ICCT'92

Proceedings of 1992 International Conference on Communication Technology

Beijing, China, Sept. 16–18, 1992

Vol. 1 of 2

Supported by
Siemens, Germany
National Natural Science Foundation of China

Sponsored by the CIE, the CIC and Tsinghua University, in cooperation with the IEEE Communication Society and the IEEE Beijing Section

A TRAFFIC GENERATOR FOR THE PERFORMANCE EVALUATION OF A SATELLITE NETWORK

N. Celandroni (+), E. Ferro (+), F. Potorti (#)

(+) CNUCE, Institute of National Research Council
Via S. Maria 36 - 56126 Pisa (Italy)
Fax: +39-50-589354 Telex: 500371
Phone: +39-50-593207/593312
Email: nedo@icnucevm.CNUCE.CNR.IT
Email: erina@icnucevm.CNUCE.CNR.IT
(#) Telespazio S.p.A scholarship-holder at CNUCE
Email: pot@fdt.CNUCE.CNR.IT

Abstract

A multi-application traffic generator (MTG), whose purpose is the generation of packets over a LAN, is presented. It was born as a tool for the performance measurement of a complex TDMA satellite access scheme currently being tested at CNUCE.

The generated traffic simulates the traffic produced by a number of both isochronous and anisochronous applications, thus allowing the measurement of many parameters relevant to the communication network. From the test point of view, data generated by the MTG system is equivalent to data generated by real applications spread over a network.

The MTG system is described and its performance figures are shown.

1. Introduction

Satellite networks represent, at the moment, a possibility to provide integrated services at bit rates of the order of Mbit/s, provided that the used satellite access scheme is able to support the requirements of different types of traffic.

The presented MTG was born as a tool for evaluating the performance of such a satellite network. It uses the FODA/IBEA(Fifo Ordered Demand Assignment / Information Bit Energy Adapter) system: a satellite access scheme operating in TDMA [11]. The modular architecture of MTG makes the evaluation of any other communication system possible, provided that some modules are replaced.

FODA/IBEA is the natural evolution of the FODA access scheme, developed at CNUCE and tested in the framework of the SATINE-II experiment on the ECS-II satellite [1, 2]. Like FODA, FODA/IBEA allows the simultaneous transmission of both isochronous and anisochronous data (*stream traffic* and *datagram traffic*, respectively), sharing the use in TDMA of a satellite channel. In addition, FODA/IBEA is able to counter the fade of the signal due to bad atmospheric conditions, by dynamically adapting the data coding and bit rates to the channel conditions.

The FODA/IBEA hardware consists of a prototype of a TDMA controller and of a variable burst-rate modem, ranging from 1 up to 8 Mbit/s. The satellite network links together 4 different earth stations in Italy, for a LAN interconnection via satellite experiment in the framework of the Olympus Utilisation Program. More stations are foreseen for an extension of the project in a European environment (COST 226 project). At each site, a router connected to a LAN collects all the traffic directed towards the satellite. The traffic is supposed to come from different types of applications (voice, video-conferencing, file transfer, interactive terminal access, etc.).

The router is connected to the earth station via two dedicated stubs of Ethernet cable. The communication between them follows a specialised protocol, called the *GA-FO* protocol [3].

The network architecture is shown in Fig. 1.

2. Why artificial traffic generation

The testing of a complex communication system in a real environment poses the problem of controlling the characteristics and allowing the reproducibility of the traffic feeding the system during the performance measurement procedure. A common way to solve this problem is a simulation approach, requiring a detailed description of the communication system under trial and of the traffic pattern feeding the system. The detail used in the simulation of the system is crucial to its quality, but it must necessarily be limited in order to avoid an excessive complexity of the simulation program. The analytic solution (queuing theory) for the performance investigation must generally introduce an even more drastic simplification than the simulation approach.

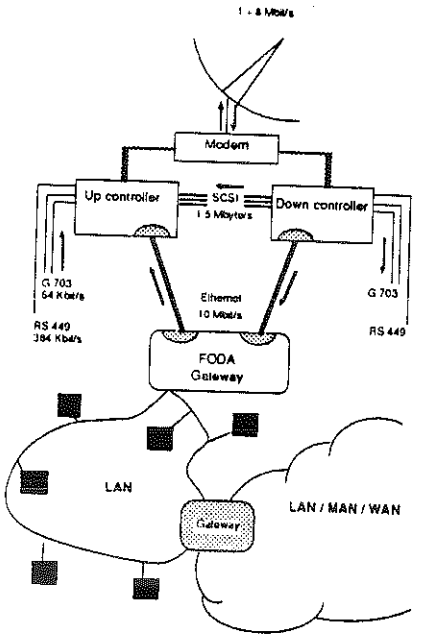

Fig. 1. The network architecture

The environment simulation technique provides the most realistic testing technique for telecommunications systems. Indeed, there is no approximation with regard to the system's behaviour, since the traffic only is simulated. Moreover, the traffic generator can be conveniently used during the debugging phase of the communication system prototype.

A number of traffic simulation approaches can be found in the literature. Most of them are dedicated simulators that can only be applied to a particular system. Other approaches deal with more system-independent concepts but they do not allow the simultaneous generation of stream and datagram traffic [6]. UNES [10], for example, is a versatile environment simulator for load tests of switching systems software, but it is designed for telephone switching systems only.

MTG is aimed at generating a traffic of stream and datagram packets on a LAN in controlled and reproducible conditions. It can simulate many different devices connected to the satellite network, making the performance investigation (such as studies on the effects that the access scheme has on network delays, throughput, congestion and quality of service) easier, and allowing the tune-up of many key parameters of the access scheme. Each of the traffic sources may be a packet voice or video source, a gateway with other LANs, a computer running independent applications which need to access the LAN, and so on.

3. The MTG architecture

MTG produces traffic that comes from the simultaneous activity of a number of user-defined independent sources called Traffic Generators (TGs). Thanks to the combination of the traffic generated by all the TGs, it is possible to simulate arbitrary complex traffic patterns, defined by a comparatively small set of parameters (detailed in section 5). The parameters of all the TGs are contained in an input file. All the TGs act concurrently during the simulation: the overall traffic generated by MTG is the sum of the traffic individually generated by each TG.

MTG consists of two distinct sections — called *controller* and *driver* respectively — that interact via a well defined interface. The controller is dependent on the communication protocol only, while the driver is tied to the LAN and it is independent of the protocol. The present implementation of MTG is based on Ethernet. The block structure organisation of MTG offers some flexibility. In fact, it will be possible to test the FODA/IBEA system over different LANs or MANs (Token Ring, FDDI, etc.) by changing the driver section only. On the other hand, any other type of network can, in principle, be tested if the controller section is changed in order to support a different communication protocol.

In the simplest configuration (one MTG only), data generated by MTG is looped back, received by MTG itself and recorded onto disk after having crossed the entire network (Fig. 2).

The packets are generated according to a few statistical

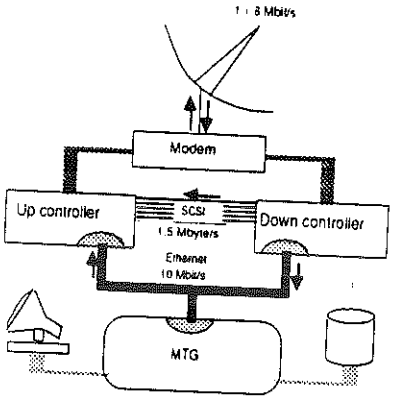

Figure 2. MTG data loop-back configuration

parameters which manage the packet generation processes, the packet lengths and burstiness.

The measurements made by MTG on each packet can be elaborated in order to produce deterministic surveys and statistical distributions of some interesting quantities relevant to the FODA/IBEA system. The most important of these quantities are: the end-to-end delay, the stream packet jitters, the queuing time in the Up and Down controllers and the data bit error rate (BER). It is also possible to tune-up the fast Eb/No estimator (internal to the controller), based on the soft decision level of the data coming from the demodulator and to check the choice of the bit and coding rates made by the system during variable conditions of the channel quality. The FODA/IBEA system supports variable-bandwidth applications which can be requested to reduce their bandwidth in particular fade conditions of the station. The simulation of these types of applications is another feature supported by MTG.

Since MTG can create arbitrary traffic situations, th FODA/IBEA system can also be tested under extreme or very

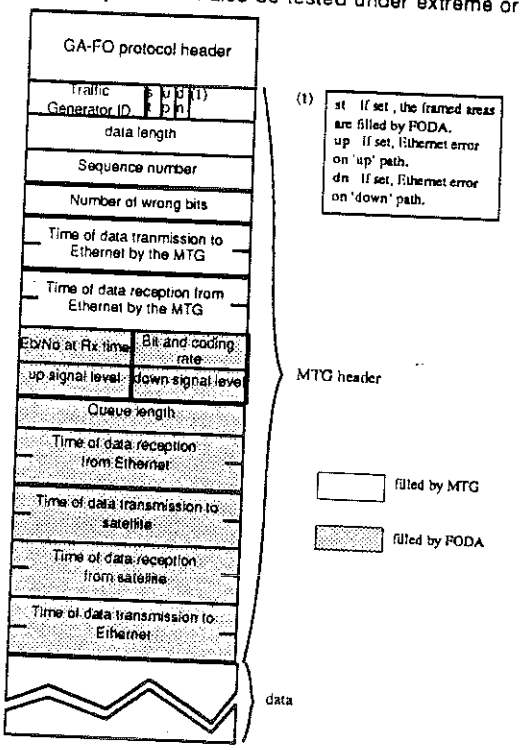

Fig.3. The MTG header

unlikely conditions, in order to check the system robustness.

4. The MTG header

An *MTG header* is put on top of the data section of each generated packet. It contains information about time stamps, data errors and the conditions of the communication channel.

The *traffic generator id* distinguishes the traffic of the various active TGs.

The *sequence number* allows the detection of packet losses.

The two bit fields - *up* and *dn* - are used to flag unusual errors on the LAN.

Both the *number of wrong bits* and the *data length* allow the BER estimation.

The *time stamps* are expressed in *micro ticks*, i.e. the time unit internal to the system (currently set to 100 µs). The accuracy of the time stamps is one of the most severe requisites of the MTG system.

The four FODA/IBEA time stamps allow the measurement of:

- the application end-to-end delay,
- the satellite network delay,
- the queuing times of packets received by Ethernet and waiting for transmission to the satellite,

- the queuing times of packets received by the satellite and waiting for transmission to Ethernet,
- the packet jitter before and after the satellite network crossing.

The end-to-end delay is experienced by a packet from when it leaves the generator until it arrives at the recorder. This parameter is significant only if the recorder MTG is synchronised with the generator MTG. When many MTGs work together, it is foreseen that a "master" MTG sends a synchronisation packet in order to permit a global consistency of the clocks.

The queuing delays are the times spent by each packet inside the Up and Down controllers. The packet queue lengths can be monitored as well. All these quantities are extremely useful in tuning the system parameters in order to improve the performance.

Generally, it is possible to retrace the history of the delay and of the jitter of each packet at different test points of the system on trial. For each received packet, MTG records the MTG header onto disk. A subsequent elaboration of such data produces the requested statistics.

5. The TG definition parameter

The TG parameters, shown in Table 1, are here briefly described.

• DATA TYPE:	Stream / Interactive / Bulk
DISTRIBUTION:	Fixed Rate / Random / Poisson / Voice
PACKET SIZE:	Value [bytes]
MEAN THROUGHPUT:	Value [Kbit/s]
PATTERN:	Fixed Pattern / Random / Incremental
START TIME:	Absolute / Relative / Operator request / Poisson
DURATION:	Absolute / Relative / Operator request / Poisson
JITTER:	Value [ms]
• COS:	Data Class of Service
OPTIONS:	Options mask
• ADDRESS:	Myself / To a particular station / Broadcast

Table I. Definition parameters

DATA TYPE allows the selection of the stream type traffic and of two different classes of datagram traffic (interactive and bulk). FODA/IBEA gives different transmission priorities to the data belonging to these two datagram classes.

Each TG generates packets according to the *DISTRIBUTION* parameter at such time instants, according to the *MEAN THROUGHPUT* and the *PACKET SIZE* parameters. A *Fixed Rate* distribution means packets with constant length, sent at constant intervals of time. A *Random* distribution means packets of variable length, sent at constant intervals of time. The length is varied according to a pseudo-random variable with uniform distribution in a given range. A *Poisson* distribution means fixed-length packets, sent at intervals of time which vary according to a pseudo-random exponential distributed variable. A *Voice* distribution emulates voice packets with the silence detection feature. A fixed rate stream of packets is modulated ON/OFF, according to the talk-silence periods of the English speech distribution. More distributions, analytically or numerically defined, can be easily added to the ones so far envisaged.

A bit *PATTERN* can be defined for the data contained in a packet. Three pattern types are defined. The basic unit in the pattern is the 32-bit word. The *Fixed pattern* is the repetition of a given word. The *Random* pattern is a sequence of random, constant-distributed words. The *Incremental* pattern is a sequence of words in ascending order, starting from a given value.

START TIME and *DURATION* define the life span of a TG. A TG starts sending packets with the given distribution at the start time and goes on for a time interval equal to the duration. *Absolute* refers to a given time. *Relative* indicates a delay with respect to the start of the run. *Operator request* means that the TG is started/stopped by an operator command (the program is fully interactive). *Poisson* (used for the simulation of phone calls) indicates that either the start or the duration times are calculated according to pseudo-random variables following negative exponential distributions.

JITTER indicates that a pseudo-random variable, in the range of $\pm Value$, can be optionally added to the sending time of each packet. At present, only the uniform distribution has been implemented for the jitter value, but the addition of other distributions is straightforward.

The *COS* (Class Of Service) field is a number used to specify the range of the requested bit error rate.

The *OPTION MASK* field is a bit mask used to specify some options, such as the setting of the *up* and *dn* bits in the MTG header, or the enabling of the wrong bits counter in real time.

ADDRESS is the destination earth station where data are addressed.

6. Random number generator

Any simulation tool needs samples of quantities defined by their probability distribution. The samples are usually the output of deterministic mathematical algorithms known as pseudo-random number generators. Particular care is required when choosing the algorithm for a particular simulation application. The issues and the constraints involved in the random number generation internal to MTG are here discussed.

All the random numbers used within MTG are generated off-line. This means that a table is created for each TG at the MTG start-up, prior to the simulation run, containing entries filled with the starting time and length of each packet. The parameters for the generation of the tables are taken from the input file which describes the characteristics of the TGs.

The choice of generating table-based random numbers has been made because the MTG is a software tool which uses the entire processing power of the CPU board. Different approaches can be found in the literature. Ramshaw and Amer [7] have built a test system based on eight traffic generators for the NBSNET at the National Bureau of Standards. Each of the traffic generators — based on a 8-bit microcomputer — has a maximum throughput of one packet per millisecond. This constraint leaves 400µs for the uniform-distributed random number generation and for the mapping to the desired distribution.

This approach is not possible for MTG. Indeed, MTG can run a number of concurrent TGs, each one able to generate packets as fast as one per millisecond, resulting in a total throughput of up to several packets per millisecond. At the same time, MTG receives the looped-back traffic and writes the packet headers onto disk. The architecture of MTG, entirely based on software, thus makes the on-line generation of the random numbers impossible, due to the extra charge that would be imposed on the CPU.

The adopted table-based solution has well known advantages and drawbacks. Indeed, it is possible to choose a good generator without worrying about computing times, and it is possible to exactly transform the uniform distributed random number sequence into an arbitrary distribution (for example, the exponential distribution). The major advantage is that at run time it is only required to read a number in a table, instead of requiring an on-line generation. The drawback is that the period of the generated sequence is equal to the length of the table.

The adopted uniform random number generator is a Lehmer's generator with modulus $2^{31}-1$ and multiplier 950706376 [8]. The formula used to transform a uniform distributed variable into a, say, exponential variable is the

inversion. The choice of the length to be adopted for the tables is not a secondary one, since the goodness of the generator can be significantly impaired by the table-based approach. To get an idea of the generator's period, let us consider the following case in which 64 Poisson TGs generate 256-byte packets at a mean rate of 64 Kbit/s. The mean inter-packet time is 8 ms. With the used hardware it is possible to dedicate 2 Mbyte of memory to the tables. Since each table entry requires 8 bytes, the generation period is 32s. All the transients observed during the simulation runs of the FODA/IBEA system showed a much shorter length.

7. Implementation and real-time problems

A Motorola Delta 3300 single-board microcomputer running the UNIX System V 3.6 is the MTG support. It features the following characteristics:

- VME bus interface,
- Motorola 68030 µp and 68882 math co-processor running at 25 MHz,
- 8 Mbyte memory,
- two independent counter-timers with 6.25 µs time resolution,
- Local Area Network Controller for Ethernet AM7990(LANCE) - Serial Interface Adapter AM7992(SIA) chip set for Ethernet interface,
- SCSI interface,
- 1.5 Mbyte/s transfer rate, 16.5 ms av. access time disk.

The UNIX system provides a comfortable environment for the programmer, but poses fundamental obstacles because of its non real-time behaviour. Real-time behaviour is vital to MTG because of the precision required by the packet dispatching scheduling times and the time stamps contained in the MTG header. Since System V currently provides non standard means to overcome this problem, a brute force approach has been taken. The devices related to MTG (the LAN manager and the timer) have been forced to generate non-maskable interrupts, while the operating system is free to carry out its work. This is necessary because MTG requests the operating system services when writing onto disk. This technique allows full use of processor power. As a consequence there is practically no disk writing overhead, if the UNIX I/O buffering parameters are suitably chosen.

# of TGs	Inter-packet generation time for each TG	packet length (bytes)	traffic volume in TX	traffic volume in Rx	header per # packets	test run #
2	1 ms 512 Kbit/s per TG	64	1.024 Mbit/s	1.024 Mbit/s	1	1
3	1 ms 512 Kbit/s per TG	64	1.536 Mbit/s	1.536 Mbit/s	100.000	2
8	1 ms 512 Kbit/s per TG	64	4.96 Mbit/s	simplex	0	3
32	32 ms 69.5 Kbit/s per TG	22+256	2.224 Mbit/s	2.224 Mbit/s	1	4
90	32 ms 69.5 Kbit/s per TG	22+256	6.255 Mbit/s	simplex	0	5
43	62 ms 195 Kbit/s per TG	1514	8.4 Mbit/s	simplex	0	6
22	30 ms 404 Kbit/s per TG	1514	8.882 Mbit/s	simplex	0	7
12	16 ms 757 Kbit/s per TG	1514	9.084 Mbit/s	simplex	0	8
4	5 ms 2345 Kbit/s per TG	1466	9.382 Mbit/s	simplex	0	9

Table 2. Performance evaluation tests scenario

The interaction between the controller and the driver does not involve frequent interactions with the operating system. System calls are limited to cases such as the start or stop of a TG, the writing of a big disk buffer containing the headers of the received packets, or the check for operator interaction. The LAN packet sending/receiving operations do not rely on the operating system. Data to be sent is contained in only one buffer for each TG. This buffer is filled (during the pre-run phase) up to the maximum length and a fixed portion (according to the current length) is sent at each dispatch time.

8. The MTG performance

At the time of writing, the FODA/IBEA software development is finished and its debugging is just starting. The aim of the present paper is to present the performance of the MTG system while the satellite network performance will be the subject of a future paper. The FODA/IBEA system has been simulated by means of an Ethernet responder which redirects the generated data to MTG. In Table 2 the MTG performance under different test conditions is presented. Each test was run for 300 seconds. Ethernet was used both in simplex and in contention mode. Neither the maximum throughput (10 Mbit/s) nor the maximum number of packets per second (14,880) allowed by Ethernet, when used in simplex mode, can be reached because only one packet at a time must be handled in order to set up accurate time stamps. Anyway, the main purpose of MTG is to allow adequate accuracy in the parameter estimation, rather than to attain the maximum possible throughput on Ethernet.

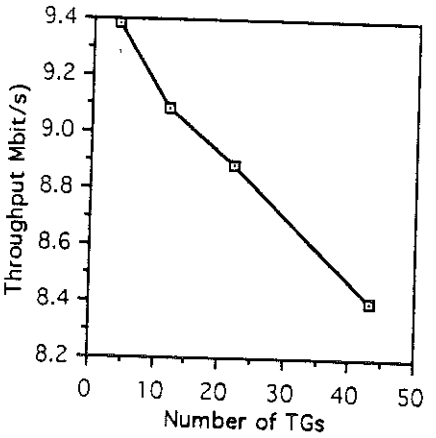

Fig.4. Max throughput as a function of the number of TGs

An Ethernet cable connected MTG with the responder during all the tests aimed at measuring the recording capabilities of MTG. The traffic looped back by the responder was collected by MTG and the header of each packet (40 bytes) was recorded onto disk. The responder was not used in the tests where the traffic generation performance only was investigated.

Tests 1÷3 were aimed at measuring the maximum number of packets per second MTG can handle. Transmission only and transmission/reception with and without disk recording were the test conditions. The packet length does not influence the MTG performance, as long as Ethernet can be viewed as an infinite bandwidth medium, so a very small packet length was used to avoid collisions.

Tests 4 and 5 give the maximum number of supportable 64 Kbit/s applications. Two results are given, with and without the packet header recording onto disk. The packet length was set to 256 bytes plus a 22-byte header containing the Ethernet and the GA-FO protocol.

In the last five tests the maximum reachable total throughput was investigated with different numbers of running generators.