An Efficient ATM Traffic Generator for the Real-Time Production of a Large Class of Complex Traffic Profiles

Dimitrios Loukatos, Lambros Sarakis, Kimon Kontovasilis, and Nikolas Mitrou

Abstract: **This paper presents an advanced architecture for a traffic generator capable of producing ATM traffic streams according to fully general semi-Markovian stochastic models. The architecture employs a basic traffic generator platform and enhances it by adding facilities for "driving" the cell generation process through high-level specifications. Several kinds of optimization are employed for enhancing the software's speed to match the hardware's potential and for ensuring that traffic streams corresponding to models with a wide range of parameters can be generated efficiently and reliably. The proposed traffic generation procedure is highly modular. Thus, although this paper deals with ATM traffic, the main elements of the architecture can be used equally well for generating traffic loads on other networking technologies, IP-based networks being a notable example.**

Index Terms: **ATM, network tools, traffic generation, traffic modeling and simulation.**

I. INTRODUCTION

In the past few years, there has been an increasing interest in network traffic generators, particularly those with an ability of producing the complex and bursty traffic streams loading modern broadband networks. And for a good reason: Indeed, traffic generation tools, in conjunction with analysis tools, can assist in tuning networks, so as to provide and guarantee improved quality of service (QoS) at reduced cost levels. More generally, traffic generators can be an asset towards an efficient and cost effective network design, configuration, and control. A key feature of these devices is their ability to realistically simulate in real time a wide range of payload conditions, which can be directly applied to actual networks for the purpose of studying their behavior. In most cases, this is easier, faster, cheaper, and more controllable than using real traffic sources.

Due to their importance, various traffic generators have been presented, some of them being commercial equipment [1]–[3], others being the outcome of research work [4], [5]. However, most of the relevant technology currently available is concerned with the generation of traffic profiles through a specification of low level characteristics. In ATM, for example, most generators support traffic profiles specified only at the cell level. Such an arrangement is not satisfactory, because the important characteristics capturing the burstiness of traffic and having an impact on the network's operation are best described by aggregate statistical quantities at a higher level.

As a contribution towards improving this situation, the paper presents an architecture of a traffic generator that is capable of producing a broad class of bursty ATM traffic profiles, each specified through a high level model. More specifically, the paper employs an efficient ATM traffic generator platform [4], [5] and enhances it by adding facilities for "driving" the cell generation process according to the model specifications. In the high-level framework, each traffic stream is represented as a semi-Markovian model featuring an arbitrary (finite) number of states (each associated with a data rate), Markovian transitions from state to state, and state holding times possessing arbitrary distribution functions.

This class of traffic profiles is quite general and can accurately represent a wide range of bursty network traffic sources originating from various applications. Plain Markov-modulated rate processes (featuring exponentially distributed state sojourns) have been successfully employed for representing voice [6], [7], video [8]–[11], and data [12], [13] traffic. The additional generality offered by the semi-Markovian models, namely the ability of specifying general probability distribution functions for the state durations, permits accurate modeling of traffic featuring heavy-tailed bursts [14] or being subjected to shaping (e.g., leaky bucket based regulation) or throttle control.

It should be noted that the proposed traffic generation procedure is highly modular. Thus, although the paper deals with ATM traffic, the main elements of the architecture can be used equally well for generating traffic loads on other networking technologies, IP-based networks being a notable example.

The rest of the paper is structured as follows: Section II identifies the basic components of the generator and discusses briefly the low-level operations that govern the actual emission of cells by the hardware and are used as the foundation for the high-level software constructs. Section III presents the traffic specification in terms of semi-Markovian models and describes the high-level algorithms that drive the cell generation engine. This section discusses several kinds of optimization that have been employed for enhancing the software's speed to match the hardware's potential and for ensuring that traffic streams corresponding to models with a wide range of parameters can be generated efficiently and reliably. Measurements on the overall efficiency of the tool are discussed in Section IV, while Section V reports on results from using a prototype of the traffic generator in experiments. Finally, Section VI provides some concluding remarks.

Manuscript received September 19, 2002; approved for publication by Chongkwon Kim, Division III Editor, January 18, 2005.

D. Loukatos and N. Mitrou are with the Electrical and Computer Engineering Department, National Technical University of Athens, Greece, email: dlouka@telecom.ntua.gr, mitrou@softlab.ntua.gr.

K. Kontovasilis and L. Sarakis are with the Institute of Informatics and Telecommunications, National Center for Scientific Research "Demokritos", Athens, Greece, email: {kkont, sarakis}@iit.demokritos.gr.

Work partly funded by the Greek General Secretariat for Research and Technology (GSRT), through research grant PENED 99 ED 92.

II. ARCHITECTURE OF THE TRAFFIC GENERATION ENGINE

The traffic generator (TG[1]) consists of hardware (a PC card) and software. The hardware is a compound PC board, made up from a 'motherboard' that hosts the PC bus interface and the cell stream generation functionality and a 'daughterboard' that acts as a NIC, undertaking the serial ATM interface to the physical layer. The two sub-boards are connected via a UTOPIA Level 1 connector. The 'motherboard' [5] allows the simultaneous generation of multiple parallel ATM streams, each of which may follow a distinct traffic profile. Cells from these parallel streams are multiplexed over the daughterboard and subsequently output over the physical interface. The prototype hardware has been implemented for OC-3 ATM links; a modified version for OC-12 also exists. The hardware configuration supports up to 16 parallel streams and can sustain production of cells at a rate up to the physical interface's speed.

As it will be discussed fully in the following, the hardware is driven by simple software-provided data, called elementary traffic events. To some extent, these act on the TG hardware as machine instructions do on a computer processor. Thus, achievement of the hardware's potential for cell generation at full rate is directly dependent on the software's ability to provide the appropriate elementary traffic events at the right pace.

In particular, two software modules are relevant to the generator: The *traffic profiles creator* (TPC) and the *TG programmer* (TGP). The TPC is an off-line application that provides a user interface for defining traffic models and storing appropriate parameters of these models to computer files for later usage. The TGP operates on-line and undertakes the tasks of associating each of the hardware's traffic stream units with previously defined traffic models (by reading files generated by the TPC), sampling high-level traffic events according to the associated traffic models, converting these samples to low-level elementary traffic events, and using the latter for driving the cell generation hardware. The TGP is also used for activating/deactivating individual traffic stream units and for monitoring the cell generation process corresponding to each of them, by receiving from the TG hardware statistics about the generated cells.

Clearly, the task of making the TG capable of producing cell streams compliant with complex semi-Markovian traffic models involves appropriate design and coding of both of the TPC and TGP software modules. Addressing TPC is easier because, besides involving only high-level constructs, it runs as an off-line process without hard performance requirements associated with it. TGP is more complex because it must output the appropriate elementary traffic events suitably fast, if a performance bottleneck is to be avoided. The algorithms undertaking high-level traffic sampling and interfacing to the low-level traffic events representation, as well as the efficiency of these algorithms, are the subject of the next section. Before embarking on that, the structure of elementary traffic events understood by the TG hardware is reviewed in the following.

As already mentioned, the TG hardware regards each of the traffic streams produced in parallel as a plain cell sequence whose pattern is determined by a succession of elementary traffic events, each corresponding to either an *active* or an *idle* period. An active period is a burst, characterized by its size NC (signifying the number of cells in the burst) and the inter-cell distance ID. Clearly, an active period lasts for $NC(ID+1)$ ATM slots. An idle period denotes a time-interval during which no cells are produced. It is characterized by its duration SD (for "silence duration") in ATM slots. The top part of Fig. 1 summarizes the relation of each kind of traffic event to its defining parameters.

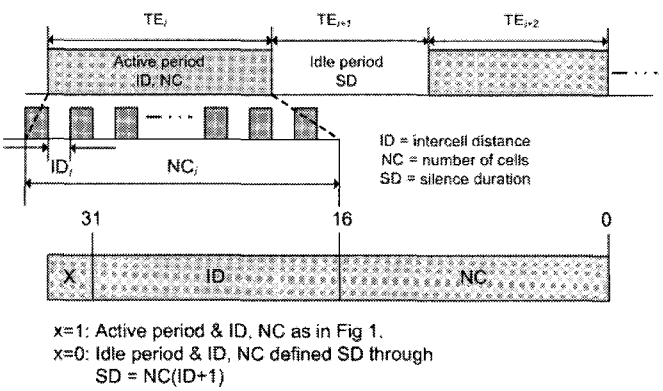


Fig. 1. Top: The structure of elementary traffic events. Bottom: NC/ID, or SD representations in the event storage memory.

It follows that an elementary traffic event should be represented as either a (NC, ID) pair or an SD value. However, to simplify the hardware, the event is always passed as a word of length 32 bits, containing the value of a (NC, ID) pair (and called elementary traffic event pair—ETEP—in the sequel), as in the layout at the bottom of Fig. 1. Whenever the most significant bit (MSB) is set, the word is taken to correspond to an active period and the hardware uses the values of the NC and ID parameters to generate the cells as displayed in Fig. 1. If the MSB is unset, then the word corresponds to an idle period and the hardware operates as in the case of an active period, except for suppressing the generation of cells. This behavior has the effect of introducing an idle period of length $SD = NC(ID+1)$. In other words, in case of an idle period the TGP software undertakes the task of constructing appropriate values for the NC and ID parameters, so as to match the user-supplied value of SD.

In relation to the range of values that can be represented in an ETEP note that, according to Fig. 1, NC is a non-zero 16-bit field, ranging in the interval $[1, 2^{16} - 1]$; similarly, ID is a 15-bit field, ranging in $[0, 2^{15} - 1]$. Thus, due to the way silence durations are represented, SD takes values in the interval $[1, 2^{31} - 2^{15}]$, with increments larger than one in the uppermost part of this range. This feature allows for an extensive range of SD-values, while also maintaining a sufficiently fine degree of granularity.

We now turn into the mechanism through which ETEPs are fed to the hardware: Each stream unit of the hardware is assigned an on-board memory bank of size 1024×32 bits, called event storage memory (ESM), which is used for buffering ETEPs (a 32-bit word each) relevant to the stream. As the hardware produces cells, the ETEPs contained in memory

[1]For a list of all main acronyms used in this paper see Appendix-D.

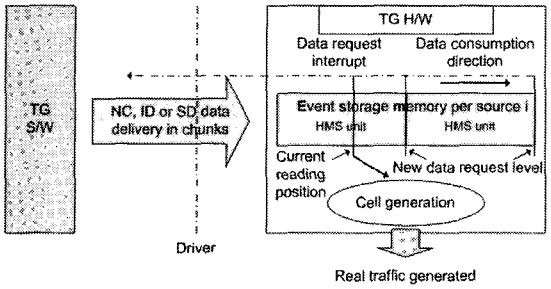


Fig. 2. Event storage memory handling.

are gradually consumed and must be replenished. Replenishment of the ESM for each stream is interrupt-triggered and occurs (in parallel for all active streams, using separate and asynchronous interrupts) in chunks equal to half the memory size, viz., $HMS = 512$ words, through a fast DMA-based PC interface that provides high rate data transfer from the host computer to the TG board. In this arrangement, once half the ESM contents have been consumed by the hardware, the driver probes through an interrupt the TGP software to provide a new set of HMS ETEPs while the board continues to generate cells from the remaining HMS words still in the ESM (see Fig. 2). These remaining ETEPs will be processed in time equal to

$$T = \sum_{i=1}^{HMS} NC_i(ID_i + 1) \qquad \text{(measured in ATM slots).} \quad (1)$$

Therefore, in order to avoid underflow of the ESM and maintain normal operation, the TGP software must prepare the new HMS words (according to the results of the high-level sampling) and transfer them to the board within time T. Consequently, the minimum possible value of T (depending on the high-level source parameters) is a critical time constant of the system.

It follows that the main challenge in the design of TGP is to ensure that ETEPs are generated fast enough to replenish the ESM for each active stream within the time determined by (1). Towards this end, the software has been structured in a such a way that most of the complex calculations are performed at a preprocessing stage, while the operations in real-time mode are kept as light as possible. The structure of TGP and the various algorithms that it employs are discussed in detail in the following section.

III. TRAFFIC GENERATION THROUGH HIGH-LEVEL MODELS

A. The Structure of Semi-Markov Traffic Models

As already mentioned, the traffic generator adopts a high-level traffic specification in terms of (time invariant) semi-Markovian models (see, e.g., [15]). According to this framework, each traffic model is characterized by a set of N states. During a sojourn at some state i, a constant, state-dependent data rate r_i is maintained. (The modeling framework assumes a "fluid-flow" approach, according to which more complex cell-level rate fluctuations within each state can be ignored. This approach is entirely adequate for representing the important burst-level phenomena, while at the same time avoiding excessive low-level complexity.) Transitions among states occur according to a discrete irreducible Markov chain with a transition probability matrix $P = [p_{ij}]$. The sojourns at a state i are independent and identically distributed random variables, of a general probability distribution function (PDF) $F_i(\cdot)$. The well-known MMRP traffic models (see, e.g., [13]), where all sojourn times are exponentially distributed, are an important special case of semi-Markovian models.

A.1 A Special Case: Superposition of Exponential On/off Sources

Many real-world traffic sources can be modeled as the superposition of a number of exponential on/off sources (see, e.g., [10] and [11]). Furthermore, many QoS theory fundamentals are based on traffic load of this form. Consequently, it is important to equip the traffic generator with the ability to generate such loads efficiently. However, it is quite expensive to allocate one stream module of the traffic generation hardware to a single on/off source. That would unnecessarily limit the maximum number of on/off sources to 16. By exploiting the generator's potential of admitting general specifications of semi-Markovian models, it is possible to assign the whole homogeneous superposition of a number N of exponential on/off traffic streams to a *single* stream module of the TG board.

Indeed, the said superposition is equivalent [12] to a semi-Markovian model comprising $N + 1$ states, each with exponentially distributed sojourn times. Given the peak rate r, and the mean on and off periods, τ and σ, respectively, of a single on/off source, the parameters of the model describing the superposition are given as follows:

- State rates $r_i = ir$, for $i = 0, \cdots, N$.
- Mean sojourn times $\gamma_i = [i/\tau + (N - i)/\sigma]^{-1}$, for $i = 0, \cdots, N$; due to exponentiality, these mean values completely determine the sojourn PDFs, as $F_i(x) = 1 - e^{-x/\gamma_i}$.
- Transition probability matrix $P = [p_{ij}]$, such that $p_{i,i+1} = \gamma_i(N - i)/\sigma$, for indices $i = 0, \cdots, N - 1$, and $p_{i,i-1} = \gamma_i i/\tau$, for $i = 1, \cdots, N$, all other transition probabilities being zero.

The traffic profiles creator (TPC) software, which provides the user interface for defining traffic models, makes the task of specifying a superposition of on/off sources particularly easy, because it requires user-input for just the parameters of a single on/off source and the number of sources participating in the superposition and undertakes itself the construction of the global model. Appendix-C provides further details on the output produced by TPC.

B. Top Level Architecture of the TGP Software

From a high level view, traffic generation corresponding to a semi-Markovian model consists of a sequence of events (high level traffic events—HLTEs). Each HLTE signifies the emission of cells at a constant rate r_i, depending on the currently occupied state i, for a duration chosen through sampling the respective PDF governing the sojourns at this state.

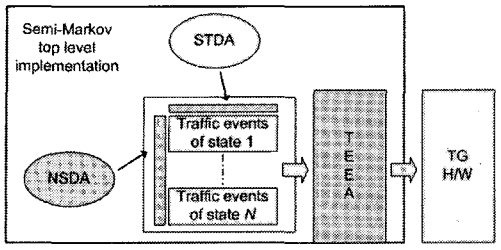


Fig. 3. Basic functional diagram for generating semi-Markovian traffic.

With the notion of an HLTE at hand, the task of semi-Markovian traffic generation requires an algorithm to decide on the next state to be occupied by the model (next state decision algorithm—NSDA) and an algorithm to sample the sojourn time that will be spent at the chosen state (sojourn time decision algorithm—STDA). Furthermore, the nature of ATM and the structure of the traffic generation hardware suggest the need for two more mechanisms: One to convert data-rates, and the durations for which these rates are sustained, to ETEPs, i.e., (ID, NC) pairs (rate to pairs conversion algorithm—RPCA) and another to encapsulate the high-level traffic event sequence into units of size HMS (high level traffic events encapsulation algorithm—TEEA).

The relation between these building blocks (commented individually in Sections III-C–III-F) is depicted in Fig. 3.

C. The Next State Decision Algorithm (NSDA)

Given as input an integer value expressing the state visited last time, the NSDA determines the state to be visited next. For this purpose, NSDA makes use of the transition probabilities matrix $P = [p_{ij}]$, as follows: During an initialization step, performed once before the actual traffic generation process, NSDA computes the aggregate transition probabilities matrix $A = [a_{ij}]$, where $a_{ij} = \sum_{k=0}^{j} p_{ik}$, for all $i, j = 0, \cdots, N-1$. During normal operation, given the state i currently occupied, the next state is decided upon by drawing a random number s in the interval $[0, 1)$ and determining the smallest index j, such that the quantity a_{ij} exceeds s.

Since the transition probability matrix P does not change over time, the partial probability sums are also invariant and having them ready into the elements of A reduces the time needed to determine the next state.

D. The Sojourn Time Decision Algorithm (STDA)

STDA consists of two modules: STDA_1, an initialization step performed once in startup, and STDA_2, which is executed repeatedly during the generation process. The preprocessing stage of STDA_1 undertakes the conversion of the sojourn PDFs into a form that allows for the efficient selection of actual sojourn samples by STDA_2 during on-line operation. Introduction of the preprocessing makes sense because the PDFs do not change over time.

More specifically, STDA_1 reads data that determine the PDFs governing the sojourn times at all states of the model. Then, for each state the algorithm computes values of the respective PDF[2] and creates an array of size M that contains inverse distribution function values (i.e., sojourn time values), which correspond to equally spaced PDF values in the range from 0 to $(M-1)/M$. The dimension M (common to the arrays of all states) is chosen large enough to ensure a sufficiently dense representation of the PDF, while also avoiding excessive storage requirements.

Once the arrays have been produced, STDA_1 exercises the RPCA algorithm upon each element of each array, taking as input the sojourn time of this element and the data rate associated with the corresponding state, and transforms this input into a collection of ETEPs. This is done in a way explained in the next subsection and further detailed in Appendix-A.

During the cell generation process, the STDA_2 performs (in a highly efficient manner) a much simpler task, namely that of drawing a random index into the array corresponding to the state picked by the NSDA. The low-level traffic event information in the chosen element is then fed into TEEA, which has the responsibility of generating HMS ETEPs for download to the hardware. TEEA is described in Section III-F.

E. The Rate to Pairs Conversion Algorithm (RPCA)

This algorithm, used during the preprocessing stage only, tackles the task of converting (*rate*, *duration*) pairs into low-level ETEPs of the form (NC, ID). (As explained in Section II, even plain silences, i.e., cases where the data rate is null, are converted to this form.) However, it is not always possible to convert an arbitrary (*rate*, *duration*) specification into a single ETEP. Indeed, for any given choice of the ID parameter, the cell-generation rate is equal to $R/(ID+1)$, where R is the maximum achievable rate (155.52 Mb/s for the STM-1 link in our implemented prototype). Since ID assumes integral values, the set of directly achievable rates is also discrete, with a granularity that is coarser in the upper range of rate values (corresponding to lower ID values) and finer in the lower range.

For generating at any other rate value, not directly contained in the abovementioned discrete set, an approximation must be made. The generator follows the approach of attaining the target rate r_i (in an average sense, over the sojourn in the respective state i) through a rate variation process, which emits NC_l cells at rate r_l followed by NC_h cells at another rate r_h. The rate r_h is chosen equal to the lowest actually achievable rate (i.e., corresponding to an integral value of ID) above r_i, while r_l is the highest actually achievable rate below r_i. The weights NC_l and NC_h are determined so as to approximate the desired average rate as closely as possible. Furthermore, the weights obey the relation $NC_l + NC_h = NC$, the latter quantity being equal to the total number of cells for the HLTE in question, determined as the total data to be generated (equal to $rate \times duration$), rounded to an integral number of cells. (The cell size used for the conversion of data units includes the physical layer overhead if the rate r_i also refers to the physical layer; otherwise both quantities refer to the ATM layer and a cell-size equal to 53 bytes is used.)

[2]The computation involves calculations based, either on a closed form formula (when the PDF belongs to a standard built-in set) or on linear interpolation over tabulated PDF values supplied in a file from the user. The second method allows for the specification of arbitrary PDFs.

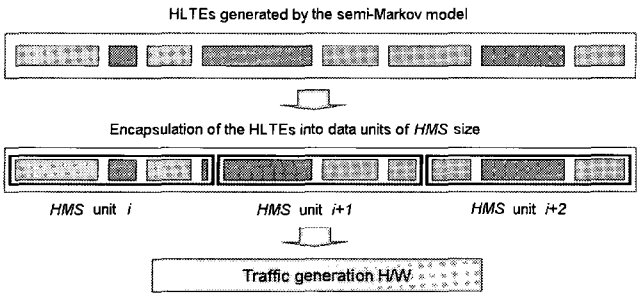


Fig. 4. Encapsulation of HLTEs into HMS units via TEEA.

One final complication is that the resulting values of NC_l and/or NC_h may be greater than the maximum value that can be represented by an ETEP (i.e., greater than $NC_{\max} = 2^{16} - 1$). In this case, instead of using just two ETEPs, the HLTE is broken down to a number of rate variation cycles, each of the form discussed previously. The decomposition in cycles may also be useful even when $NC_{l,h} < NC_{\max}$, in order to manage the rate fluctuations better, thus approximating the target rate with higher fidelity. Further details on the RPCA mechanisms may be found in Appendix-A.

F. The Traffic Events Encapsulation Algorithm (TEEA)

As already discussed in previous occasions, every HLTE is mapped to a number of ETEPs. TEEA, triggered by a hardware interrupt, undertakes the task of encapsulating these data into units of length HMS and downloading them to the hardware. In doing so, TEEA also acts as the caller of NSDA and STDA_2, for obtaining "fresh" samples, when these are required.[3] More specifically, when triggered by a data request interrupt, TEEA executes as follows:

Restore unprocessed part of current HLTE;
While (data unit of HMS is not filled)
  If (the HLTE has run out of ETEPs) **then**
    Obtain a new HLTE by applying NSDA and STDA_2;
  End if
  Add next ETEP of the HLTE into that data unit;
End while
Download data unit of HMS via the driver to the TG hardware;
Save unprocessed part of current HLTE;

The mechanics of the TEEA are further illustrated in Fig. 4, the upper part of which displays the sequence of HLTEs (each containing a number of ETEPs) picked by NSDA and STDA_2. The length of each HLTE in the figure denotes the number of corresponding ETEPs while different shades/colors correspond to different high-level states, and thus rates. The lower part of Fig. 4 illustrates the encapsulation of the ETEPs into HMS units (indicated by the surrounding frames). As it can be observed, the ETEPs of a single HLTE may be distributed in more than one successive chunks of length HMS.

[3]As explained in Section III-D, RPCA processing is performed only during the preprocessing stage, in the context of STDA_1 operations.

G. Optimization Issues

As mentioned in previous occasions, TGP must be capable of producing the ETEPs driving the cell generation hardware within $T_{\min}$, the minimum value of the critical time in (1) at the end of Section II. To ensure that this constraint is respected, many time-consuming conversion operations are executed off-line, during a preprocessing stage. Such operations include RPCA in its entirety and the most complex part of STDA, the STDA_1. Thus, only the simple operations of STDA_2, NSDA, and TEEA are invoked on-line. With these arrangements, most semi-Markovian models, even those with stringent parameter values, can be generated within a time-window of length $T_{\min}$. The range of achievable model parameters in the implemented prototype is summarized in Appendix-B.

Moreover, there is special provision for particularly "nasty" models (i.e., those containing sojourn times of extremely small duration-less than 0.1 ms) that may give rise to very small values of $T_{\min}$. In order to maintain the integrity of the traffic generation process in such extreme circumstances, an alternative mode of operation is used. In that mode, NSDA and STDA_2 remain as previously described, but are invoked in a "batch fashion" during the preprocessing stage, so as to produce a large number, namely $K \times HMS$, of ETEPs, where K is a large power of 2. This 'pool' of data is stored in the host computer's memory, from where TEEA (which functions differently from the description given in Section III-F) 'pumps' HMS neighboring ETEPs, at any time an interrupt is being serviced. The start of this block of neighboring elementary pairs is selected randomly in the interval $[0, K \times HMS - 1]$. Although this alternative mode of operation imposes some distortion on the random aspects of the model, the result is quite acceptable, as long as K is large enough and the number of states is not excessive.

IV. MEASUREMENT OF THE SOFTWARE'S EFFICIENCY

In order to validate the efficiency of the TGP software, experiments were performed. An old and slow PC (featuring a Pentium I processor at 166 MHz and 64 MB of RAM) was intentionally used as host of the traffic generation tool, aiming at the maximization of possible bottleneck phenomena during traffic generation. Performance of the TG software was captured by a monitoring tool logging the CPU utilization.

The values for the traffic model parameters were appropriately selected to reflect demanding cases of traffic generation. The most stringent such cases occur when the total cell rate is close to the link rate and when the states of each semi-Markovian model feature very small mean sojourn times (the particular shape of the sojourn PDF is less critical). Both these factors translate into fast consumption of ETEPs by the hardware, thus to more frequent interrupts for ESM replenishment. Indeed, the higher the state rates, the smaller the ID values in the corresponding ETEPs become (see (4) in Appendix-A), reducing the time span over which the ETEP is active (viz., (7) in AppendixVI-A). Similarly, smaller mean sojourn times call for more frequent state transitions (thus calls to NSDA) and each of them requires a fresh set of ETEPs (obtained through STDA_2).

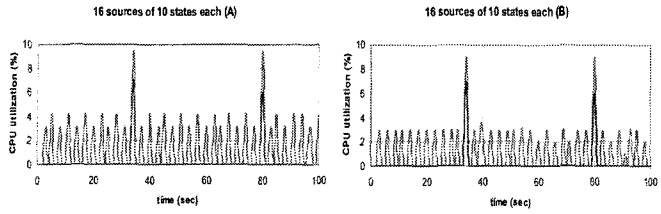


Fig. 5. CPU utilization during traffic generation.

We now present results for a particular experiment addressing traffic generation according to stringent model parameters. In this experiment, a semi-Markovian model with 10 states was used. Traffic rates and mean sojourn times were fixed at 9 Mbps and 1 ms (a very low value), respectively, for all states. All 16 traffic stream units of the hardware were activated to produce traffic according to this model, for a total traffic rate of 144 Mbps. It is mentioned that, since all state-rates are equal, the model is equivalent to a trivial constant-bit-rate source. Although the reduced model would have been employed in traffic generation practice the more complex semi-Markovian representation was deliberately employed in the experiment, to enforce a complex multi-state setting and equally stringent parameters (and the consequent high computational requirements) over all states.

Fig. 5 displays traces of the CPU utilization (as captured by the performance monitoring utility) at the host computer, for 100 s of traffic generation. The left trace corresponds to a run during which traffic generation proceeds according to the normal mode, i.e., when STDA_2, NSDA, and TEEA are invoked online. The average CPU utilization is 1.4% and the peaks do not exceed 4% (the two peaks around 10% are unrelated to the main traffic generation functionality and will be discussed later). The right trace in the figure corresponds to a run of the generator in the alternative mode intended for particularly 'nasty' models, as explained in Section III-G. The average CPU utilization is now 1.1% and the peaks are around 3%. The drop of about 25% in the processing load (reflected in both the mean and peak utilization values) is due to the fact that in the alternative mode the high-level STDA_2 and NSDA operations are omitted (being invoked in "batch" fashion during the preprocessing stage, instead).

The trace comparison just mentioned also indicates that about 25% of TGPs processing is devoted to STDA_2 and NSDA operations, while the remaining 75% goes to the processing of interrupts and the downloading of ETEPs to the hardware through TEEA. This is further confirmed by observing the trace marked by the 'dark' line in both subfigures (almost overlapping with the horizontal axis and the large peaks around 10%), which corresponds to the processing load when interrupts are not processed and the initial batch of HMS traffic events are wrapped around by the TG hardware. In this case even the TEEA processing is nullified and the CPU utilization drops to 0.01%.

The peaks around 10% are still captured in the 'dark lined' trace, however, and this indicates that they are unrelated to the main traffic generation process. Indeed, these peaks correspond to the computations required for the periodic refresh (approximately every 46 s) of graphical representations, employed by the user interface in the TG software for illustrating characteristics of the generated traffic. This operation is expensive on the host computer used in the experiment, which, as already mentioned, was old, slow, and with limited graphics support. It would have required a much smaller proportion of the CPU's power in a modern system. In any case, this load does not scale-up as the model's parameters become more stringent. Furthermore, this monitoring capability can be deactivated, leaving more CPU resources free for the fundamentals of traffic generation.

In concluding, even with a model featuring sojourn times as low as those in the experiment, there is room for increasing the rates (and the processing load required, which would grow proportionally) by at least a factor of 10. (This assumes a host system dedicated to traffic generation and OS functions, and that the graphical monitoring facility is retained). Taking into account that modern PCs are faster than the one used by more than an order of magnitude, it follows that TGP is capable of supporting full rate traffic generation at a level of at least OC-192, provided that upgraded TG hardware becomes available at these rates.

V. USAGE OF THE TRAFFIC GENERATOR IN EXPERIMENTS

Besides the performance efficiency measurements discussed in the previous section, the traffic generator was subjected to extensive testing, during which it demonstrated remarkably good behavior in accurately implementing complex user-defined traffic profiles. For validation purposes, in particular, the generator was setup to generate traffic pursuant to specially selected profiles. For each such case, the generated traffic stream was captured and tested by an ATM traffic analyzer. Relevant results are summarized in Fig. 6.

The upper left part of the figure relates to a model that contained a state featuring normally distributed sojourns (with negative samples converted to zero values). Sample visit durations at this state were captured by the analyzer (which was detecting states by a rate-change beyond an appropriately set threshold) and are displayed in histogram form. The shape of the histogram (and further numerical parameters pertaining to it) confirms that the sojourns were being generated according to the specifications.

The upper right part of Fig. 6 displays a time-trace of the rate fluctuations in another traffic stream, corresponding to a semi-Markovian model with 3 states, of a constant duration each. Different rate levels r_i were associated with the various states, specifically $r_0 = 0$ Mbps, $r_1 = 4.95$ Mbps, and $r_2 = 7.06$ Mbps. Again, the captured trace confirms the fidelity to the model specifications. It is mentioned that, in the implemented prototype generator, featuring an STM-1 output link at 155.52 Mbps, the level of 4.95 Mbps cannot be generated by means of a single ETEP, of the form (NC, ID), and was approximated by ETEPs of two neighboring rate-levels $155.52/(32+1)$ and $155.52/(31+1)$, in accordance with the description of the RPCA algorithm in Section III-F.

Lastly, the lower part of Fig. 6 displays (in semi-log scale) the probabilities with which content thresholds are exceeded at the output buffer of an ATM multiplexer, when the traffic load is

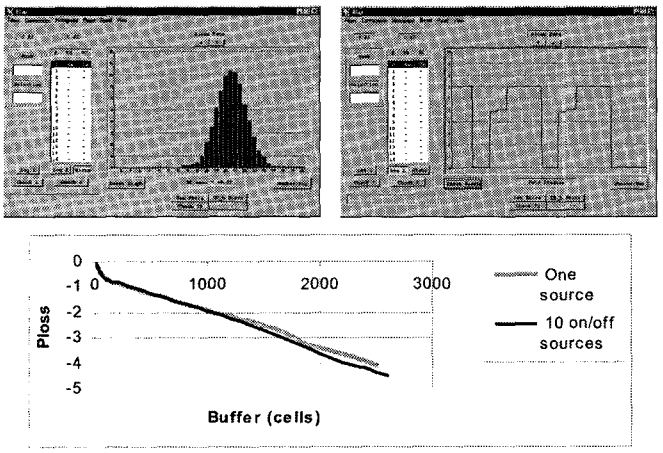


Fig. 6. Results from traffic generation tests.

a homogeneous superposition of 10 exponential on/off sources. Two curves have been obtained by tracing losses on the buffer. The curve labeled "one source" corresponds to generation of the traffic load through one traffic-stream module on the TG hardware, using the aggregate semi-Markovian model of Section III-A.1. The other curve corresponds to separate generation of the individual on/off sources, by means of 10 distinct hardware modules. Clearly, the two sets of results are quite close, and it would have been even closer, had the time of the experiment been greater, so as to reduce statistical errors associated with the data-gathering process.

We close this section by presenting results from an experiment relating to the large-scale multiplexing of traffic, performed in the context of a research project[4] that explored the usage of asymptotic theories for the modeling of congestion in large networking environments. The experiment involved the generation of many statistically identical on/off traffic streams (featuring a peak rate of 1500 cells/s and exponentially distributed on and off periods, of means 100 ms and 400 ms, respectively), which were forwarded (and multiplexed) at the same output port of an ATM switch. The port's output capacity was configurable to different values, through appropriate commands to the switch's management software. Three units of the prototype ATM generator were employed, each contributing 16 independent stream modules. Furthermore, each module was configured to produce traffic corresponding to the superposition of 10 on/off sources, for an overall effect of $3 \times 16 \times 10 = 480$ on/off streams.

The buffer contents at the common output port were being traced (by software that was making regular queries to the switch's network management system, through an out-of-band IP-link) and these samples were used for estimating the probability with which content thresholds were exceeded in the buffer. The 'overflow probability' graph was then compared to numerical results produced according to the theory for the homogeneous multiplexing of Markovian on/off fluid streams (following [12], with appropriate embellishments [16] that allow enhanced precision and better numerical stability for a large number of multiplexed sources).

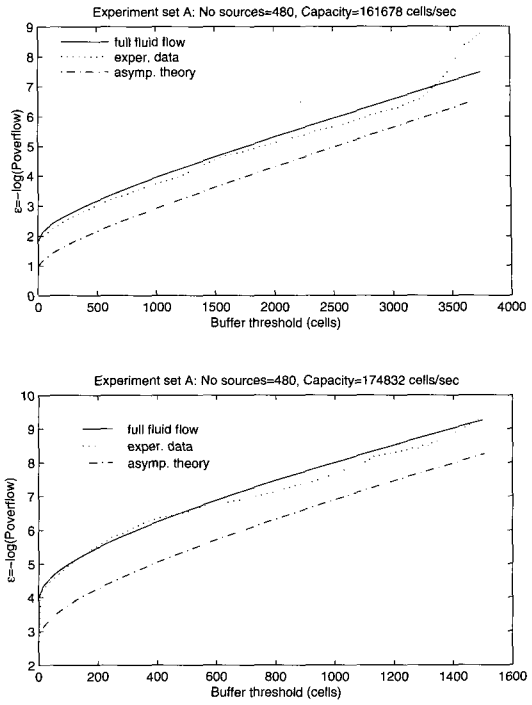


Fig. 7. Experimental vs. theoretical buffer overflows in on/off traffic multiplexing.

The 'solid-lined' curves at the top of the two subfigures within Fig. 7 display the theoretically predicted graphs of the overflow probabilities, for two different values of the output port's capacity. The dotted curves at the top of the subfigures capture the actually observed sample overflow probabilities. (The 'dash-dotted' graphs at the bottom of the subfigures refer to a corresponding asymptotic result, not relevant to the context of this paper.) In both cases, the sampled graph is accurate down to probability percentiles of about 5×10^{-3}, which, in the negative logarithmic scale (of base e) used, translates to an upper bound of about 5. This bound is due to a processing bottleneck in the buffer-polling software, which was issuing SNMP requests at a maximum rate of about 100 requests/sec. For an overall experiment duration of approximately 15 min, about 10^5 samples were collected, and this number, combined with the fact that about 500 observations at a certain occupancy level were at least required for reliable estimation, leads to the abovementioned probability threshold. The threshold could be further reduced by enlarging the experiment's duration.

As it may be evidenced from the figure, there is a close match between the observed and theoretical behavior for both cases, something that provides additional indirect evidence for the correctness of the traffic generation provided by the tool. The experiment just described may also serve as evidence to the potential of the generator for producing quite complex traffic loads (like the superposition of 480 individual on/off streams in the particular case at hand), which would hardly be realizable in a testbed setting, had the tool not been available.

[4]Funded by the Greek Secretariat for Research & Technology, under contract PENED 99 ED 92. The needs of this project provided part of the motivation for developing the traffic generator presented herein.

VI. CONCLUSIONS

This paper presented an advanced architecture for a traffic generator capable of producing ATM traffic streams compliant with fully general semi-Markovian stochastic models. The software that drives the traffic generation hardware employs several kinds of optimization, for enhancing the software's speed to match the hardware's potential and for ensuring that traffic streams corresponding to models with a wide range of parameters can be generated efficiently and reliably. A prototype of the traffic generator has already been successfully used in traffic experiments and this experience verified the capabilities and the effectiveness of the tool.

The architecture proposed in the paper is highly modular. Thus, although the case studied deals with ATM traffic, the main elements of the architecture can be used equally well for generating traffic loads of other networking technologies, IP-based networks in particular. In fact, IP traffic generation is simpler in some respects, because packets may be generated at arbitrary points in time and the size of the packets need not be fixed as in ATM. Therefore, when realizing semi-Markovian rate models for IP, the state transition algorithm NSDA and the sojourn-determination algorithm STDA remain unchanged, while the analog of RPCA (used to produce low-level rate specifications during a state sojourn) may be considerably simplified, by employing any desirable value for the packet interarrivals (which need not be an integral multiple of a base quantity). The ETEPs produced may be used by low-level driver software for actually creating the packets and feeding them to the host computer's NIC. Note that in the IP setting, a single ETEP per sojourn suffices.

Future research will address the issue of applying the ideas in this paper towards the production of tools for the generation of IP traffic.

APPENDIX

A. Analysis of the RPCA Processing

As mentioned in Section III-E, when the rate corresponding to some state, say r, does not correspond to an integral ID-parameter, it is attained (in an average sense over the state sojourn), through a rate variation process, which involves the two closest actually achievable rates r_l and r_h. By definition of these rates,

$$r_j = \frac{R}{1 + ID_j}, \qquad j = l, h, \tag{2}$$

where R is the link rate and $ID_{l,h}$ are integral. Furthermore, since the respective rates are neighboring,

$$ID_h = ID_l - 1. \tag{3}$$

In light of (2) and (3), the requirement $r_l < r < r_h$ translates to $ID_l < R/r < ID_l + 1$, thus

$$ID_l = \lfloor R/r \rfloor. \tag{4}$$

Equations (2), (3), and (4) fully determine the rates to be used for cell generation. At this point it is reminded that all rates must uniformly refer to either the physical layer (thus including the overhead for cell encapsulation) or to the ATM layer.

The next thing that must be determined by RPCA is the number of cells, NC_l and NC_h, that must be generated at rates r_l and r_h, respectively, in order to achieve an average rate equal to r over the state sojourn T. Obviously,

$$NC_l + NC_h = NC, \tag{5}$$

the total number of cells to be generated during the state sojourn, a quantity directly expressible in terms of the high-level specifications as

$$NC = [rT/d], \tag{6}$$

where d is a constant equal to the cell-size and where $[\cdot]$ denotes rounding[5] to the nearest integral value. (If the rates refer to the ATM layer $d = 424$ bits, otherwise the size is adjusted to include the cell-encapsulation overhead.) Further, observe that, for any choice of NC_l and NC_h, the time spent at each rate level (measured in ATM slots) is equal to

$$T_j = NC_j(ID_j + 1), \quad j = l, h, \tag{7}$$

and that, if an average rate equal to r is to be achieved, the relation $r_l T_l + r_h T_h = r(T_l + T_h)$ must also hold. By virtue of (2), (3), (5), and (7), this last relation is equivalent to $NC(R/r) = NC \times ID_l + NC_l$ which, by (4) and rounding to an integral value yields

$$NC_l = [NC(R/r - \lfloor R/r \rfloor)]. \tag{8}$$

Equations (5) and (8) fully determine the quantities $NC_{l,h}$. Moreover, they automatically take care of cases where the target rate is directly achievable, since then (8) yields $NC_l = 0$ and all NC cells are generated at rate $r_h = r$. Further inspection of (3) and (5)–(8) reveals that, if the rounding effects in (6) and (8) are ignored, $T_l + T_h$ becomes equal to the overall sojourn specification T, as it should. The rounding may be shown to impose a relative error for the sojourn value smaller than $1/NC$, which is negligible since, in all reasonable applications, $NC \gg 1$.

According to the discussion up to this point, an HLTE featuring a non-achievable rate is represented through two ETEPs, (NC_l, ID_l) and (NC_h, ID_h), with parameters determined by (3)–(6) and (8). However, this arrangement breaks down if any of $NC_{l,h}$ exceeds $NC_m = 2^{16} - 1$, the maximal NC-value that can be passed to the hardware for an ETEP. In order to deal with such 'overflow' instances, RPCA divides the total number of cells to be generated in $L + 1$ groups, where

$$L = \lfloor NC/NC_m \rfloor, \tag{9}$$

and where each of the first L groups contains a total of NC_m cells, while the last group is of size

$$NC_f = NC - L \times NC_m. \tag{10}$$

Each of the groups is then treated as if it referred to the whole state visit, by further being subdivided in two parts, corresponding to the emission of cells at rates r_l and r_h, respectively. The

[5]Rounding may be achieved in terms of the floor operator, through the identity $[x] = \lfloor x + 0.5 \rfloor$.

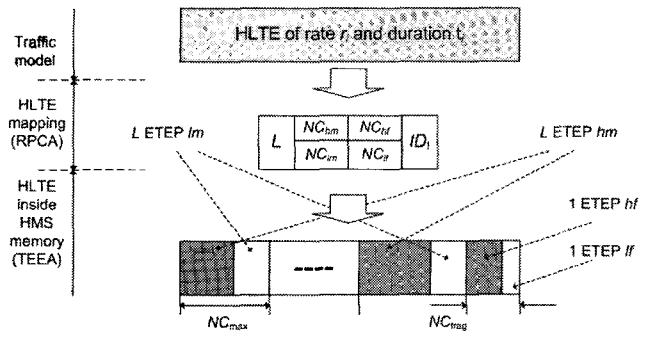
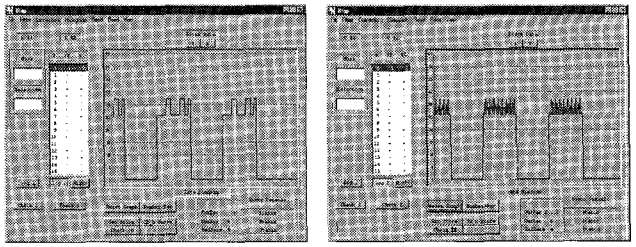


Fig. 8. Conversion of HLTEs into ETEPs.



(a) (b)

Fig. 9. The effect of different NC_m threshold values.

L groups of size NC_m are split in parts of length NC_{lm} and NC_{hm}, determined by equations analogous to (8) and (5), in which the quantity NC has been replaced by NC_m. A similar arrangement exists for splitting the last group in parts of length NC_{lf} and NC_{hf}. The rate-related ID-parameters are always determined through (4) and (3).

By the method of groups, an HLTE may ultimately be translated into $2(L+1)$ ETEPs (except when the division in (9) has zero remainder, signified by $NC_{lf} = NC_{hf} = 0$, in which case $2L$ ETEPs are produced). Cases that do not require a breakdown in groups correspond to $L = 0$ and result in 2 ETEPs, as explained earlier. Finally, whenever the rate r is directly achievable, then $NC_l = 0$ (for multiple groups $NC_{lm} = NC_{lf} = 0$), the events corresponding to r_l become void, and half the number of ETEPs are ultimately produced.

In general, for any HLTE given as input to RPCA the algorithm produces as output a set of six integral values for L, NC_{lm}, NC_{hm}, NC_{lf}, NC_{hf}, and ID_l (the latter also determining ID_h, through (3)). These values are stored in the corresponding element of the matrix produced during the STDA_1 preprocessing (see Section III-D) and are looked up by STDA_2, when called from TEEA, for producing the ETEPs corresponding to the HLTE retrieved. Fig. 8 sketches the production and usage of RPCA output.

Note that, although breakdown into groups was introduced as a means to overcome the limitation associated with the maximal NC-value imposed by the hardware to an ETEP, the technique is also useful for obtaining better management of the rate fluctuations during a state sojourn, towards approximating the target rate with higher fidelity. This is of particular importance when the rate specification corresponds to small values of $ID_{l,h}$, so that the difference between r_l and r_h becomes appreciable. Representing such an HLTE with just two ETEPs is not very accurate, because it essentially replaces the constant-rate duration with two smaller durations at distinctly different rates. Introducing alternating fluctuations between the two rate-levels is better, because the quicker rate oscillations require a smaller time-window for sufficient convergence to the corresponding average rate (equal to the target specification in both cases) and thus become unnoticeable in more contexts.

The rate-alteration effect can be achieved in all cases through the introduction of groups, as discussed previously, by adjusting the bound NC_m to a suitable value lower than the 'hard' limit $2^{16} - 1$. The traffic generation software provides for that, by allowing the user to define the ratio between the mean NC-value (i.e., the mean burst-size) and NC_m, individually for each of the model states, at the initialization stage of the overall traffic generation process.

The impact of different NC_m thresholds is shown in Fig. 9, which displays the output of an ATM analyzer tracing the traffic rates produced by the generator. The same traffic model has been used in both subfigures (an on/off model with constant sojourns and a peak rate that cannot be directly produced), but the NC_m used at the left is larger than the one at right. Consequently, and although the average talkspurt rate is the same for both examples, the left one features longer and fewer rate variations. It is noted that when the traffic of Fig. 9 is generated with an even smaller NC_m value, the analyzer cannot distinguish the rate fluctuations and reports a constant peak rate equal to the model specification.

B. Range of Model Parameters

The acceptable range of model parameters is very wide. The lower threshold for state sojourns is determined by the slot duration, itself dependent on the output link's capacity. For the prototype's STM-1 link at 155.52 Mb/s a slot lasts about 2.726×10^{-3} ms, thus, in principle, state sojourns as low as a few microseconds can be specified through $NC = 1$. Even when a lower bound is imposed on NC (say $NC = 100$) for assuring a small relative error between the specified and actual sojourns (see Appendix-A) sojourn values can still be as low as a fraction of 1ms.

The upper bound of a state's duration depends on the corresponding rate, the worst case being traffic generation at the link rate ($ID = 0$). In this case, each ETEP can last at most $NC_{\max} = 2^{16} - 1$ slots and, by breaking the whole sojourn in groups, as discussed in Appendix-A, a duration up to $L_{\max} \times NC_{\max}$ may be achieved,[6] where $L_{\max}$ denotes the maximum number of groups that can be represented internally. In the prototype, L-values are unsigned short quantities (i.e., 16-bit long), allowing for state durations longer than 3hr, for all rate values.

The maximum number of states allowed in a model (also determining the maximum number of sources in specifications of homogeneous on/off superposition) is a compile time-constant, which in the prototype has been set equal to 20 and may be readily increased.

[6] $L_{\max}$ is not pre-multiplied by 2 because the link rate is directly achievable, as explained in Appendix-A.

```
Number of States = 4

Rat_A =   2.120
Rate0 =   0.000  Duration0 = CONSTANT( 100.000 )
Rate1 =   2.000  Duration1 = EXPONENTIAL( 0.000 , 20.000 )
Rate2 =   4.000  Duration2 = NORMAL( 20.000 , 4.000 )
Rate3 =   5.000  Duration3 = FILE( D:\tg\trafcr010320\Dis1.ds )

Probability Matrix:

0.0000 0.0000 0.5000 0.5000
0.4000 0.0000 0.3000 0.3000
0.4000 0.2000 0.0000 0.4000
0.2000 0.8000 0.0000 0.0000
```

Fig. 10. Typical TPC output file.

```
Number of States = 6

Duration_ON = 10.00  Duration_OFF = 20.00

Rat_A =   1.667
Rate0 =   0.000  Duration = EXPONENTIAL( 0 , 4.0000 )
Rate1 =   1.000  Duration = EXPONENTIAL( 0 , 3.3333 )
Rate2 =   2.000  Duration = EXPONENTIAL( 0 , 2.8571 )
Rate3 =   3.000  Duration = EXPONENTIAL( 0 , 2.5000 )
Rate4 =   4.000  Duration = EXPONENTIAL( 0 , 2.2222 )
Rate5 =   5.000  Duration = EXPONENTIAL( 0 , 2.0000 )

Probability Matrix:

0.0000 1.0000 0.0000 0.0000 0.0000 0.0000
0.3333 0.0000 0.6667 0.0000 0.0000 0.0000
0.0000 0.5714 0.0000 0.4286 0.0000 0.0000
0.0000 0.0000 0.7500 0.0000 0.2500 0.0000
0.0000 0.0000 0.0000 0.8889 0.0000 0.1111
0.0000 0.0000 0.0000 0.0000 1.0000 0.0000
```

Fig. 11. TPC output for the homogeneous superposition of on/off traffic.

Finally, the range of rate specifications and the granularity of achievable values have been discussed in Section II. For the prototype's output link, the lowest nonzero achievable rate is 4.746 kbps (corresponding to $ID = 2^{15} - 1$) and the highest one is equal to the link capacity, approximately 155.52 Mb/s (both thresholds referring to the physical layer). It is reminded that zero rate values are achieved through silence events, specifying an SD-value.

C. Storage of Traffic Model Specifications into TPC-Produced Files

The traffic profile creator (TPC) software provides a user-interface for the specification of traffic model parameters. TPC processes the user-input, performs validity checks on the parameter values, estimates some derivative quantities (like the model's mean traffic rate) and stores a description of the model into a computer file of a standard format. The output file is read and parsed by the main traffic generation software (the TGP) during initialization, when some or all of the generator's hardware modules are being associated with corresponding traffic models. TPC runs independently from TGP (even at a separate computer) and its output files may be stored for later usage.

The contents of a typical TPC output file are displayed in Fig. 10. This file corresponds to a model with 4 states, of rates ranging from 0 to 5 Mbps (the implicitly assumed rate unit in TPC files). Three out of four states use state sojourn PDFs of predefined form, with appropriate additional parameter specifications (the mean state duration and, in the truncated normal case, the standard deviation, all assumed in ms-units), while the 4-th state employs a PDF specification through a table of values, contained in a designated input file. The user-specified transition probabilities are also contained in the file. TPC has calculated the mean traffic rate for this model (denoted as 'Rat_A') and the value is included in the file for reference.

When TPC is used for specifying models corresponding to the homogeneous superposition of exponential on/off sources, the user just needs to enter the number of on/off sources, the peak rate of each source, and the mean on- and off-durations. TPC undertakes the construction of the semi-Markovian model for the superposition (as described in Section III-A.1) and outputs the appropriate file in the standard form. Example output is shown in Fig. 11, corresponding to the superposition of 5 sources with a peak rate of 1 Mbps and mean on- and off-durations equal to 10 ms and 20 ms, respectively. All numerical quantities in the file were computed by TPC.

Table 1. Acronyms.

Acronym	Explanation
ESM	Event Storage Memory
ETEP	Elementary Traffic Event Pair
HLTE	High Level Traffic Event
HMS	Half Memory Size
ID	Inter-cell Distance
MSB	Most Significant Bit
NC	Number of Cells
NSDA	Next State Decision Algorithm
PDF	Probability Distribution Function
RPCA	Rate to Pairs Conversion Algorithm
SD	Silence Duration
STDA	Sojourn Time Decision Algorithm
TEEA	Traffic Events Encapsulation Algorithm
TG	Traffic Generator
TGP	Traffic Generator Programmer
TPC	Traffic Profile Creator

D. List of Acronyms

Table 1 summarizes the acronyms most frequently used in the paper.

REFERENCES

[1] ADTECH, "The ADTECH AX/4000 - The art of ATM testing," 1997.

[2] ALCATEL-STR, "ALCATEL 8643 - ATGA user manual."

[3] WANDEL&GOLDERMAN, "ATM-100, description and operation manual."

[4] S. Hondas *et al.*, "A flexible and cost-effective ATM traffic generator," in *Proc. 5-th IFIP Workshop on Performance Modeling and Evaluation of ATM Networks*, 1997.

[5] S. Hondas *et al.*, "ATM traffic generator card: An integrated solution," in *Proc. ISCC'98*, 1998, pp. 161–166.

[6] P. T. Brady, "A statistical analysis of on-off patterns in 16 conversations," *Bell System Tech. J.*, vol. 47, no. 1, pp. 73–91, 1968.

[7] J. G. Gruber, "A comparison of measured and calculated speech temporal parameters relevant to speech activity detection," *IEEE Trans. Commun.*, vol. COM-30, pp. 728–738, 1982.

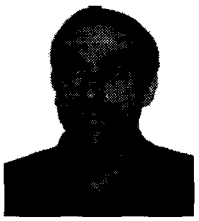
[8] D. P. Heyman *et al.*, "Modeling teleconference traffic from VBR video coders," in *Proc. ICC'94*, 1994, pp. 1744–1748.

[9] D. P. Heyman, A. Tabatabai, and T. V. Lakshman, "Statistical analysis and simulation study of video teleconference traffic in ATM networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, no. 1, pp. 49–59, 1992.

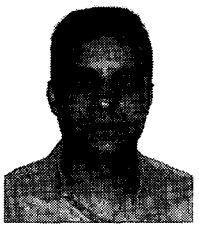
[10] B. Maglaris *et al.*, "Performance models of statistical multiplexing in packet video communications," *IEEE Trans. Commun.*, vol. COM-36, pp. 834–844, 1988.

[11] P. Sen *et al.*, "Models for packet switching of variable-bit-rate video sources," *IEEE J. Select. Areas Commun.*, vol. 7, pp. 865–869, 1989.

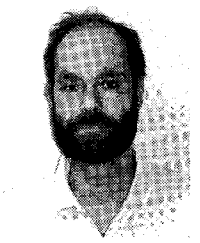
[12] D. Anick, D. Mitra, and M. M. Sondhi, "Stochastic theory of a data-handling system with multiple sources," *Bell System Tech. J.*, vol. 61, pp. 1871–1894, 1982.
[13] T. E. Stern and A. I. Elwalid, "Analysis of separable Markov-modulated rate models for information-handling systems," *Adv. Appl. Prob.*, vol. 23, pp. 105–139, 1991.
[14] O. Boxma and V. Dumas, "Fluid queues with long tailed activity period distributions," *Comput. Commun.*, vol. 21, pp. 1509–1529, 1998.
[15] E. Cinlar, *Introduction to Stochastic Processes*, Prentice Hall, 1975.
[16] K. P. Kontovasilis and N. M. Mitrou, "Bursty traffic modeling and efficient analysis algorithms via fluid-flow models for ATM-IBCN," *Annals of Operations Research*, vol. 49, pp. 279–323, 1994.



Dimitrios Loukatos is currently a research associate at the Electrical Engineers School of the National Technical University of Athens, in Greece. He received the Diploma in Electrical and Computer Engineering and the Ph.D. degree in Telecommunications and Computing, both from the National Technical University of Athens, in 1997 and 2002, respectively. His research interests include computer systems design, communication traffic modeling and analysis, and mobile and wireless networking. He is a member of the Technical Chamber of Greece since 1998.



Lambros Sarakis received the Diploma in Electrical Engineering & Computer Science from the National Technical University of Athens (NTUA) in 1996, the M.Sc. in Communications & Signal Processing from Imperial College in 1997, and the Ph.D. in Computer Science from NTUA in 2002. Since 2004, he is a research associate with the Institute of Informatics & Telecommunications of the National Center for Scientific Research "Demokritos." His research interests include traffic characterization and analysis, QoS support in IP networks, and hardware/software co-design for embedded systems. He is a member of the IEEE and of the Technical Chamber of Greece.



Kimon Kontovasilis received the Diploma in Electrical Engineering from the National Technical University of Athens (NTUA) in 1987, the M.Sc. in Computer Science from the North Carolina State University in 1990, and the Ph.D. in Electrical Engineering from NTUA in 1993. Since 1996, he is with the Institute of Informatics and Telecommunications of the National Center for Scientific Research "Demokritos" as a member of the research staff, currently ranking as Principal Researcher. His research interests include modeling, performance evaluation, traffic control and resource management of wired and wireless networks, the management of heterogeneous jointly operated wireless networks, and distributed architectures for network management systems, as well as queueing theory. He is a member of IFIP WG6.3 and of the Technical Chamber of Greece.

Nikolas Mitrou received the Diploma degree in Electrical Engineering from the National Technical University of Athens (NTUA) in 1980, the M.Sc. degree in Systems and Control from the UMIST, Manchester, in 1982, and the Ph.D. degree in Electrical Engineering from NTUA in 1986. He is now a full professor in the School of Electrical and Computer Engineering of NTUA. His research interests are in the areas of digital communications, communication networks, and networked multimedia in all range of studies: Design, implementation, modeling, performance evaluation & optimization, and applications. Prof. Mitrou is a member of the IEEE and a member of the IFIP (TC6).