ELSEVIER

Performance Evaluation 24 (1995) 137–159

PERFORMANCE EVALUATION
An International Journal

ATM simulation with CLASS

M. Ajmone Marsan [a,*], A. Bianco [a], T.V. Do [b], L. Jereb [b], R. Lo Cigno [a], M. Munafò [a]

[a] *Dipartimento di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy*

[b] *Department of Telecommunications, Technical University of Budapest, Sztoczek 2, 1111 Budapest, Hungary*

Abstract

The paper describes an efficient, versatile and extensible software tool for the analysis of the quality of connectionless services in ATM networks. The tool is named CLASS for ConnectionLess ATM Services Simulator. CLASS is a time-driven, slotted, synchronous simulator, entirely written in standard C language. CLASS allows the performance analysis of ATM networks adopting the viewpoint of both the end-user, and the network manager; the measured performance parameters include the cell and message loss probabilities and the cell and message delay jitters. The investigation of the impact of shaping and policing techniques, and of the use of connectionless servers on the network performance is also supported. With CLASS, the network synthetic workload can be modeled choosing from a variety of traffic generators ranging from simple Poisson traffic sources to sources modelling the traffic produced when higher level protocols, like TCP, access the ATM services.

Keywords: ATM networks; B-ISDN; Simulation tool; Performance analysis

1. Introduction

The Asynchronous Transfer Mode (ATM) has been deemed the most versatile and reliable transport technology for the Broadband Integrated Services Digital Network (B-ISDN). The wide variety of services and the high quality of service (QoS) that will be provided by B-ISDN pose quite a high demand on the underlying transport network. In particular, the data traffic based on message exchange (often called connectionless traffic) requires a very low cell loss probability, while other services, ranging from the enhanced versions of standard telephony to video conference and to the distribution of digital video on demand, require a very low delay jitter in the delivery of cells.

While ATM was originally conceived for the transport of connection-oriented traffic over virtual channels (VCs) and virtual paths (VPs), it is foreseen that the first wave of B-ISDN will be mostly devoted to the interconnection of local and metropolitan area networks (LANs and MANs), whose

* Corresponding author.

SSDI 0166-5316(95)00011-9

services at the lower layers of the protocol pile are connectionless; a clear understanding of the techniques for the interworking between ATM on one side and LANs and MANs on the other side is thus needed for the design of the first ATM networks.

In general, the dynamics of ATM networks are far too complex to be tackled with an analytical approach. As a consequence, simulation tools are necessary to assist the design and development of B-ISDN. Simulation tools for the performance analysis of data services in ATM networks are in particular needed for the design of the first experimental B-ISDN deployments.

In this paper we describe an efficient, versatile and extensible software tool for the analysis and design of ATM networks. The tool is named CLASS for ConnectionLess ATM Services Simulator. The aim of the tool is the performance analysis of ATM networks from the point of view of several different QoS parameters, such as cell and message loss probabilities and cell and message delay jitters, with or without connectionless servers [1,2].

The performance indices are measured adopting two different points of view. Results referring to the message transfer are collected, thus adopting the viewpoint of the end-user of the data services. Results referring to the utilization of the resources, and to traffic control are collected, thus adopting the viewpoint of the network manager. The ATM network design can thus be tackled from both viewpoints.

CLASS allows the specification of almost arbitrary network topologies, with a great freedom in the choice of parameters such as buffer sizes, link data rates, and so on. Moreover, the ATM network synthetic workload can be modeled choosing from a variety of sources ranging from simple Poisson traffic generators, to sources that mimic the traffic produced by the ATM Adaptation Layer (AAL) when higher level protocols, like TCP, invoke the AAL primitives to access the ATM services. CLASS also allows the investigation of the impact of shaping and policing techniques on the network performance.

CLASS was designed so as to be easily extensible: new functionalities can be simply integrated in the tool by a user, to produce a customized version of CLASS targeted to some specific goal.

CLASS is not a commercial software product for users with limited experience, rather it is a research tool whose users are expected to be experienced telecommunications engineers with a significant background in programming. For this reason, many differences exist between CLASS and existing commercial software tools for the simulation of telecommunication networks, such as Bones [3] and Opnet [4]. The most evident differences are that (i) the I/O interface of CLASS is based upon simple ASCII files, not upon a sophisticated graphical interface; furthermore, the model specification in CLASS is based on a formal grammar that allows checking the model consistency; (ii) CLASS is entirely written in standard C language, paying extreme attention to the efficiency of the simulator, and obtaining quite good portability with no need for different software versions; (iii) the addition of new features in CLASS requires the development of segments of C code and their integration within the existing software.

Also, the tool is very specifically tailored to ATM networks, thus significant differences exist with other general purpose research tools, like Ptolemy [5].

The choices made in the design of CLASS seem to reflect the current trends in the design of software tools to be used for research in the field of ATM networks (see, for example, [6]).

The rest of the paper is organized as follows. Section 2 describes the modelling approach adopted in CLASS, providing some details on the network and workload models. This section also describes the performance indices measured with CLASS. Section 3 presents the architecture of the simulation

tool from the software point of view. The simulator components are described from a functional point of view, and some results are also presented concerning the implementation goals reached with CLASS. Since the paper is mainly devoted to the description of the features of the tool, not to the performance results obtained using the tool itself, we only briefly present in Section 4 an example of the application of CLASS, considering the simulation of a possible Italian ATM network. The interested reader can find simulation results obtained using CLASS in [7–10]. Section 5 closes the paper with some concluding remarks.

2. The modelling approach in CLASS

The generation of a functional model of the system under investigation is the focal point in any simulation study; once generated, the functional model must be converted into a simulation program where all the functional units are correctly represented, together with their interactions. The characteristics of the chosen simulation tool must match the level of abstraction in the system representation of the functional model, so that the behaviors of interest can be observed, and the desired performance parameters can be measured.

This is specially true for complex systems with layered architectures, such as telecommunication networks, where many different levels of abstraction are interesting: data networks can be simulated at the transmission level (modelling the flows of signals and bits), at the packet level (modelling the flows of cells, packets and messages), at the connection level (modelling the connection generation, acceptance and termination), at the service level (modelling the interactions for the service provision).

For this reason, we begin our illustration of CLASS with a general description of the modelling approach that was followed in the development of the simulation tool. The simulator software architecture will be discussed in Section 3.

CLASS allows the development and the simulation of functional models of ATM networks at the packet level: it provides tools for the simulation of the flows of cells and messages in the network. The functional model of a B-ISDN in CLASS is based on an architecture comprising two levels:

(1) The upper level is devoted to the description of the higher layers of the user protocol pile, including the ATM Adaptation Layer (AAL), and transport protocols, TCP in particular. This level also comprises the description of the network workload, i.e. the models of the behavior of users in their access to the communication services provided by B-ISDN. Moreover, the special functions for the provision of connectionless services implemented by connectionless servers are also modeled in this level. Thus, this level models the flow of packets and/or messages.
(2) The lower level describes the ATM and physical layers of the user protocol stack; thus, it models the flow of ATM cells over VCs and VPs, and inside ATM switches.

In the following subsections, we present first the workload models (Section 2.1), i.e. the available traffic models presently supported by CLASS; then, in Section 2.2, we describe the functional models of network elements, such as nodes, links, buffers, and connectionless servers. The traffic control functions that can be used to smooth the flow of traffic in the network are described in Section 2.3; finally, the performance parameters computed both at the cell and at the message level are presented in Section 2.4.

2.1. The workload models

The wide variety of possible application scenarios in B-ISDN translates into the need for different traffic models, so as to adequately describe different characteristic traffic patterns. The traffic models available in CLASS can be grouped into three categories:

- *cell generators*, that produce an unstructured flow of cells directed toward a single user, describing, for instance, a video transmission;
- *message generators*, that produce messages that are then segmented into cells transferred through the ATM network;
- *complex source models*, that produce complex traffic patterns, such as models of DQDB MANs or TCP connections.

2.1.1. Cell generators

These generators can produce two different traffic patterns: Constant Bit Rate (CBR) traffic, or bursty (On-Off) traffic. The former models the generation of cells at fixed time intervals. The latter models the generation of cells at fixed intervals during the On period only, the On and Off periods being geometrically distributed. These sources generate a continuous stream of cells (i.e., cells are not grouped into messages), with the same destination address (defined at the beginning of the simulation) throughout the whole simulation. The main goal of these source models is to create a background traffic for the analysis of the services of interest.

2.1.2. Message generators

The generators belonging to this category produce user messages that are subsequently segmented by the AAL sublayer into batches of cells that must be transferred by the network. All the source models in this category generate a workload whose average level and whose destination are controlled by a traffic matrix that allows the specification of each individual traffic relation. The destination is randomly selected on a message basis, following the probability distribution specified in the traffic matrix.

Several simple generators that produce messages according to a Poisson arrival process belong to this category. The length of the messages can be chosen from three possible distributions: constant, bimodal (i.e., two different message lengths), or truncated geometric. The constant message length is typical of a file transfer type of traffic, in which, almost regardless of the specific higher level protocol used, all the packets are of the same length, normally the maximum allowed message length. The bimodal distribution reflects the observation that most of the traffic in LANs (statistics were gathered for Ethernet and FDDI) is mainly due either to relatively short packets (telnet-like application) or to long packets (ftp-like applications). The geometric distribution was considered because of its nice mathematical properties that allow in some simple cases the validation of the simulator against analytical results.

These message generators can be used as abstract models that represent the aggregate traffic injected by a group of users into the ATM network.

2.1.3. Complex source models

The cell generators and the message generators with Poisson arrival process can be very helpful in understanding and interpreting the network operations, but may not provide a realistic description of the behavior of the traffic arriving at an ATM network from a tributary network with cell based transport, such as a DQDB MAN.

A more sophisticated traffic source model was implemented for the particular case of a DQDB tributary network [11]. The interest in DQDB is due to the fact that when the MAN/B-ISDN gateway does not reassemble packets, the DQDB segmentation process and MAC protocol can significantly modify the traffic pattern at the MAN–ATM network interface (the same cannot be said for Ethernet and FDDI LANs where user data are not segmented into cells before reaching the B-ISDN entry point).

Also, in order to be able to study the performance of the most widely used transport protocol for data application, a specific model to describe the behavior of TCP connections was developed.

Both the traffic model for a DQDB tributary network, called MAN Source Model (MSM), and the model of the traffic generated by the TCP transport protocol are described in separate papers [8,9,12]. We provide below a very concise description of such traffic sources.

MSM. This generator implements a simplified functional model of the DQDB MAC protocol behavior based on a simple queuing system.

MSM is based upon a set of queues, called *miniusers*, and a scheduler that defines which of the queues must be served in the present slot.

The miniusers generate messages that can be labeled either "external" or "internal". External messages represent the traffic effectively offered to the ATM network, while internal messages represent the messages between DQDB users that are not injected into the ATM network.

For the sake of simplicity, it is supposed that the gateway is located at one of the head-ends, and it is assumed that a queue with higher order corresponds to a downstream user.

The basic idea underlying MSM is that in a DQDB MAN, under certain conditions, a transmitting user can block the access of downstream users to the transmission medium. Thus, in the model the service policy always follows a round robin schedule, except in the case when queue i is being served alone, a message arrives at a queue $j > i$ (corresponding to a downstream user) and the round-trip delay between the two users is greater than the mean message length; in this case queue i is served exhaustively, and queue j is served only when queue i is empty. If a message arrives at a third queue, the round robin discipline is immediately resumed.

The results obtained with MSM are generally accurate and robust in the representation of the distribution of the interarrival times of the cells and of the distribution of the number of messages being transferred from the MAN to the ATM network; the MSM validation considers a wide range of numbers of users, network spans, message length distributions and levels of traffic in the MAN [12]. The simulation time required to run MSM is dramatically shorter than the time required for a detailed simulation of a DQDB network.

The TCP source model. The TCP protocol is suitable for the transport of different traffic patterns; higher level services like *ftp* file transfers and *telnet* remote connections are two examples of possible services that will be carried over ATM networks. However, in the currently available release of

CLASS, only a bulk data traffic model has been introduced: an *ftp* connection has a very long (virtually infinite) stream of data to send, and tries to transmit maximum size segments as fast as allowed by the window control mechanism.

The approach adopted in CLASS to obtain a model for the TCP protocol consists in an adaptation of the officially distributed C code of the BSD 4.3-reno release, whose main features include the slow start algorithm, the fast retransmission procedure, and the delayed acknowledgment option. Since the currently available BSD 4.3-reno TCP release is not tailored to high-speed networks [13], slight modifications were required to adapt its code to the ATM environment; the most important aspects of this adaptation process are described in [7–9].

For details on simulations of TCP connections in high bandwidth-delay product networks, see also [7–9,14,15].

2.2. The network model

In CLASS, the functional model of a telecommunication network comprises the cell switching and transfer functions. The model is thus mainly composed of instances of three basic entities:

- the *nodes* that perform the cell switching functions;
- the *links* that perform the cell transmission function, transferring cells from one node to another;
- the *buffers* that allow the storage of cells within the nodes when they cannot be immediately transmitted due to contention on output links; each buffer is associated with one output link.

The model of a generic network can be composed of an arbitrary number of nodes that are connected to one another by links. The number of links per node is arbitrary, with the restriction that all links must be bidirectional, i.e., if the model contains a link originating from node i and reaching node j, it must also contain an identical link originating from node j and reaching node i.

Only the FIFO queuing discipline is presently considered in the node output buffers; more sophisticated queuing and scheduling techniques, such as weighted fair queuing (WFQ) [16], will be implemented in the near future.

One of the possibilities being considered for the management of connectionless traffic in ATM networks is based on the use of specialized functions located in special network objects, called connectionless servers (CLSs). The CLSs operate on messages generated by the users of connectionless services.

The main reason for the use of CLSs is the reduction of the overhead due to the allocation of VCs and VPs. The most important functions of a CLS include reading the destination address of the message in the BOM (Begin Of Message) cell, making a routing decision, (this may require a considerable amount of time), and finally queuing all the cells of the message into the appropriate output queue. These functions are typical of a network layer protocol, thus the CLS functions cannot be implemented within a standard ATM node that operates at the ATM layer, as described in [17].

2.3. The traffic control functions

A large amount of bursty traffic deriving from the segmentation of connectionless messages for transmission over ATM networks may adversely affect the QoS provided by the network. Thus, the

tight QoS requirements of connectionless services are difficult to meet without traffic control and bandwidth management functions.

Three main traffic control functions can be identified in ATM networks:

Connection admission control: The Connection Admission Control (CAC) functions operate at the signalling level: when a request for the setup of a VC is submitted to the network, the VC is established only if enough resources are available.

Traffic shaping: The term *shaping* is used to address the traffic control functions adapting the traffic that a given user injects into the network to the parameters that were negotiated between the user and the network at connection setup. This function is performed at the edge of the network, generally by the user itself or by a traffic concentrator. From an architectural point of view this function is not part of the ATM layer. For this reason the shaping functions are taken into consideration at the upper level of the CLASS architecture.

Traffic policing: The term *policing* is used to address the traffic control functions devoted to the enforcement of the parameters that were negotiated between the user and the network at connection setup. The policing function thus verifies that the traffic produced by a given user is compliant with the negotiated parameters. This open-loop traffic control function is performed within the nodes of the ATM network, on the traffic arriving from the input links.

Among the three described traffic control functions, only the traffic shaping and the traffic policing functions are available in CLASS. The CAC function is not implemented in CLASS, because it refers to models constructed at the connection level rather than at the packet level. In fact, the VC dynamics is relevant on a time scale that is orders of magnitude longer than the one used in CLASS; we believe that the goal of obtaining an efficient and fast simulation tool is not easily addressable when we must deal with phenomena relevant on so different time-scales. For this reason, only Usage Parameter Control (UPC) functions (shaping and policing) are available in CLASS.

UPC, both preventive and repressive, can be implemented with any algorithm that allows an automatic control of one or more characteristics of the traffic: average bandwidth, peak bandwidth, burstiness, etc. Many UPC algorithms were proposed in the literature [18–20]; the UPC policy adopted in CLASS is the Generic Cell Rate Algorithm (GCRA) specified in the ITU-T Recommendation I.371 [21] and ATM Forum ATM User-Network Interface Specification Version 3.0 [22] as a traffic policing mechanism; the traffic shaping function was obtained with an adaptation of the GCRA that delays cells instead of discarding them.

The GCRA is described in the flow chart of Fig. 1; it depends on two key parameters, specified in the traffic contract between the user and the network:

- the bandwidth allocated to the group of VCs on which the algorithm operates; this parameter is specified through its inverse T, the nominal cell interarrival time;
- the allowed cell burst length; this parameter defines the cell delay variation tolerance which is denoted by τ.

The goal of the GCRA is to check if the cell arrival pattern satisfies the constraint fixed in the traffic contract via the parameters T and τ. When the GCRA is used by the policing function, which is a repressive function whose goal is "checking" the traffic pattern, the cells found not to be compliant are discarded. Marking non compliant cells is an option which is under development and will be available in the next release of CLASS. Traffic shaping instead, is a preventive function implemented

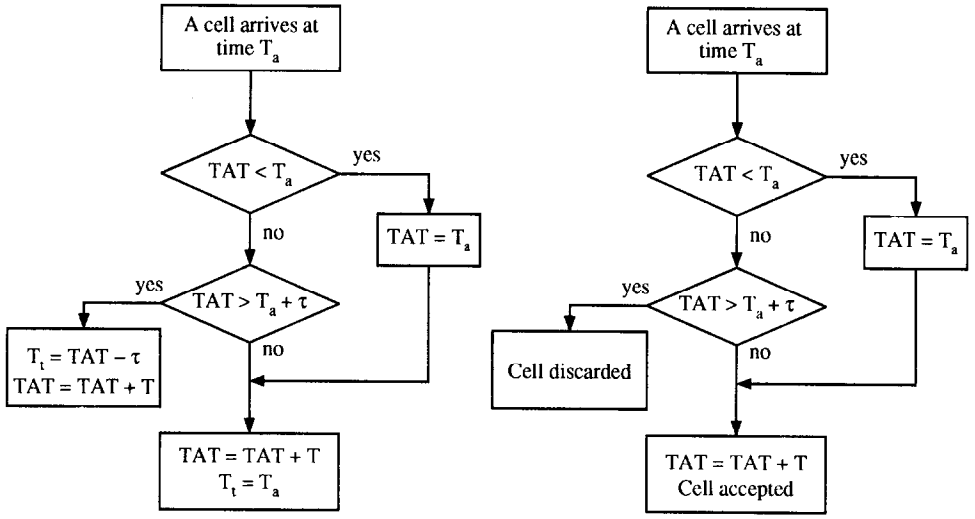

Fig. 1. Flow chart of the GCRA for shaping (left) and for policing (right).

by the user (or by the network access equipment) and has the role of *modifying* the traffic pattern. As a consequence, the GCRA in this latter case is used to compute the time at which the cell is eligible for transmission.

The output of the GCRA is a flow of cells compliant with the characteristics determined by the parameters T and τ.

2.3.1. Shaping

The GCRA defined by ITU-T specifies a control algorithm on a single flow of cells. The architecture of a real shaper [23] however is much more complex, since, in general, a great number of flows of cells must be handled. The shaping function is generally performed at the user premises in order to ensure that the traffic crossing the user-network interface (UNI) meets the characteristics specified in the traffic contract at connection setup.

In order to understand the architecture of the shaper used in CLASS, it must be kept in mind that an ATM user is generally a traffic concentrator that handles a great number of VCs coming from different traffic sources; even in the case of a single traffic source (e.g. a mainframe or workstation) the number of connections with different characteristics that are open at the same time can be very high.

Since no standard shaper structure has yet been defined, we chose a general architecture that can be tailored by the user of CLASS through a small set of parameters.

Figure 2 depicts the architecture of the shaping device implemented in CLASS. A single shaper has several elements implementing the GCRA, (in the figure N GCRA elements are depicted), each one operating on a *group* of VCs. The message generation corresponds to the invocation of an AAL primitive from a higher level protocol. For instance, if the shaper is placed in a LAN gateway, a message generation can correspond to the arrival at the gateway of a LAN packet that must be forwarded to B-ISDN.

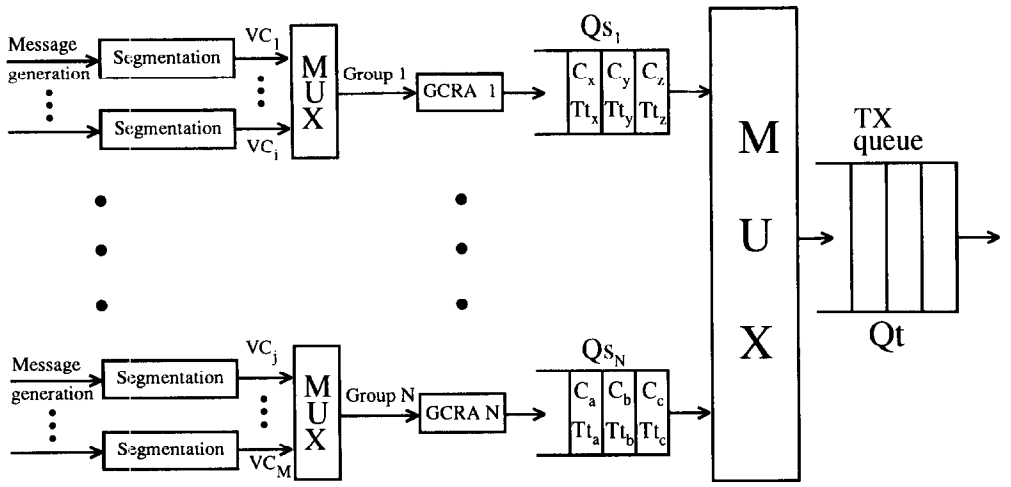

Fig. 2. Architecture for traffic shaping.

The first stage that multiplexes VCs into groups needs some explanation. Since no standard or recommendation specifies that the shaping of traffic must be performed on a VC basis, the user can group together VCs in order to better exploit the characteristics of the network. For instance, the shaping can be performed on a VP basis instead of a VC basis, or can be performed together on all the VCs coming from the same source, or going to the same destination. At the two extremes of the options available to the user of CLASS, the shaper can either operate on each VC separately or on the whole traffic. After the grouping stage, the cell flows are forwarded to the elements implementing the GCRA; the cells are inserted into FIFO queues with a time stamp indicating the instant of time when they are eligible for transmission (Tt_i in the figure). Several grouping policies have been studied with CLASS and their comparison is reported in [10].

A typical shaper however has still to cope with a second problem: potentially many flows of shaped cells have to be sent over the same transmission link. A multiplexing stage is thus needed. Adding a multiplexing stage after the traffic has been shaped is a fairly critical issue, since it can nullify much of the effort done in smoothing the traffic characteristics; the burstiness of the single flow could again increase significantly. This problem has been recognized in general by ITU-T for any multiplexing stage, but no solution has been envisaged [21]. In CLASS, the multiplexer polls all the queues Qs_i in the same order in each slot, taking the head of the line cell, provided that the transmission time for the cell has been reached. This is a straightforward stage of multiplexing, but it minimizes the burstiness increase if the FIFO order has to be maintained. More sophisticated multiplexing strategies not respecting the FIFO strategy can be devised, and future releases of CLASS will include one or more of them.

It is important to emphasize the fact that the user of CLASS has complete control over the parameters of each shaping device. By appropriately selecting such parameters, the CLASS user can decide what VCs should be grouped before shaping (or decide to individually shape each VC) and select the parameters of each GCRA algorithm.

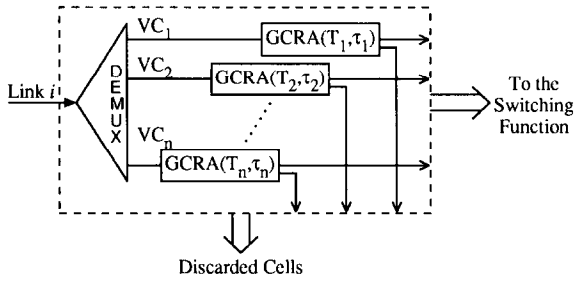

Fig. 3. Architecture for traffic policing.

2.3.2. *Policing*

As we noted before, traffic policing ensures that the parameters of a given traffic flow comply with the values that were negotiated between the user and the network at connection setup. Traffic policing is a *repressive* measure and has the role of discarding the cells whose characteristics may be dangerous for the network operations (source of congestion and/or buffer overflow), or simply do not comply with the traffic contract.

The architecture of the traffic policing device modeled in CLASS is depicted in Fig. 3. The applied policing algorithm (shown in Fig. 1) is the GCRA, where cells found to be non-compliant are discarded. In the next release of CLASS the option of marking cells as low priority will also be given.

The traffic policing is performed within the ATM network nodes, and can be enforced on any node input link.

2.4. *Estimated performance parameters*

The estimation of the performance of connectionless services requires the computation of performance parameters both at the cell and at the message levels. These parameters are estimated considering both the whole network, and individual buffers and VCs.

The performance parameters computed with CLASS can be divided into three categories, as follows:

Cell and message loss probabilities: Loss probabilities can be subdivided into three different contributions: the first is due to overflows in the user transmission buffers (the whole message is discarded in this case); the second is due to overflows in the buffers associated with links between network nodes; the third is due to overflows in the buffers associated with the links that connect the destination node to the final destination (a user). A message is considered to be lost when at least one of its cells is lost. Cells can also be discarded within an ATM network because of the traffic policing function; these cell loss events are separately recorded.

Cell and message delay jitters: Cell and message delays comprise a constant part, due to propagation delays along links, and processing delays within nodes, and a variable part, due to the waiting time in the user transmission buffer, and to queuing in the buffers associated with links between nodes. Only the variable part is considered and measured. For each delay jitter, the histogram, the average value, and the variance are computed.

Number of useless cells brought to destination: Since the loss of only one cell causes the loss

of the entire message to which the cell belongs, all the other cells of the same message that successfully reach their destination result in a useless effort by the network. It is important to count these useless cells, since their number can be greatly affected by different traffic control policies. We call *useless load* the fraction of useless cells, i.e., of cells belonging to lost messages, that are brought to destination.

As pointed out above, results are collected separately for different sections of the network. There are two different reasons for keeping separate the contributions from different network sections. The first concerns the identification of bottlenecks and pathological conditions within the network. The second, instead, concerns the difference between the internal elements of the network and the elements at the border of the network. The former are for example the links connecting two nodes, and the latter are for example the links connecting a node and a user. This separation is useful because the phenomena observed within the network and at the UNI quite often have a different origin, and may require different actions. For instance, overflows within a user transmission buffer may be solved with a looser shaping of the input traffic, while congestion on a node-to-node link probably calls for more strict traffic shaping or policing, or a faster link.

3. The software architecture of CLASS

We now illustrate in some detail the implementation of CLASS, describing the main modules of the CLASS software, called *entities*, together with the main features of the tool.

3.1. The main characteristics of the software

CLASS is a slotted, synchronous, time-driven simulator. CLASS is entirely written in standard C language, which makes it a highly portable simulation tool. The present version of the software was installed on several different architectures: VAX/VMS, OpenVMS/AXP, DEC Ultrix, Next, HPUX, AIX IBM, 32 bit MS-DOS, with no need for software adaptation.

The choice of the slotted simulation approach is dictated by efficiency requirements, and it is justified by the fact that the data units within the network (the cells) have constant size and by the interest in performance indices related to user messages and cells, not to smaller units of data.

The shortest time interval that is handled by the simulator, thus defining the simulation time unit, is the slot duration on the fastest link, or, more precisely, the maximum common divisor of the slot durations on all the links in the network.

In order to manage different link speeds, in CLASS each link has an attribute, called `timescale`, that is an integer number that identifies the link data rate: the slots on the link are `timescale` times longer than the time unit in the network. This is not a serious limitation in ATM networks, that are expected to employ digital links obtained from an SDH transmission facility, where data rates are multiples of one another.

The time-driven simulation technique is best suited for the simulation of systems where each component is involved in some operation at any time instant, while systems whose components are idle most of the time are in general more efficiently simulated with the event-driven technique. ATM

networks belong to the first category when heavily loaded, while they may belong to the second when lightly loaded.

In general, the simulation of lightly loaded systems is not very expensive in terms of memory and CPU time requirements, because of both the small number of events, and the small variability of the samples collected for the estimation of the performance indices of interest. Instead, the simulation of heavily loaded systems may be extremely expensive in terms of memory and CPU time requirements because of the very large number of events, but mostly due to the fact that the samples collected for the estimation of the performance indices of interest typically exhibit quite a remarkable variability. Hence, if we aim at a constant accuracy of the performance estimates, the simulation cost grows much more than linearly with the load. On the other hand, quite often intuition is sufficient to guess that lightly loaded communication networks can provide good QoS; on the contrary, the assessment and the choice among alternate configurations for communication networks under heavy load requires a careful quantitative estimation of the network performance in order to guarantee that the required QoS are met.

The choice of the time driven approach in CLASS was due to the above considerations, and to the search for efficiency in the design of CLASS.

3.2. The main functions

Five main entities can be identified in CLASS, namely MANAGER, USER, NODE, CLS and STATISTIC. The first four are described in some detail below while the fifth is described in Subsection 3.3. In the rest of the paper, when referring to the *entities* of CLASS, small capitals will be used (e.g. USER, NODE, MANAGER) to distinguish these entities from the generic objects with the same name.

MANAGER: The MANAGER entity provides some of the functions pertaining to network administration and management. Typical functions are the definition of the VC routing within the network. VCs are opened at the beginning of the simulation, and remain open until the end of the simulation experiment, that is, no CAC is implemented by the MANAGER. The MANAGER in CLASS has a complete view of the network, thus avoiding the necessity of implementing signalling functions.

USER: The USER entity in CLASS provides the functionalities of traffic generation and sink. Several types of USERs are available in CLASS, implementing the traffic sources described in Subsection 2.1. The USERs in CLASS always represent an aggregation of end users, connected to either a MAN or a LAN, or, in more abstract terms, a collection of traffic generators with the same characteristics. When a USER receives a cell, it reads from it the information needed for the estimation of the performance parameters, and then places it into a free-list. The USERs in CLASS also provide, upon request, the shaping functions described in Subsection 2.3.1.

NODE: The NODE entities perform the key internal operations characterizing the behavior of an ATM network. Notice that they are the only entities involved in this task, since in CLASS links and buffers are completely passive (they are simulated as linked lists). The NODEs route and switch cells from one link to the next; in other words they "move" the cells through the network, following, for each VC, the path assigned by the MANAGER during the setup of the simulation.

The NODEs execute a fairly simple algorithm: they inspect input cells, look up a routing table in which, for each VC, the output link where the cell must be routed was written by the MANAGER, and queue cells for transmission on the output link. This means that in CLASS we do not consider the internal structure of the switching fabric, or, equivalently, that the internal node structure is non-blocking with output queuing.

The simplicity and efficiency of the simulation of this task is of paramount importance, because this is the most frequently performed operation during the execution of a simulation run: the number of times this operation is executed is roughly equal to the number of simulated cells multiplied by the mean number of nodes crossed by each VC.

The switching operation itself consists in moving the pointer to the cell from the linked list representing the input transmission link to the linked list representing the output buffer and link. When links with different data rates exist, this operation may be necessary only for few links during each simulation step: testing all the links of each node during each simulation step leads to a significant waste of resources.

To avoid this waste, CLASS builds a *minimum scheduling sequence* that represents the shortest periodic sequence of NODE invocations, so that the only test that must be performed when a NODE is activated is the presence of cells at the input ports. The NODEs are invoked, during the minimum scheduling sequence, a number of times that is proportional to the inverse of the minimum `timescale` of their links. The slots of the sequence are separated by inserting a *timing event*, that represents the end of a simulation step.

Figure 4 shows an example of a minimum scheduling sequence in a network with links with three different data rates. For the sake of simplicity, the network comprises only four NODEs, and no USERs are shown. The data rate on two links is taken to be 200 Mbit/s, while on the two other links it is assumed to be 300 Mbit/s, and 600 Mbit/s, respectively. The value of the parameter `timescale` for the fastest link is by definition set equal to 1. The values of the parameter `timescale` for the remaining links are computed by dividing the highest data rate in the network by the link data rate; thus, the link with data rate 300 Mbit/s has `timescale` = 2, and the two links with data rate 200 Mbit/s have `timescale` = 3.

The periodicity of the minimum scheduling sequence equals the minimum common multiplier of the link `timescales`; each NODE appears in the minimum scheduling sequence each time the `timescale` of one of its links divides the index of the slot within the sequence. The first slot within the minimum scheduling sequence is labeled "0" and every node is invoked in this slot.

It is easy to verify in the example that starting from slot number 6 (the number of the slot is reported below the minimum scheduling sequence) the scheduling of entities is repeated periodically.

The NODE entities also provide the policing functions described in Subsection 2.3.2 on the links where policing is required.

CLS: The CLS entities in CLASS implement the following algorithm. When the BOM cell of a new message is received, a FIFO queue is created for the new message, and all the incoming cells of that message are stored in the queue. The CLS first computes the routing of the message and then begins to forward the message cells towards their destination, that can be either the final user or another CLS in the network. In CLASS the computation of the routing information requires a fixed amount of time, of the order of a few hundred slots, since the delay is often made constant in a

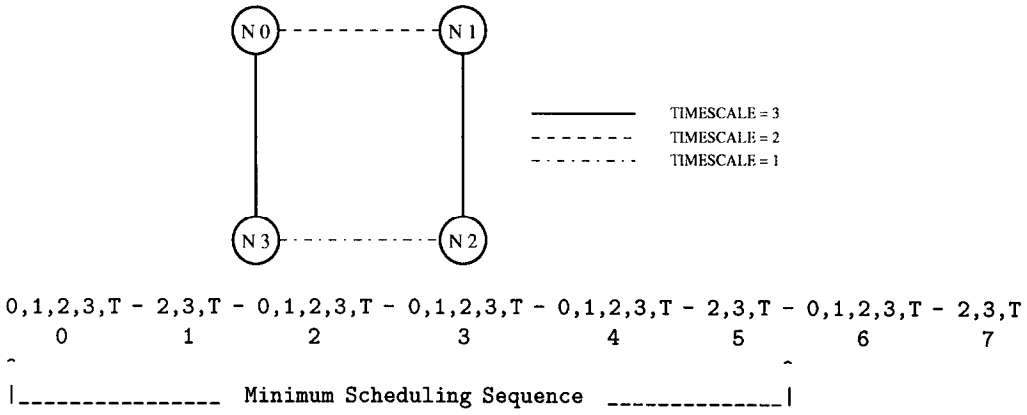

Fig. 4. Scheduling sequence for a network with different link data rates; T is the timing event.

CLS (as stated in [24]), in order to upper bound the real time taken for the routing decision. The reason is to avoid that messages between the same source and destination arrive in the wrong order, a condition that may lead to undesired retransmission requests by higher level protocols. Another reason for setting a fixed delay in CLSs is due to the SMDS (Switched Multi-megabit Data Services [25]) requirements for which a maximum delay of 7 ms is allowed: if a message cannot be routed within that time, the message is discarded. This feature, however, was not implemented in CLASS since the routing information is always available. The CLS in CLASS operates without reassembling messages, forwarding each cell with the appropriate delay.

3.3. Statistical analysis routines

The CLASS simulation experiments are automatically stopped when the desired user-specified accuracy is reached on a selected performance parameter. The entity STATISTIC automatically determines the duration of the initial simulation transient period, collects the measured data, computes aggregate performance figures, and provides an estimate of the accuracy reached by the results, using the "batch means" technique, implemented following the guidelines in [26].

The main statistical parameters available to the user are the following:

- confidence level, i.e. the probability with which the point estimate falls within the confidence interval;
- accuracy, i.e. the relative half width of the confidence interval;
- maximum number of samples to be collected during the simulation experiment;
- batch size, i.e. the size of the batches of samples to be used in the statistical analysis;
- minimum length of the transient period, measured in number of samples;
- transient accuracy, i.e. the relative half width of the interval that must contain the most recent transient batch means in order to declare the end of the initial transient period;
- transient batch size, i.e. the size of the batches of samples to be used for the test about the end of the initial transient period;
- transient sequence length, i.e. the number of most recent batch means that must satisfy the transient accuracy in order to declare the end of the initial transient period.

Note that, even if the simulation stops due to the collection of the maximum number of samples, the confidence intervals and accuracies of all parameters under observation are computed and presented to the user. This can be extremely helpful for example when no cell losses were observed in very long simulation runs, since the higher end of the confidence interval can provide a probabilistic upper bound to the cell loss probability even for these cases.

The use of variance reduction techniques in CLASS will be investigated in the future, although it seems very difficult to apply standard techniques, such as importance sampling, to scenarios such as those resulting from the simulation of complex traffic sources, like TCP or MSM. If simpler source models can be sufficient for the study, then the application of importance sampling to the bias of the input processes, as proposed in [27], may lead to phenomenal savings of CPU time.

3.4. The user interface

CLASS is a telecommunication research tool, not a commercial software product. For this reason, the I/O interface of CLASS is based upon simple ASCII files, not upon sophisticated graphical interfaces.

The input interface is based upon a formal grammar that allows the topological description of the network. The NODEs of the network are described together with their links leading to neighboring NODEs. The workload of the network is specified by defining an arbitrary number of USERs with their characteristics and their location (the NODE to which they are connected). USERs with different traffic generation models can be mixed within the same network. The traffic pattern of each user can be either chosen to be uniformly distributed among all the NODEs of the network (the default option), or specified with the definition of a traffic matrix assigning a given portion of the global workload of the network to each traffic relation within the network; a traffic relation can define the traffic going from any NODE i to any NODE j, or from any USER i to any USER j, with $j \neq i$.

The output interface is again based upon ASCII files where the aggregate results are recorded, together with some relevant events of the simulation. The files are written in a format that makes their post-processing easy with standard tools for the production of drawings.

3.5. Code characteristics and performances

The key elements for the acceptance and success of a software simulator are ease of use, and efficiency, in terms of both CPU time and memory requirements necessary to run reasonable simulation experiments. Large memory requirements are quite often a primary cause of the growth of the CPU time, since programs requiring large amounts of memory waste a lot of time swapping in and out of the main memory.

The development of CLASS resulted in a simulation tool that is quite efficient in terms of both memory and CPU requirements.

Table 1 reports the memory requirements of CLASS, dividing them into code and data requirements. The values reported in the table refer to a single instantiation of each entity or data structure; USERs require different amounts of memory depending on the underlying generation model: the required memory space grows starting from the simplest models up to the more complex MSM and TCP models. The global data structure requirements grow as described in the table, where n is the number of NODEs in the network, and k is the number of USERs. A CLS also requires a small amount of

Table 1
Memory requirements in bytes for CLASS entities and data structures (n= number of nodes, k = number of users, x = number of messages in the CLS)

CODE	NODE	USER		CLS	Link	Cell	Global data structures
		Min.	Max.				
150000	380	108	408	$304+32x$	90	40	$56n^2 + 80k^2 + 5nk^2$

memory (32 bytes) for each message that is being processed. The maximum number of messages processed at the same time is unknown, since it depends on the network and traffic characteristics, and is indicated with x in the table.

As an example of the effective memory required to run a simulation, consider the application example reported in Section 4: the memory space used by the simulation runs for that example never exceeds 260 kbytes, 150 for the code and 90 for data.

In order to improve the CPU efficiency characteristics of CLASS, particular attention was paid to avoid time wasted in function calls due to excessive stack operations and to avoid test replication. The construction of the minimum scheduling sequence described in Section 3.2 is a typical example of such efforts.

Another optimization was obtained in the test for the links enabling. Usually a link is enabled, i.e. the data on it are valid, when the current simulation time is a multiple of the link `timescale`. The natural way to test the enabling is to perform a MOD operation each time a link is polled. With this solution a large amount of CPU time (up to 20% of the total in some cases!) was spent in this test, because the MOD function involves at least a division every time it is executed. In order to reduce this annoying waste of CPU time we adopted the following strategy. An *enabled*[i] array is built, where i can assume any of the allowed `timescale` values, that contains the remainder of the division between the current time and i. This array is updated once at the beginning of each simulation step, together with the timing event. The link enabling is then checked with a simple test on the value of *enabled*[*timescale*]. Finally, also the MOD operation at the beginning of each simulation step is avoided by computing the remainder in the current slot based on its value on the previous slot, with an operation that implies just an increment and a test. With these simple artifices, the CPU time spent in checking the link enabling was reduced to a negligible amount.

The CLASS code was written considering optimization issues typical of modern computer architectures, and the code structure was organized so that any existing pipeline can be exploited. For example, the branches are, when possible, organized so that the condition that is more often true implies no jump in the code. When necessary, branches were completely avoided by duplicating the software and inserting the call to the correct function in the minimum scheduling sequence during the simulation initialization.

Table 2 reports some numerical data about the performance of CLASS in terms of CPU requirements, referring to a DEC 3000/300x AXP workstation running OpenVMS. Given the time-driven architecture of CLASS, the CPU requirements do not depend significantly on the individual simulation run, but depend on the network topology and load. We used two different network topologies for this test:

- a simple topology, shown in Fig. 5, which is symmetric and uniformly loaded;
- a more complex topology shown in Fig. 6, the Italian topology, that is asymmetric, with unbalanced load and contains links with different `timescales`.

Table 2
Number of cells per CPU second transferred from source to destination by CLASS in the reference simulation scenarios

UPC enforced	Simple network		Italian network	
	Heavy load	Light load	Heavy load	Light load
None	19455	16315	12955	10765
Shaping	14550	10985	9210	7210
Shaping + policing	13290	10875	9075	6600

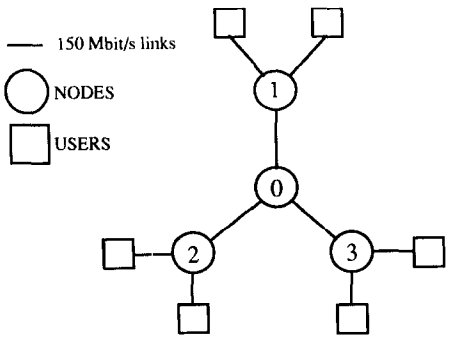

Fig. 5. Topology of the simple network used for the efficiency tests.

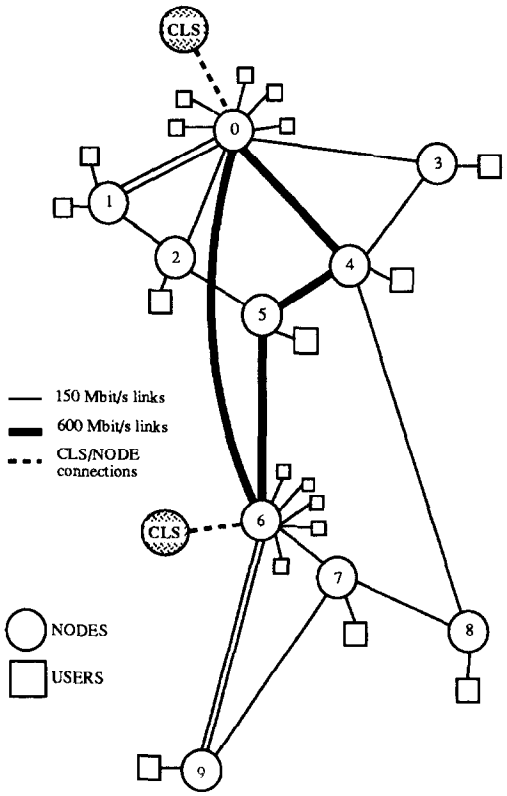

Fig. 6. Topology of the Italian network.

Both topologies were simulated under light and heavy load, and either with or without UPC functions.

For the simple network the light load corresponds to a 50% load on the transmission links between nodes, while the heavy load corresponds to an 80% load of such links.

In the case of the Italian network the light load corresponds to a total traffic equal to 800 Mbit/s, while the heavy load corresponds to a total traffic of 1.4 Gbit/s.

The data in Table 2 show that CLASS performs better with heavily loaded networks, and this was indeed expected. The performance becomes poorer when UPC functions are simulated, and this again is no surprise since UPC functions require a number of operations on each cell. Finally, CLASS is faster with small and homogeneous networks than with large and asymmetric ones, as obvious, but

Table 3
Percentage of CPU time spent in the execution of different code segments

NODE (10 items)	22.5
USER (19 items)	27.5
STATISTIC	6.5
Simulation control	19.5
Cell management	10.0
Miscellaneous	14.0

the interesting thing is that the performance difference is limited to about 30%, while the increase in complexity from the simple to the Italian network is indeed quite large.

What must be emphasized is the overall performance: on the average, CLASS is capable of simulating the transfer of about 10^4 cells per CPU second. This means that the transfer of about 10^7 cells, which allows the estimation of cell loss probabilities in the range of 10^{-6}, requires about a quarter of an hour of CPU time.

In order to investigate the CPU time requirements of the different parts of the simulator, the code was profiled with the aim of trapping and optimizing the critical code sections. The fraction of the CPU time spent in the execution of the different code segments in one of the scenarios for which numerical results are reported in Section 4 is presented in Table 3.5. The time spent in each section is rather evenly distributed, showing a good balance of the code, and hence the absence of significant bottlenecks.

4. An application of CLASS

As an example of the application of CLASS, this Section presents some results obtained with the simulation of a topology suitable for an early deployment of an experimental Italian ATM network. The network topology is shown in Fig. 6. The traffic distribution is highly asymmetric, the network having essentially two hot spots in Rome and Milan (NODEs 6 and 0, respectively). For the sake of simplicity, the simulations were initially run assuming that only connectionless traffic exists in the network; this may be a reasonable scenario for an ATM crossconnect operating as a backbone among MANs and ATM concentrators that collect business traffic. Message lengths are distributed according to a truncated geometric with mean 20 cells. All the simulated users are MSM, with the exception of 5 sources: 2 in Milan, 2 in Rome, and one in Turin (node 1), which are simple Poisson generators. For MSM, the fraction of outgoing traffic is taken equal to 0.75.

We report results for a network where the buffering capacity at NODEs was set to 100 cells per port, while the USER transmission buffer capacity was set to quite large values in order to avoid losses in these buffers.

As far as the shaping function is concerned, results are presented for three different cases labeled in the figures as follows:

NOS: no shaping is performed, and the user traffic is allowed to enter the network at full speed (150 Mbit/s);

SUG: (source user grouping) the traffic on all the VCs that originate in one user is collectively shaped;

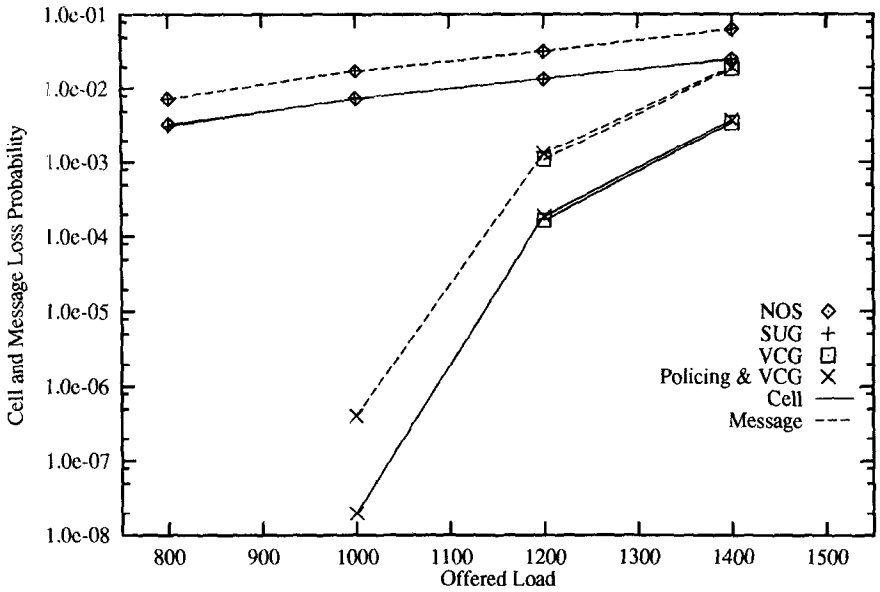

Fig. 7. Cell and message loss probabilities versus the network load, without shaping, with the SUG and VCG shaping policies, and when policing is adopted with the VCG policy.

VCG: (VC grouping) the traffic on each VC is individually shaped.

The shaping function is implemented with the shaper architecture illustrated in Subsection 2.3.1; the algorithm parameters are

$$T_s = \max\left(1, \left\lfloor \frac{C}{2B_w} \right\rfloor\right), \qquad \tau_s = 20$$

(note that we added a subscript s to the parameters to indicate that they refer to the shaping function) where B_w is the average bandwidth of the group of VCs on which the shaper operates, and C is the link data rate.

The policing function, when implemented, follows the architecture illustrated in Subsection 2.3.2; the algorithm parameters are

$$T_p = \max\left(1, \left\lfloor \frac{C}{2B_w} \right\rfloor\right), \qquad \tau_p = 100$$

(here we added a subscript p to the parameters to indicate that they refer to the policing function).

The value of the cell delay variation tolerance (τ) is not equal to the one used in the shaping algorithm because the multiplexing stage at the output of the shaper introduces an increase in the burstiness, so that a policer with the same parameters as the shaper would discard an unreasonable number of cells.

The CLS model introduces a constant delay (equal to 100 cell times) on every cell.

In [8–10,17] we reported many simulation results generated with CLASS. Here we only present a few figures and tables to demonstrate the capabilities of the simulation tool.

Figure 7 shows the cell and message loss probabilities versus the network load without shaping, for the SUG and VCG shaping policies, and when traffic policing is applied at the nodes entering

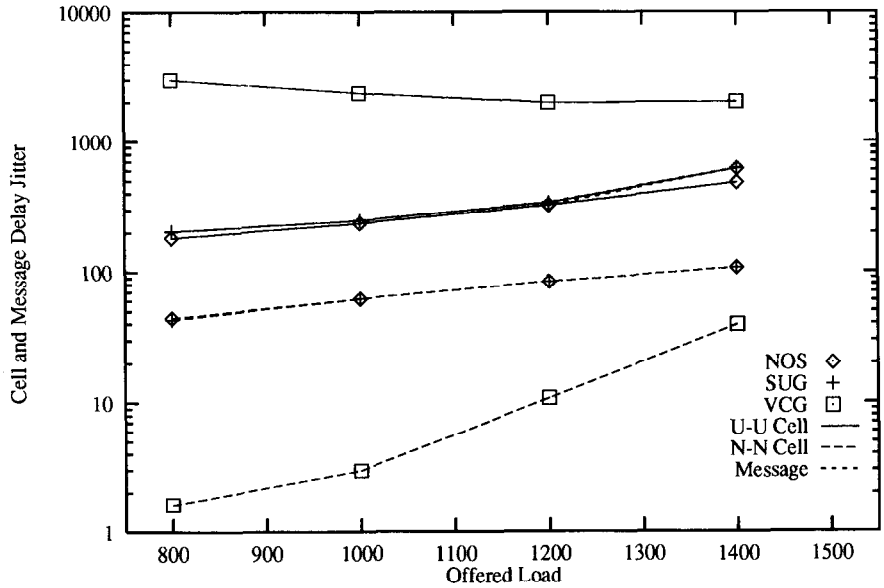

Fig. 8. Average of the user to user cell and message delay jitters and of the cell delay jitter inside the network, versus the network load without shaping and for the two grouping policies considered.

the ATM network, together with a VCG shaping at each user. The curves clearly show that shaping is effective only when separately applied to each VC. The message loss probability is always greater than the cell loss probability, as can be expected since a message is lost even if only one of its cells is lost. However, it must be noted that the ratio between the message loss probability and the cell loss probability is smaller than the average message length (20 cells); this indicates that cells are generally lost in bursts. This ratio decreases with the increase of the cell loss probability and is smaller for NOS and SUG. This is explained by the fact that in these cases the cells of one message are not interleaved at the source with those of other messages.

Considering the results obtained when the policing function is applied, it is interesting to notice that, even if the policing parameters allow a significantly greater cell delay variation tolerance than those of shaping, still some cells are discarded by the policer, thus increasing the cell loss probability. For 1 Gbit/s load, the policers discard few cells (although the absolute number is very small and the result can be considered statistically not very reliable), while no cell losses are recorded for the network without policer.

No losses were recorded during the simulations with VCG for loads lower than 1.2 Gbit/s; the number of simulated cells is $5 \cdot 10^7$, so that the upper bound on the cell loss probability estimated with a confidence level of 95 % is about $6 \cdot 10^{-8}$ (the lower bound is of course 0 — actually, the point estimate is zero, and the confidence interval half width is about $6 \cdot 10^{-8}$). It is worth emphasizing that this remarkably low upper bound was obtained with little more than one hour of CPU time.

Figure 8 shows the average of the user to user (U-U) cell and message delay jitters and of the cell delay jitter inside the network (N-N), versus the network load without shaping and for the two grouping policies considered. In all cases, no policing is considered. U-U cell delay jitters are longer, since they include the time spent in the user buffer. Also in this case the results for NOS and SUG are almost identical. VCG achieves shorter delay jitters inside the network, thanks to the better

Table 4
TCP throughput in Mbit/s and efficiency for a file transfer from Milan to Rome

Background load	Throughput			Efficiency		
	NOS	SUG	VCG	NOS	SUG	VCG
0.8 Gbit/s	104.4	105.4	135.4	0.985	0.985	1.0
1.0 Gbit/s	56.9	52.9	135.4	0.929	0.929	1.0
1.2 Gbit/s	19.4	20.6	71.6	0.780	0.785	0.951

traffic characteristics, but this is paid for with long waits in the user buffers, before admission into the ATM network. It is interesting to notice that the U-U cell delay and the message delay are in any case almost identical. A further interesting behavior is the one of the U-U cell (and message) delay with VCG: it is almost independent of the load of the network, showing a slight decrease with increasing load. This behavior can be explained by the fact that the increasing load does not change the number of VCs in the network, but increments their bandwidth, so that the waiting time in the USER transmission buffer decreases due to the faster service rate of the shaper.

The constraints imposed by the presence of CLSs on the VC routing has quite a negative impact on the load of individual links. The throughput of the network if all the traffic is routed through the CLSs is much lower than in the other case, as was expected since the CLSs represent hot spots in the network. The cell and message loss probabilities are not reported because the network shows a peculiar behavior related to the saturation of the link from NODE 0 to NODE 6 where all the traffic between the two CLSs flows: no cell and message losses are recorded up to a load of about 500 Mbit/s, while, with a load of 600 Mbit/s, the cell loss probability jumps up to 10^{-1} and all the losses are recorded on the above link.

Table 4 reports the throughput and the efficiency for a file transfer from Milan to Rome, using the TCP protocol. These results refer to the network topology shown in Fig. 6, where all users, apart from the TCP source and destination, are simple Poisson generators. Message lengths for the Poisson generators are again distributed according to a truncated geometric with mean 20 cells.

The throughput is measured at the receiver as the total amount of received bits over the simulation time, and the efficiency is defined as the ratio between the throughput at the receiver and the traffic injected into the network by the transmitter. The efficiency is a measure of the amount of traffic that must be retransmitted due to cell losses. The TCP connection has a maximum window size of about 180 kbytes (20 segments), that allows the connection to obtain a reasonable throughput. Results are reported for different values of the background traffic load, which ranges between 0.8 Gbit/s and 1.2 Gbit/s, and for the three traffic shaping conditions. In all cases, no policing is considered. Shaping is performed at all users, with the exception of the source of the TCP connection; thus, the results show the effect of traffic shaping on the background traffic, not on the TCP connection itself.

Once more, it can be observed that VCG is quite effective. With a background load of both 0.8 Gbit/s and 1.0 Gbit/s, with VCG shaping, the TCP connection suffers no cell loss, and reaches the maximum throughput achievable with the chosen maximum window size and propagation delay. When the background load is raised to 1.2 Gbit/s, the throughput of the TCP connection decreases significantly, but it is still considerably higher than with either NOS or SUG. Also the efficiency with VCG is close to 100% in all the scenarios presented. Conversely, the SUG shaping algorithm seems to have little impact on the TCP traffic, since the performance results are almost identical to the case in which no shaping is applied.

5. Conclusions

The main features and characteristics of CLASS, an efficient software tool for the analysis of the quality of connectionless services in ATM networks have been described; CLASS is a time-driven, slotted, synchronous simulator, entirely written in standard C language, allowing the performance analysis of ATM networks adopting both the viewpoint of the end-user of the data services, and the viewpoint of the network manager. Many different performance parameters can be computed with CLASS; the most important are cell and message loss probabilities and cell and message delay jitters. CLASS also allows the investigation of the impact of shaping and policing techniques and of the use of connectionless servers on the network performance.

With CLASS, the network synthetic workload can be modeled choosing from a variety of traffic generators that simulate either simple Poisson traffic generation, or the traffic produced when higher level protocols, like TCP, access the ATM services.

Some example simulation results have been presented in order to give a flavor of the capabilities of the present version of CLASS, which is currently being used for the investigation of the characteristics of possible topologies for experimental Italian and Hungarian ATM networks [10].

Future enhancements of CLASS will include a wider variety of traffic sources and traffic control algorithms, models of IP servers and of *telnet* connections, a more sophisticated VP routing scheme, clever policing schemes with cell marking, sophisticated cell queuing schemes inside nodes, and the possibility of modelling private virtual networks.

Acknowledgements

This work was performed in the framework of a research contract between Politecnico di Torino and CSELT (Centro Studi e Laboratori Telecomunicazioni), and with the support of the European Community through the PHARE-ACCORD contract n. H-9112-0353.

References

[1] ITU-T Recommendation I.211, B-ISDN Service Aspects, Geneve, Switzerland, December 1990.
[2] ITU-T Recommendation I.362, Support of Broadband Connectionless Data Service on B-ISDN, Geneve, Switzerland, June 1992.
[3] Comdisco Systems Inc., BONeS Designer User's Guide, Version 2.5, Foster City, Calif., 1993.
[4] MIL 3 Inc., OPNET Modelling Manual, Release 2.4, Washington, D.C., 1993.
[5] University of California, Berkeley, Department Of Electrical Engineering and Computer Science, The Almagest: Manual for Ptolemy, Version 0.5.1, Berkeley, Calif., 1994.
[6] C. Liu, H.T. Mouftah and M. Sivabalan, A Testbed for Dynamic Routing over ATM Networks, *IEEE Symp. on Planning and Design of Broadband Networks*, Montebello, Canada, October 1994.
[7] A. Bianco, Performance of the TCP Protocol over ATM Networks, *ICCCN'94*, San Francisco, Calif., September 1994.
[8] M. Ajmone Marsan, A. Bianco, R. Lo Cigno and M. Munafò, TCP over ATM: Some Simulation Results, *4th Open Workshop on High Speed Networks*, Brest, France, September 1994, pp. 110–116.
[9] M. Ajmone Marsan, A. Bianco, R. Lo Cigno and M. Munafò, Shaping TCP Traffic in ATM Networks, *IEEE ICT '95*, Bali, April 1995.
[10] M. Ajmone Marsan, T.V. Do, L. Jereb, R. Lo Cigno, R. Pasquali and A. Tonietti, Simulation of Traffic Shaping Algorithms in ATM Networks, *Second Workshop on Performance Modelling and Evaluation of ATM Networks*, Bradford, UK, July 1994, pp. 28/1-28/16.

[11] IEEE P802.6 – Metropolitan Area Networks, Distributed Queue Dual Bus (DQDB) Subnetwork of a Metropolitan Area Network (MAN), *Draft Standard – Version D15*, October 1990.
[12] M. Ajmone Marsan, R. Lo Cigno, M. Munafò and A. Tonietti, A Source Model for Connectionless Traffic in B-ISDN, *5th IFIP Conf. on High Performance Networking*, Grenoble, France, June 1994.
[13] Van Jacobson, R. Braden and D. Borman, TCP Extensions for High Performances, *RFC 1323*, May 1992.
[14] A. Romanow and S. Floyd, Dynamics of TCP Traffic over ATM Networks, *ACM SIGCOMM'94*, London, UK, September 1994.
[15] T.V. Lakshman and U. Madhow, Performance Analysis of Window-Based Flow Control using TCP/IP: The Effect of High Bandwidth-Delay Products and Random Loss, *IFIP Transactions C-26, High Performance Networking V*, (1994) 135–150.
[16] A. Demers, S. Keshav and S. Shenker, Analysis and Simulation of a Fair Queuing Algorithm, *ACM SIGCOMM '89*, Austin, Texas, September 1989.
[17] M. Ajmone Marsan, R. Lo Cigno, M. Munafò and A. Tonietti, Simulation of ATM Computer Networks with CLASS, *7th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, Vienna, Austria, May 1994.
[18] A.E. Eckberg, D.T. Luan and D.M. Lucantoni, Bandwidth Management: A Congestion Control Strategy for Broadband Packet Networks – Characterizing the Throughput-burstiness Filter, *ITC Specialist Seminar*, Adelaide, Australia, September 1989.
[19] A.E. Eckberg, B-ISDN/ATM Traffic and Congestion Control, *IEEE Network Magazine*, **6** (5) (1992).
[20] J. Turner, Managing Bandwidth in ATM Networks with Bursty Traffic, *IEEE Network Magazine*, **6** (5) (1992).
[21] ITU-T Recommendation I.371, Traffic Control and Congestion Control in B-ISDN, Geneve, Switzerland, 1992.
[22] ATM Forum, *ATM User-Network Interface Specification Version 3.0*, Prentice-Hall, Englewood Cliffs, N.J. (1993).
[23] E. Wallmeier and T. Worster, The Spacing Policer, An Algorithm for Efficient Peak Bit Rate Control in ATM Networks, *ISS'92*, Yokohama, Japan, October 1992.
[24] D. Deloddere, P. Reynders and P. Verbeeck, Architecture and Implementation of a Connectionless Server for B-ISDN, *XIV Int. Switching Symp.*, Yokohama, Japan, October 1992.
[25] G. Clapp, LAN Interconnection Across SMDS, *IEEE Network Magazine*, (September 1991) 25–32.
[26] K. Pawlikowsky, Steady-State Simulation of Queuing Processes: A Survey of Problems and Solutions, *ACM Computing Surveys* **22** (2) (1990) 123–170.
[27] V.S. Frost and B. Melamed, Traffic Modelling for Telecommunications Networks, *IEEE Communications Magazine* **32** (3) (1994) 70-81.