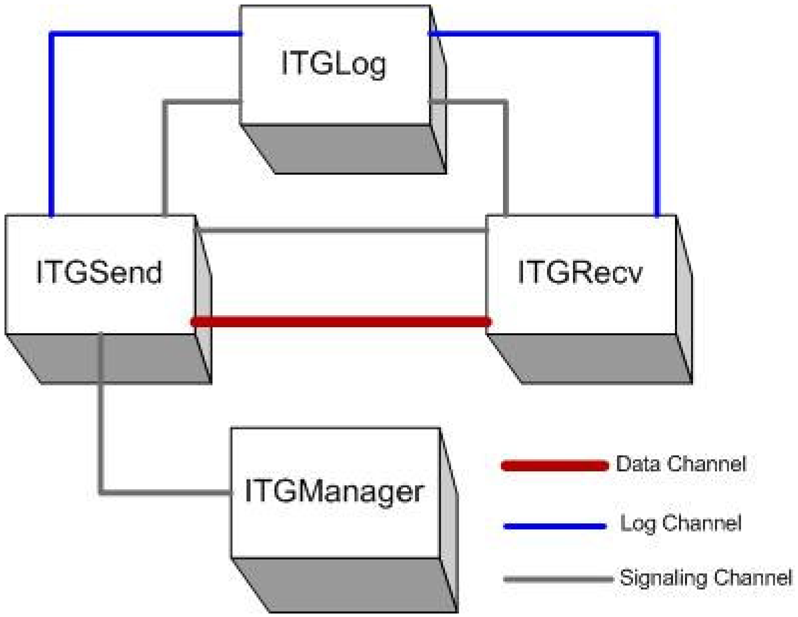


Distributed Internet Traffic Generator

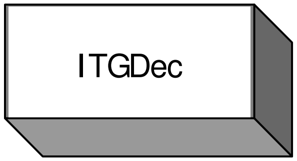
- Some of the basic features:
  - ✓ Runs on Linux, Windows and Linux Familiar platforms.
  - ✓ Both one-way-delay and round-trip-time meter are present.
  - ✓ Allows to simultaneously generate multiple flows:
    - Both the sender and the receiver are multi-threaded applications; each thread manages a single flow.
  - ✓ Both sender and receiver can store information on sent/received traffic:
    - In this way it is possible to isolate the devices' dependencies and network dependencies.
  - ✓ Can reproduce *realistic* traffic patterns. A big effort is devoted to accurately model traffic from several Internet protocols.
    - Currently: TCP, UDP, ICMP and VoIP traffic.
    - Telnet and DNS traffic prototype are under testing.
    - HTTP and Traffic for Internet Games (i.e. Starcraft) are under observation.

D-ITG Architecture

- ITGSend
  - ✓ Sending processes
- ITGRecv
  - ✓ Receiving processes
- ITGLog
  - ✓ Storage server
- ITGManager
  - ✓ Manager for the remote control
- TSP (Traffic Specification Protocol) and the signaling channel
  - ✓ Communications among the D-ITG entities

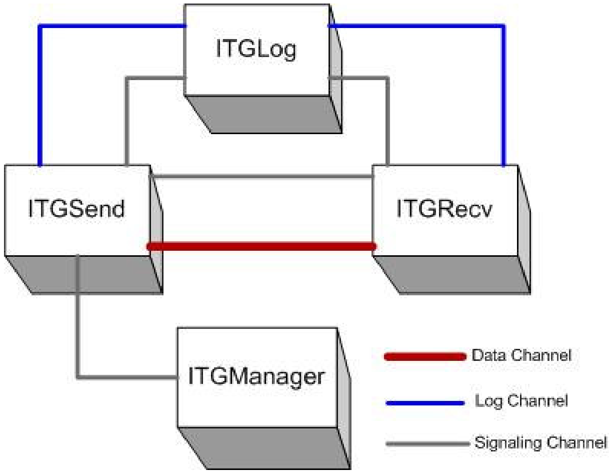


- ITGDec
  - ✓ Results analysis
    - Packet loss
    - Throughput
    - Jitter
    - Delay (OWD and RTT)



Traffic Specification Protocol (TSP)

- We introduced a novel protocol for the definition of each experiment requirements: sender and receiver decide the experiment parameters and control the traffic generation by using TSP.
  - ✓ Create a connection between a sender and a receiver;
  - ✓ Authenticate a receiver;
  - ✓ Exchange information on a generation process;
  - ✓ Close a sender-receiver connection;
  - ✓ Detect generation events.

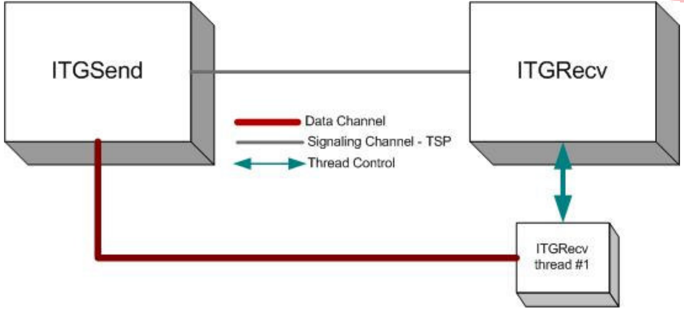


Signaling Channel

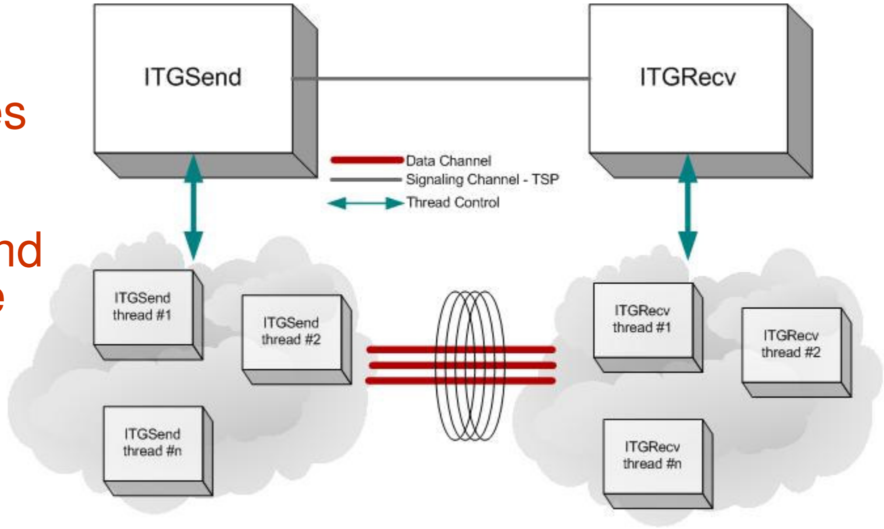
- D-ITG implements the TSP protocol over a *TCP signaling channel* between ITGSend and ITGRecv.
- Thank to the multithreaded implementation of both ITGsend and ITGRecv, each signaling channel can be used for multiple flows generation.
- For each multiple flows generation experiment, a TSP connection between an ITGRecv controller thread and an ITGSend controller thread is established.
- This thread implements the TSP protocol and it is responsible for instantiating and terminating the threads that generate and receive the simulated flows.
- The coordination between the controller thread and the threads that are delegated to generate or receive packets is made using the *Inter Process Communication (IPC)* in Unix-like systems, or the *event communication* in Windows systems.

ITGSend

- Single flow mode: ITGSend generates a single flow; a single thread is responsible for the generation of the flow and the management of the signaling channel through the TSP protocol.



- Multiple flows mode: ITGSend generates a set of flows; it operates as a multithreaded application. One of the threads implements the TSP protocol and drives the generation process, while the others generate the traffic flows.

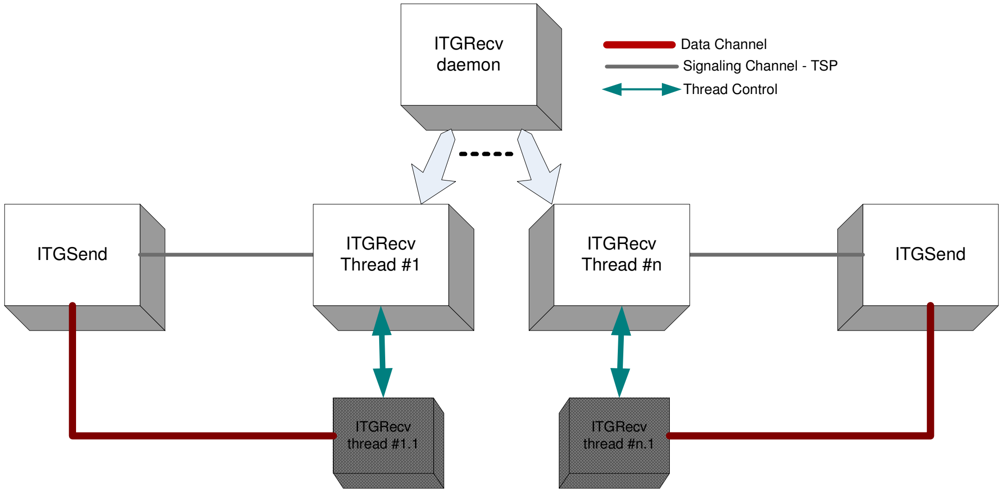


- Daemon mode: ITGSend is remotely controlled by ITGManager using the ITGApi.

- ITGSend can generate a log file that describes at packet level each sent flow. This log file can be stored locally or remotely using the log server ITGLog.

ITGRecv

- ITGRecv always works as a concurrent daemon:
  - ✓ It listens for new TSP connections.
  - ✓ When a TSP connection request arrives, ITGRecv generates a new thread that is responsible for the TSP protocol implementation.
  - ✓ Each single flow is received by a separate thread.
- Before starting a generation experiment ITGRecv and ITGSend implement a challenge-response authentication protocol.
- Similar to ITGSend, *ITGRecv generates a log file that describes at packet level each received flow*. This log file can be stored locally or remotely using the log server ITGLog.



ITGManager

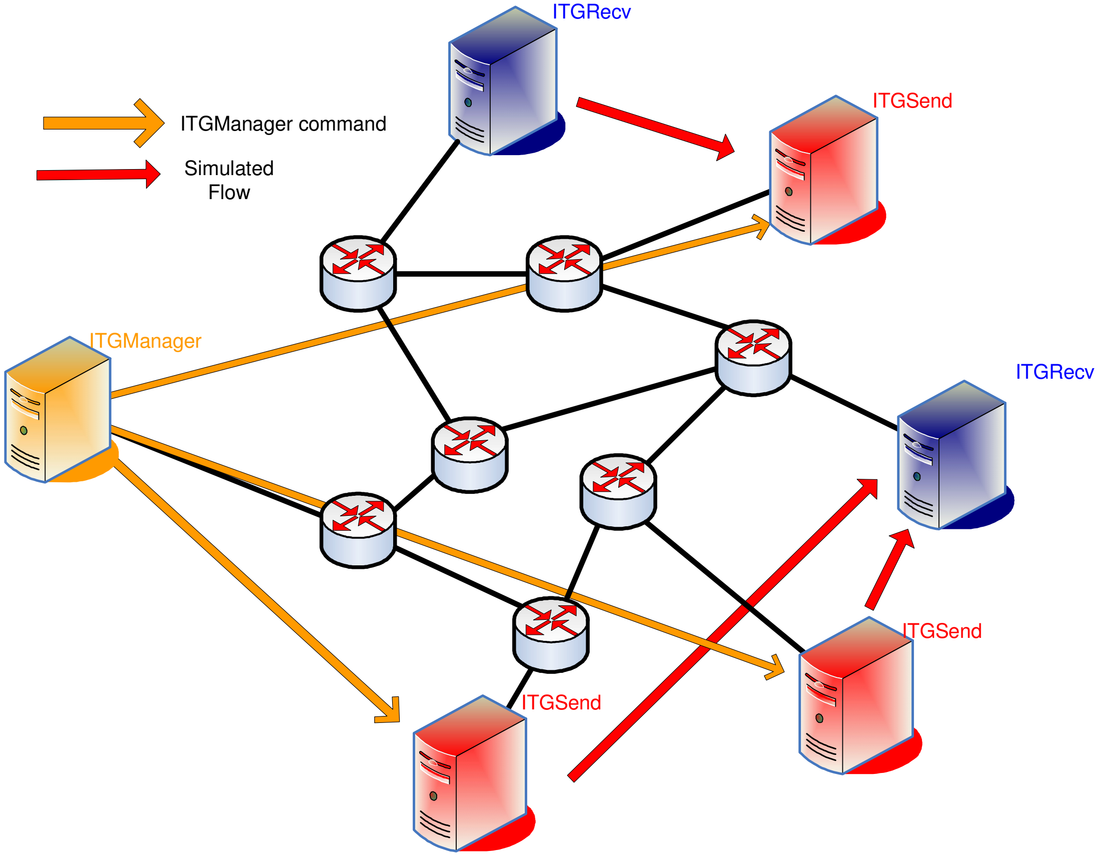
- ITGSend can be launched in daemon mode and stay idle waiting for commands from ITGManager.

- ITGManager uses the ITGApi to remotely control ITGSend.

- ITGApi is an API that can be used to send a message to ITGSend. This message specifies the parameters (destination IP address and port, inter-departure time characterization,…) of the flow to be generated.

- The syntax of messages is the same as that used to require a flow generation from the command line.

- ITGManager can remotely control more than one ITGSend: **ITGManager can control the whole traffic crossing the network**.



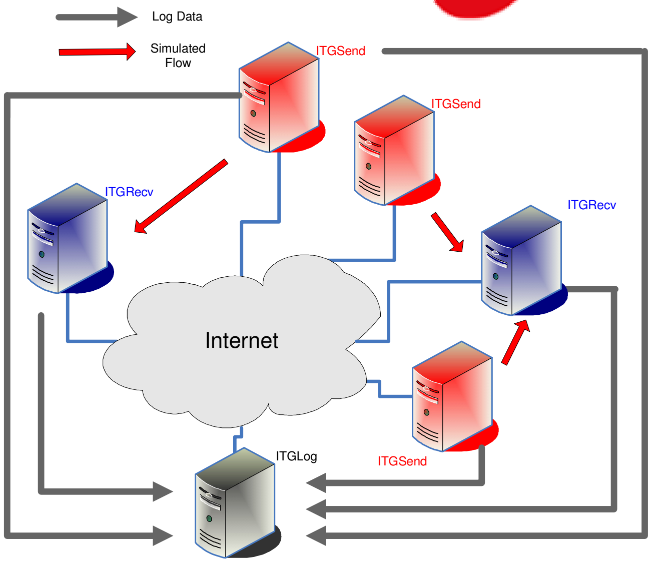
ITGLog (1/1)

- To collect statistics on the generation process ITGSend and ITGRecv can store detailed information (at packet level) about the generated/received flows:
  - ✓ (1) flow number;
  - ✓ (2) sequence number;
  - ✓ (3) source address;
  - ✓ (4) destination address;
  - ✓ (5) transmission time;
  - ✓ (6) receiving time;
  - ✓ (7) packet size.
- This information can be stored *either in a local log file* or *in a remote log file* using the log server ITGLog.
- Log files are processed at a later stage by **ITGDec** in order to provide, for example, average delay (either one-way-delay or round-trip-time), loss rate, jitter…

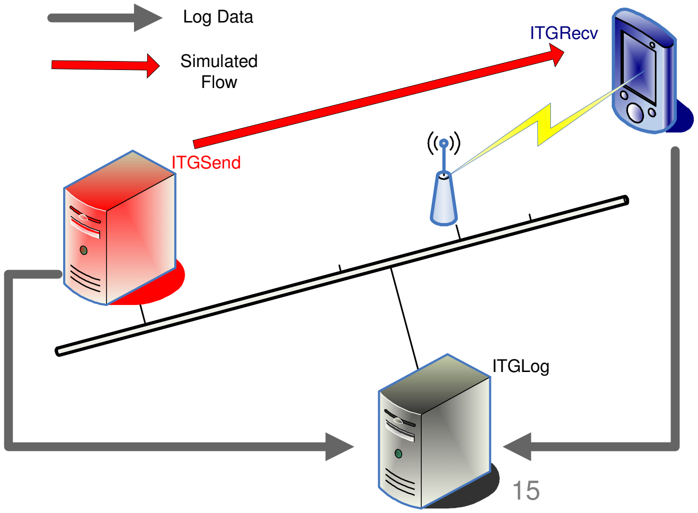
- ITGLog is a “log server”, running on a different host with respect to ITGSend and ITGRecv, which receives and stores the log information from multiple senders and receivers.
- The logging activities is handled using a signaling protocol. It allows each sender/receiver to register on, and to leave, the log server.

ITGLog (2/2)

- The log information can be sent using either a reliable channel (TCP) or an unreliable channel (UDP).
- Since the logging operations are demanded to the log server, senders and receivers do not waste time in storing data. By eliminating the interference of logging operations on generation and reception activities, the performance of both senders and receivers was improved.



- ITGLog can be used in different scenarios such for example:
  - ✓ wide area traffic generation: when D-ITG is used in a wide area distributed scenario ITGLog can be used to easily collect the log file of all the senders/receivers. In this way it is possible to implement *a centralized and possibly "on_the_fly" results analysis*;
  - ✓ devices with limited storage resources: if *a device with limited storage resources*, such as for example a PDA (Personal Digital Assistant), is used to send or receive a traffic flow, ITGLog can be used to collect the log information that can not be stored over a such device.

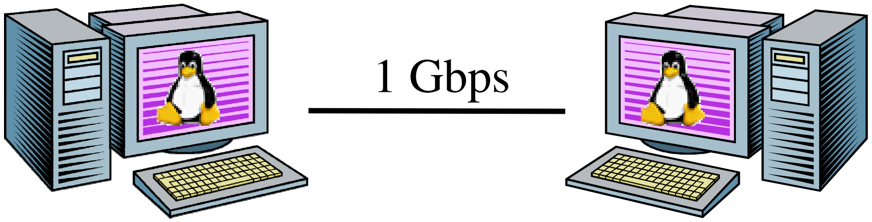


ITGDec

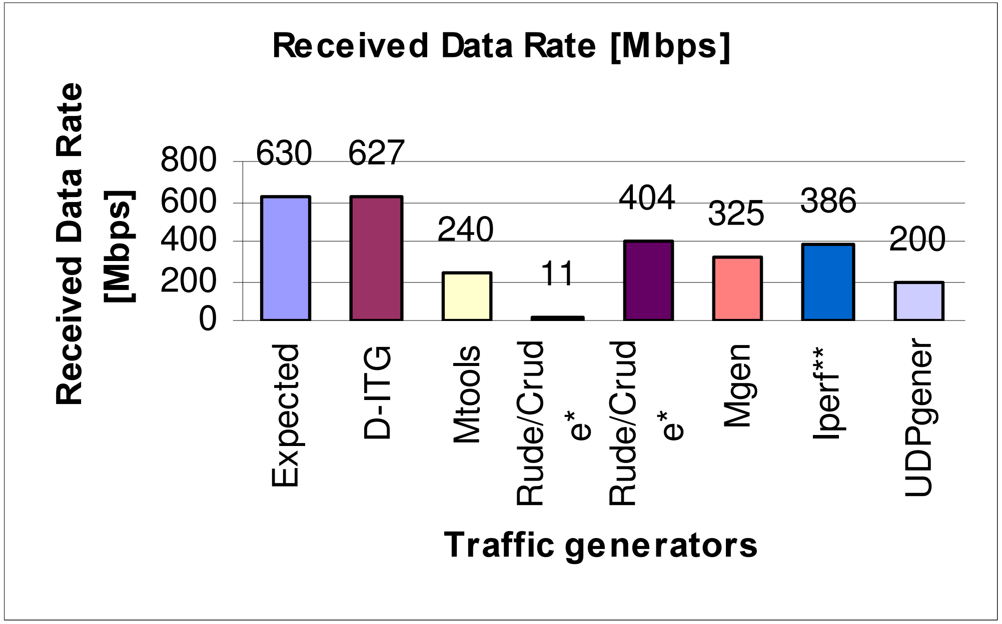
- Statistics related to the generated traffic flows can be collected by analyzing the information stored by both the sender and the receiver.
- ITGDec enables to determine the average values of throughput, delay, jitter and packet loss not only on the whole duration of the experiment, but also on windows of the desired duration.
- Currently, measured parameters are:
  - ✓ Throughput
  - ✓ Delay
  - ✓ Jitter
  - ✓ Packet Loss

D-ITG Performance analysis (1/4)

- Linux platform.
- Flow specifications (UDP):
  - ✓ IDT=77000 pkt/s (constant)
  - ✓ PS=1024 bytes (constant) per packet



- Log file only at receiver side.



Intel Pentium 4 2.6 Ghz

1GB RAM

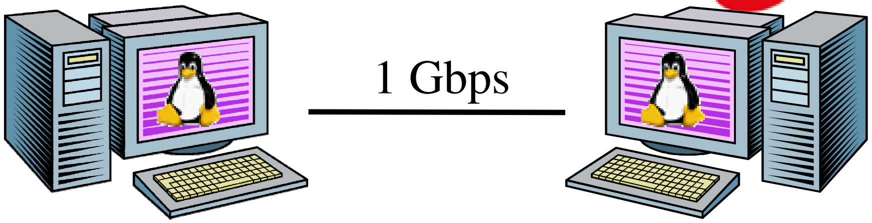
CPU Cache 512kB

NIC: 3Com Gigabit LOM 3c940

2.4.21 Linux kernel

D-ITG Performance analysis (2/4)

- Linux platform.
- Flow specifications (UDP):
  - ✓ IDT=75000 pkt/s (constant)
  - ✓ PS=1024 bytes (constant) per packet
- Log file at both receiver and sender side.



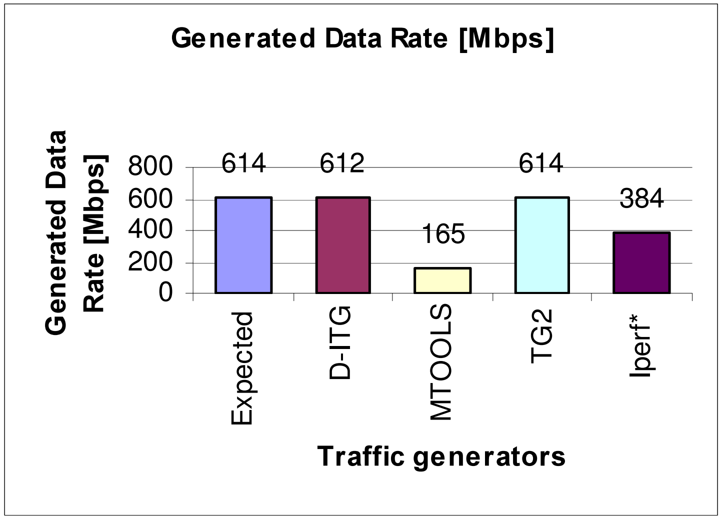
Intel Pentium 4 2.6 Ghz

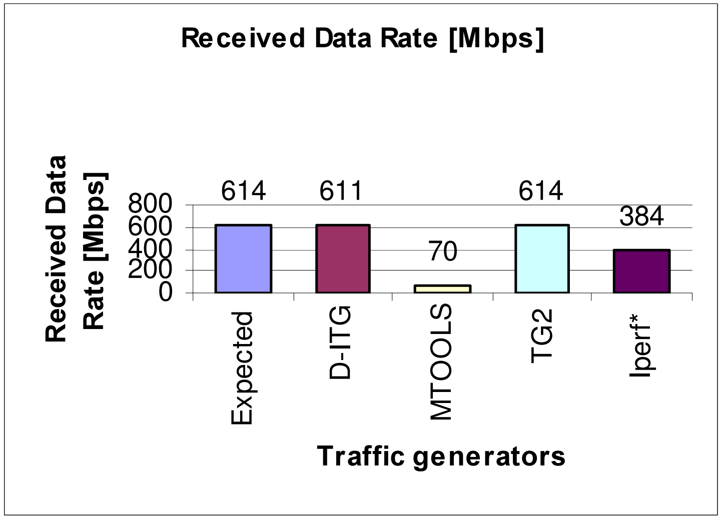
1GB RAM

CPU Cache 512kB

NIC: 3Com Gigabit LOM 3c940

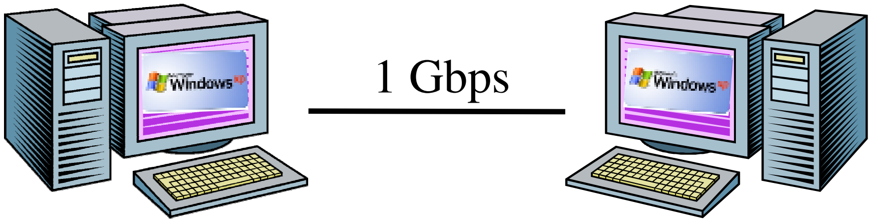
2.4.21 Linux kernel

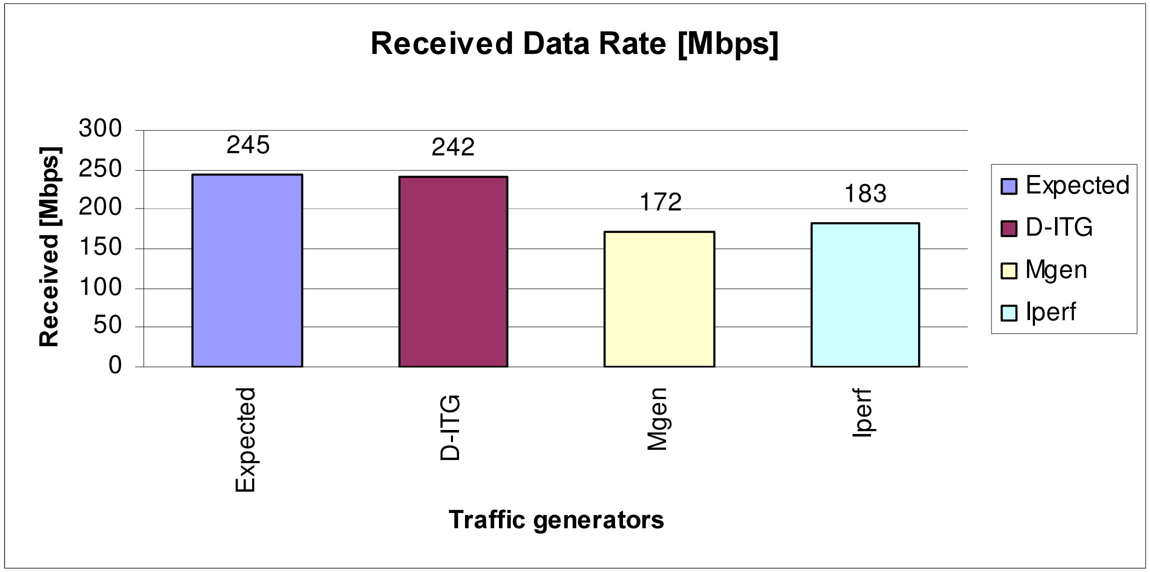




D-ITG Performance analysis (3/4)

- Windows platform.
- Flow specifications (UDP):
  - ✓ IDT=30000 pkt/s (constant)
  - ✓ PS=1024 bytes (constant) per packet
- Log file only at receiver side.





Intel Pentium 4   2.6Ghz

1GB RAM

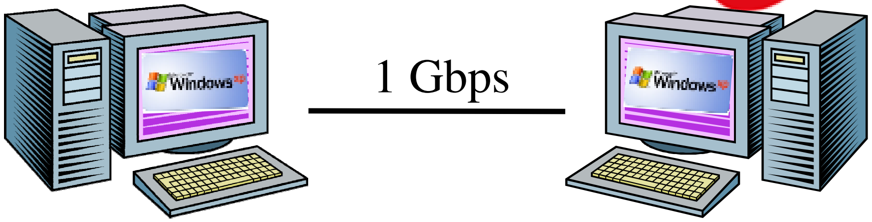
CPU Cache 512kB

NIC : 3Com Gigabit LOM 3c940

Windows XP

D-ITG Performance analysis (4/4)

- Windows platform.
- Flow specifications (UDP):
  - ✓ IDT=30000 pkt/s (constant)
  - ✓ PS=1024 bytes (constant) per packet
- Log file at both sender and receiver side:
  - ✓ D-ITG appears to be today the only traffic generator that, over Windows platform, permits to log at both sender and receiver side.



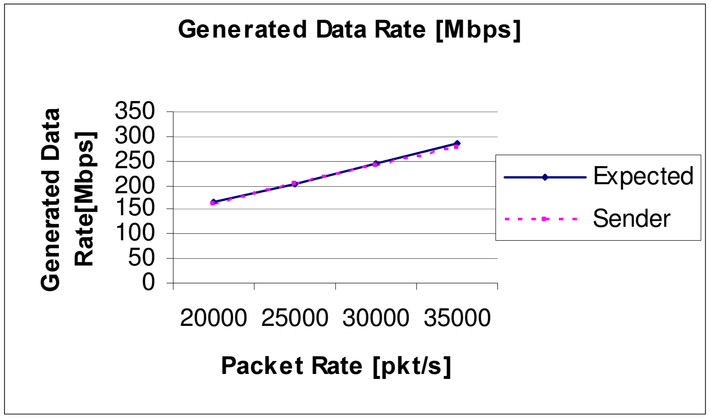
Intel Pentium 4 2.6Ghz

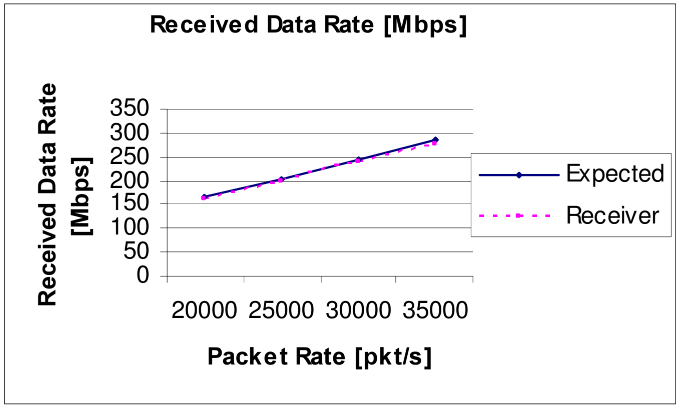
1GB RAM

CPU Cache 512kB

NIC : 3Com Gigabit LOM 3c940

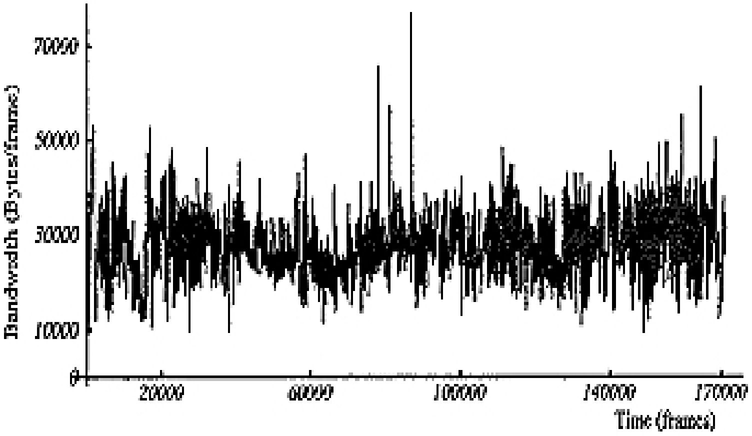
Windows XP





An example of traffic generation

- *UDP* traffic generation, where *PS* follows a *Normal Distribution* and *IDT* is equal to *constant*. In particular, we show the D-ITG accuracy in VBR (Variable Bit Rate) video traffic replication.

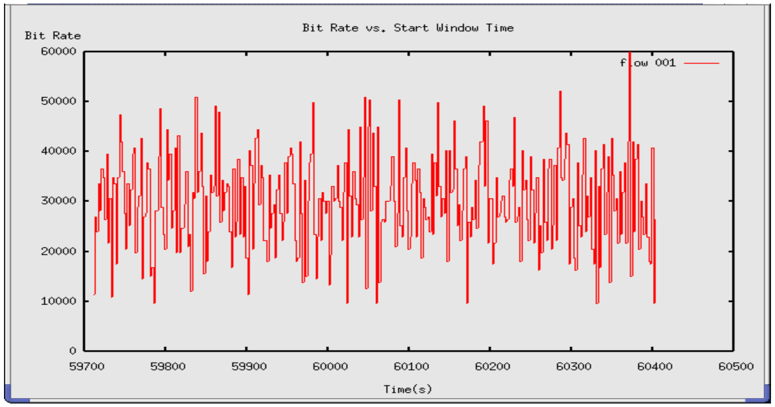


Real trace of VBR video traffic

Duration : two hours

24 frame/s

PS equal to normal with μ = 27791 bytes e σ = 6254 bytes [2]



D-ITG synthetic traffic generation

Duration : 12 minutes

1 frame ≡ 1 packet

IDT constant and equal to 24 pps

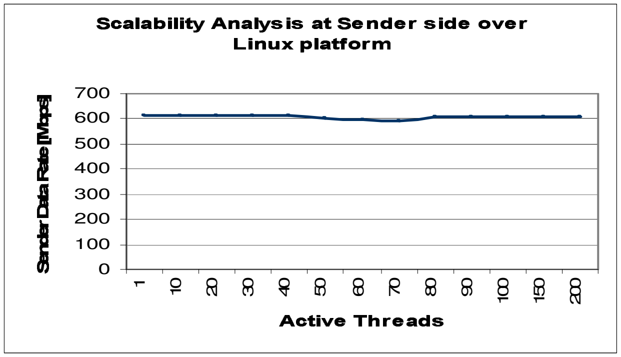
PS equal to normal with μ = 27791 bytes e σ = 6254 bytes

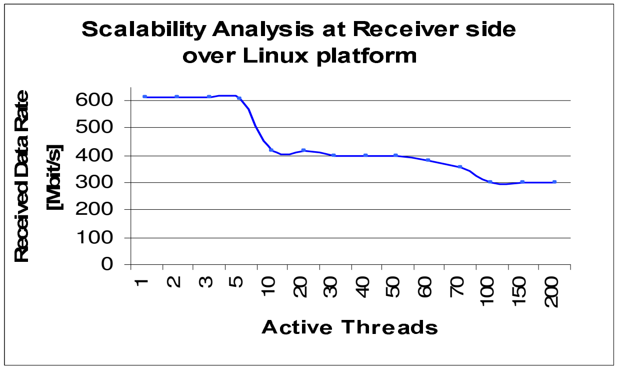
[2] W. Willinger and M.W.Garrett *"Analysis, modeling, and generation of self-similar VBR video traffic"*. In Proc. of the SIGCOMM'94 Conference,pages 269--280, Sept. 1994.

Scalability analysis

- We think that “scalability” is of paramount importance in a networking tool as D-ITG.
- We measured the sender and receiver performance with reference to the number of active threads/flows.

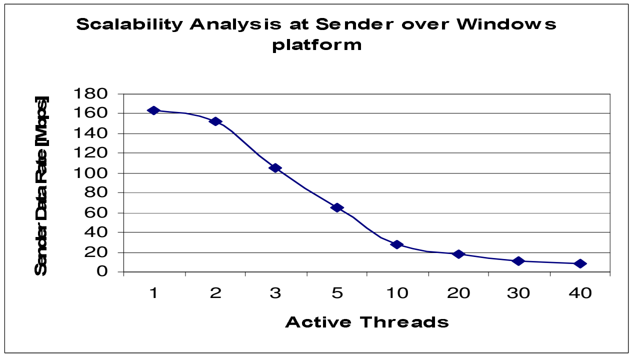
Linux



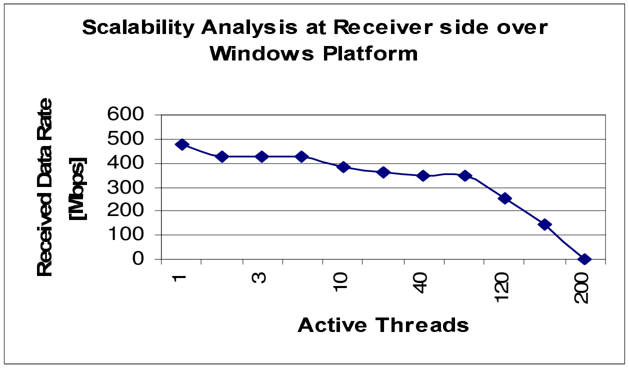


$$\frac{75000}{n} pkt/s$$

Windows



$$\frac{20000}{n} pkt/s$$



Ongoing and Future Work

Ongoing Work:

- IPv6 support.
- Multicast support.
- Porting on FreeBSD and Windows CE/Pocket PC platforms.
- MPI (Message Passing Interface) version.
- Bug fixing.

Future Work:

- Possibility to remotely change transmission rate "*on-the-fly*".
- Adding the models of other application layer protocols (SMTP, SNMP, HTTP, …).

- While (TRUE)

    Bug fixing
- Endwhile

D-ITG: website

- D-ITG is freely (under GPL license) downloadable from:

  http://www.grid.unina.it/software/ITG/

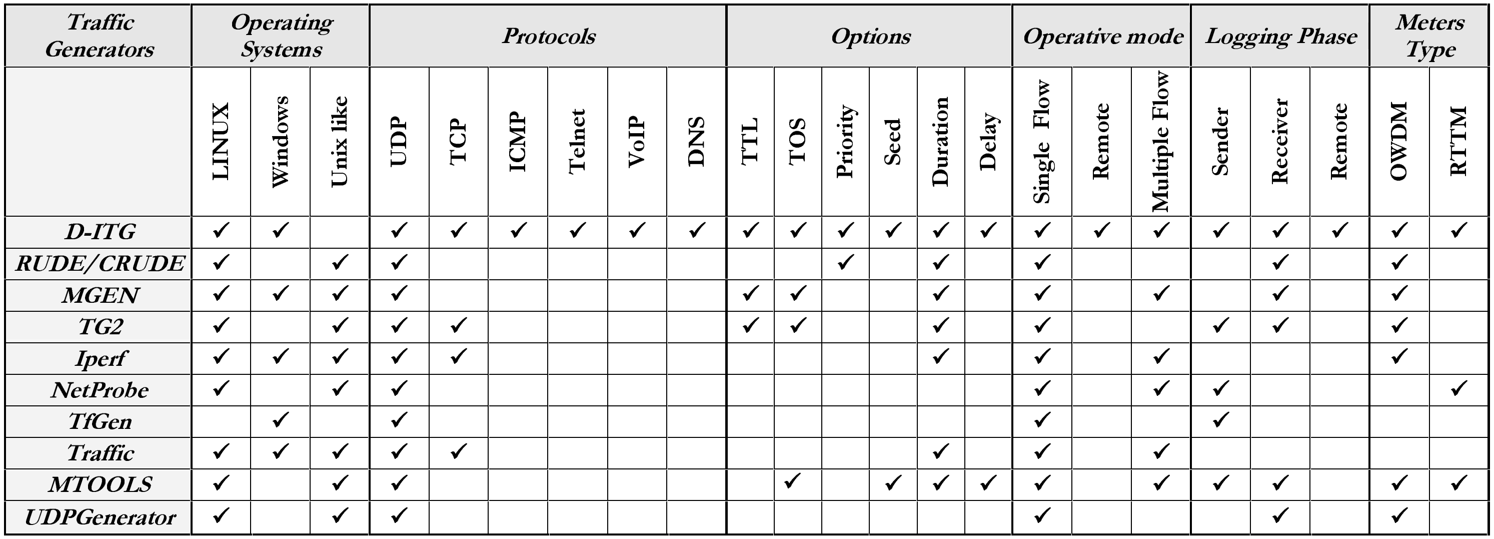
- Currently we release the 2.3 version.
- At the end of september 2004 the 2.4 version will be released.

Thank you !!!

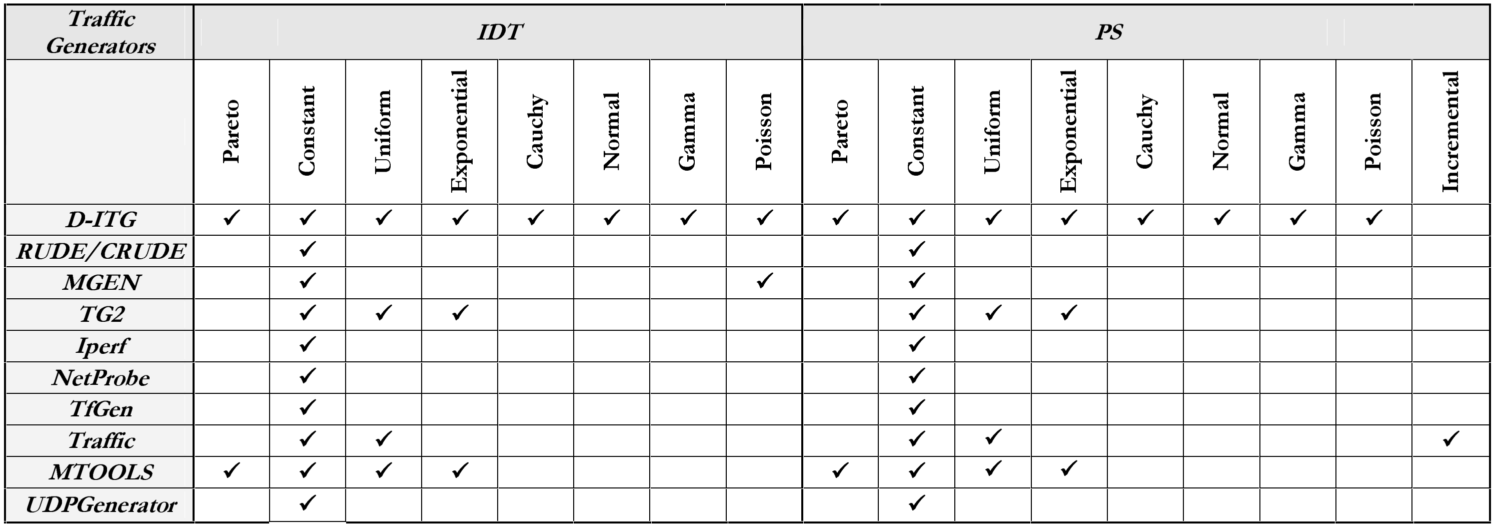
Any Questions ?

http://www.grid.unina.it/software/ITG/

Related work (2)



Traffic Generators	*Operating Systems*			*Protocols*						*Options*						*Operative mode*			*Logging Phase*			*Meters Type*	
	LINUX	Windows	Unix like	UDP	TCP	ICMP	Telnet	VoIP	DNS	TTL	TOS	Priority	Seed	Duration	Delay	Single Flow	Remote	Multiple Flow	Sender	Receiver	Remote	OWDM	RTTM
D-ITG	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
RUDE/CRUDE	✓		✓	✓								✓		✓		✓				✓		✓	
MGEN	✓	✓	✓	✓						✓	✓			✓		✓		✓		✓		✓	
TG2	✓		✓	✓	✓					✓	✓			✓		✓			✓	✓		✓	
Iperf	✓	✓	✓	✓	✓									✓		✓		✓				✓	
NetProbe	✓		✓	✓												✓		✓	✓				✓
TfGen		✓		✓												✓			✓				
Traffic	✓	✓	✓	✓	✓									✓		✓		✓					
MTOOLS	✓		✓	✓							✓		✓	✓	✓	✓		✓	✓	✓		✓	✓
UDPGenerator	✓		✓	✓												✓				✓		✓	



Traffic Generators	*IDT*								*PS*								
	Pareto	Constant	Uniform	Exponential	Cauchy	Normal	Gamma	Poisson	Pareto	Constant	Uniform	Exponential	Cauchy	Normal	Gamma	Poisson	Incremental
D-ITG	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
RUDE/CRUDE		✓								✓							
MGEN		✓						✓		✓							
TG2		✓	✓	✓						✓	✓	✓					
Iperf		✓								✓							
NetProbe		✓								✓							
TfGen		✓								✓							
Traffic		✓	✓							✓	✓						✓
MTOOLS	✓	✓	✓	✓					✓	✓	✓	✓					
UDPGenerator		✓								✓							