Parallel Simulation of End–to–End ATM Models

I. F. Akyildiz & I. Joe, * R. M. Fujimoto, † I. Nikolaidis ‡

Abstract

The problem of end–to–end connection performance in Asynchronous Transfer Mode (ATM) networks is extremely important because its solution provides the information (such as end–to–end delay distribution and cell loss ratio) necessary for the definition of the Quality of Service (QoS) guarantees for real–time connections. Due to the analytical complexities inherent in this problem, we are seeking to develop an efficient simulation technique for the fast production of results. The presented approach is a combination of an earlier time–parallel technique and probabilistic routing. An implementation of this technique on a multiprocessor with modest capabilities shows promising speed–up. Using the developed simulator, several experiments are conducted for the end–to–end behavior of ATM connections using bursty traffic models, including ones with active periods derived from heavy tailed distributions. An important feature of the study is that the interfering traffic is of the same type as the end–to–end traffic. Thus, no assumptions are made about traffic smoothing or approximations of the aggregate interference process by Poisson arrivals. The presented end–to–end performance results include the Cell Loss Ratio (CLR) and the delay distribution at the multiplexers and at each intermediate switch of the connection path. Among the presented findings, it is illustrated that if only few sources with heavy tailed active periods are multiplexed and if the utilizations are low, then the end result is not necessarily a worse performance as one would anticipate when larger aggregations of such sources asymptotically approximate a self–similar process.

Key Words: *Time Parallel, End-to-End, Burst-Level, Splitting Ratio, End-to-End Cell Loss Ratio, End-to-End Delay.*

---

*Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, {ian,inwhee}@ee.gatech.edu; This work was supported in part by NSF under grant # CCR-9404726

†College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, fujimoto@cc.gatech.edu

‡Computing Science Dept., University of Alberta, Edmonton, Alberta Canada T6G 2H1, yannis@cs.ualberta.ca

1 Introduction

The Asynchronous Transfer Mode (ATM) is the most significant standardized network technology in the area of high–speed broadband communications. Since ATM networks carry a complex mixture of traffic including voice, video, data, still image, etc. over the same medium (typically, fiber optic), a crucial problem that needs to be solved is that of providing real-time applications with the necessary Quality of Service (QoS) guarantees. Certain real-time applications such as interactive packet voice and video are susceptible to the delay and delay variation. In such applications, cells are considered as "lost" when they are excessively delayed and cell losses may cause significant performance degradation on playback at the receiver. Regardless of whether the losses are due to delays or due to buffer overflows, the impact of the Cell Loss Ratio (CLR) on the traffic streams is important. Given that the QoS requirements can be defined on an individual connection basis, the network performance must be evaluated on a per connection basis as well, but subject to interference by other (pre-existing) connections. In the past, most of the studies have focused on an isolated stage of the network. However, in order to understand the interplay between traffic and quality of service parameters, it is necessary to establish end-to-end network performance models.

The end-to-end model for ATM networks given in Figure 1 is represented as a serial connection of a multiplexer at the front, N intermediate switch nodes, and a demultiplexer at the end. The source station is attached to the multiplexer and the destination to the demultiplexer. Source traffic is modeled by some typical arrival processes, ranging from traditional Poisson processes to self-similar processes. The buffer capacity of each node and the multiplexer/demultiplexer is finite. The multiplexer/demultiplexer and all switch nodes have constant service times, due to the fact that the cell size is fixed. At each node, additional connections may either join (joining cross traffic) or leave (leaving cross traffic) the switch node. As a result, the traffic through the network usually experiences two basic operations, merging and splitting. For example, ATM multiplexers perform a merging operation and demultiplexers carry out a splitting operation. ATM switch nodes perform both merging and splitting operations. It should be noted that arrivals and departures occur in fixed length time slots, where a time slot is defined as the time to transmit one ATM cell. Therefore, discrete time models have been applied for these basic operations to closely capture the behavior of the system [1, 2, 3, 4]. Finally, a reasonable simplification which will also be justified in the following is to assume that all servers are first-come first-served (FCFS).

Despite repeated efforts for the solution of the end–to–end model, it remains a fact that the simulation of an end–to–end model is an indispensable tool for the performance evaluation of systems. In fact, the accuracy of analytical techniques is often evaluated by comparison against simulation results. Unfortunately, the running times for such simulations are extremely long due to the need to study rare events (such as cell losses). Even for a simple network model, the execution

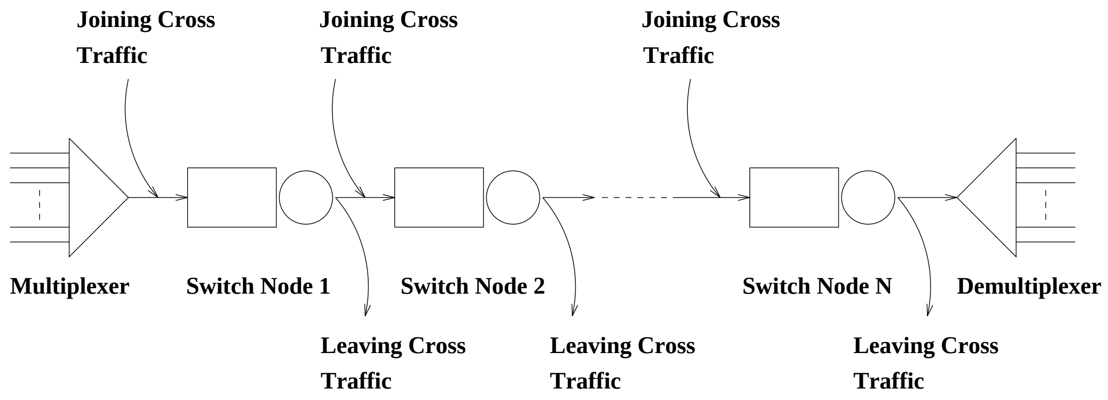


Figure 1: Model of an ATM end–to–end connection.

of a simulation may take several days. In order to develop fast simulations, two techniques will be combined. The first is the exploitation of the specific system topology and of the specific family of bursty arrival processes while the second is a time–division parallel simulation technique. The reason why a parallel simulation technique is selected is because in all sequential event–list based simulations, there exists an inherent limit on how many events can be processed in a unit of time, depending on the event list algorithm. In a cell–level simulation of an ATM network, the event list is accessed every time a cell is generated or consumed. The objective of the first technique is to improve on the limit put by the event list algorithm by developing a burst–level simulation of the network which avoids operating in a cell–by–cell fashion but still provides the necessary cell–level statistics, including the delay distribution.

The second component of the presented solution is the use of *time–parallel* simulation, where each logical process (LP) of the simulates the system for a separate slice of time and it does so for all the components of the network. Therefore, each LP is responsible for the correct calculation of the state of the system at each element of the network for the given slice of time. In doing so, each LP *pipelines* the traffic from one component to the next. That is, contrary to a traditional simulation, each departure from one network element is not directly modeled as an arrival to the next, but rather, a sequence of departures is forwarded into the next element. Evidently, such formulation limits the technique from being applicable to networks with feedback. However, in the context of our study, feedback is not a concern.

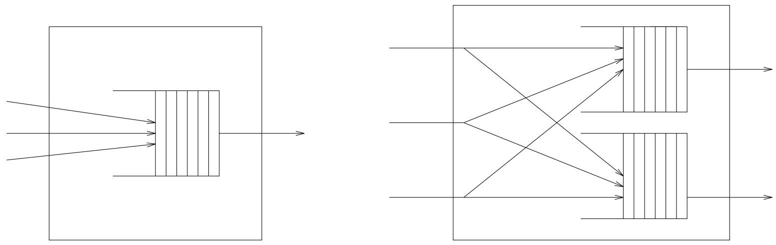
The simulation computation is formulated similarly to [5, 6] as a sequence of tuples, each describing the traffic over an interval of time, and as network elements that operate by performing certain transformations on these sequences of tuples and by producing a sequence of output tuples. We observe that the state of a multiplexer is the number of cells buffered in its output queue. Given that the buffer is finite, we can inspect the sequence of tuples that describe the arrivals and identify the ones that are guaranteed to cause either an overflow or an underflow at the multiplexer. Immediately after any such tuple, a simulation can begin with perfect knowledge of the state of the component, i.e., of the multiplexer queue. This observation allows us to avoid performing fix–ups, that other time–parallel techniques require, at the minor expense of inspecting the trace of tuples, which is both efficient to perform and is completely parallelizable.

It must be underlined that the approach taken in this paper is to evaluate the end–to–end connections not in the sense of providing bounds for the measures of interest, as a large section of recent research has been striving to do. Instead, it provides results in the "traditional" sense, that is, stochastic results, i.e., distributions and event likelihoods. This approach is justified by the fact that to this day the presented bounds in the literature are not tight and they generally reflect particular non work–conserving schedulers and, as a consequence, underutilized networks. Hence, our selection of FCFS as the service discipline, over static or dynamic priority schemes, is consistent with the context of this study and it allows to derive a set of results that represent the operation of the network without the inclusion of particular shaping and other mechanisms that distort the arrival processes.

The paper is organized as follows: Section 2 presents the structure of the simulation model and introduces the particular description of the traffic at the various stages of an end–to–end connection, while also providing the calculations necessary to derive the cell–level performance. In Section 3, we detail our time-division parallel technique with the traffic being virtually pipelined through the entire connection. In Section 4, we present performance results using the simulation technique and some experimental results for a variety of parameters including sources with heavy tailed active periods. Finally, we conclude with a discussion of the possible extensions for the proposed techniques and future research directions.

2 Simulation Model

At the multiplexer, several traffic sources may be fed into a high-speed output link. Each traffic source represents a single call which has already accepted by the network, and is active for the entire duration of the simulation. At subsequent switch nodes, a certain number of incoming links are merged onto one link of equal or higher speed, and split into different links at the same time. Specifically, the links that are used for high-speed transfer between switches are usually called as "trunks". The switches are internally non-blocking, output-buffered fast packet switches. For simplicity, a switching delay of zero is assumed. Transmission link speeds are often specified as integer multiples of a basic rate. Therefore the ratio of the output link speed to a single input link speed can be used to describe an output link speed. If the speed of the input link to the multiplexer is normalized to 1, then all other link speeds are expressed as ratios to the input link speed for each source. For example, if ten STS-3c (Synchronous Transport Signal - 3 concatenated) are



(a) Multiplexer (b) Switch Node

Figure 2: Typical network elements: (a) a multiplexer and (b) a switch.

statistically multiplexed onto one STS-12c link, the fan-in, number of inputs, of the multiplexer is 10 and its output link speed is 4. Similarly, a switch node with two STS-12c input links and an STS-12c output link has a fan-in of 2 and an output link speed of 1.

All network devices including multiplexers/demultiplexers and switch nodes have finite buffers. ATM cells arriving at a network device when the device's buffer is full will result in cell losses. Since transmission lines are assumed to be error-free, all cell losses are due to buffer overflow. Another important aspect is the source traffic model which must be able to capture the bursty nature of various traffic sources, such as voice, video and data traffic. A well–known family of models used in the characterization of bursty arrivals from an individual source is the ON/OFF model which is also used in analytical models [7]. The ON and the OFF periods strictly alternate. During an OFF period, no cells arrive. During the ON period, it is assumed that one cell arrives in each time slot of the input link. However, actual traffic sources do not fit a simple distribution for the length of the ON and OFF periods. As a result, no assumptions will be made concerning the duration of the ON and OFF periods during the development of the simulation technique. Subsequently, the simulation technique can be used for i.i.d. ON and i.i.d. OFF periods derived from arbitrary marginal distributions.

2.1 The Structure of the Model

The building blocks of most networks including ATM networks are multiplexers, demultiplexers, and switch nodes (Figure 2). Multiplexers collect the traffic from a number of incoming links and output the aggregate traffic on an outgoing link. Demultiplexers split the incoming traffic of an input link to several output links according to the destination. Switches are used for routing the traffic from input links to output links. Within a switch, both merging and splitting operations are performed on traffic streams, e.g., an incoming traffic split into two separate outgoing links like a demultiplexer, or the traffic from two incoming links merged into one outgoing link like a multiplexer.

We focus on the two performance measures, that cell loss ratio and the delay distribution. The parameters which influence the performance at a finite buffer multiplexer are:

- The number of input links, N, also called the *fan–in* of the multiplexer.
- The output link rate, C, (also called service rate) normalized relative to the input link rate assuming input links of equal speed.
- The finite buffer size, K, in cells.
- The arriving traffic model.
- The interfering traffic model.

The parameters N, C and K are described by integer values. However, as the next section explains, the description of the arriving, the interfering and the departing traffic is somewhat complex. The following approach for describing the traffic processes has previously been used in [6] where an extensive analysis of the performance of the simulation technique is also presented.

2.2 Traffic Description

2.2.1 Multiplexer Level

Much of the potential for constructing fast parallel simulations lies in the exploitation of the specific structure of the problem. In our scheme, we adopt a *burst-level* description for the traffic arrivals and departures instead of using individual *cell-level* description. Since bursts of cells are simulated instead of individual cells, we eliminate redundant information related to the temporal characteristics for each cell. The more cells in a simulated burst, the better is the overall efficiency compared to a traditional *cell-level* simulation.

The source traffic is described as a sequence of tuples of the form $\langle s_i, d_i \rangle_S$ for the i-th tuple, where s_i is the state of the source (1 for ON, 0 for OFF) and d_i is the duration in input link slot times. The subscript S indicates that the tuple describes source traffic. For example, a tuple of the form $\langle 1, 5 \rangle_S$ describes five consecutive cell arrivals, while $\langle 0, 20 \rangle_S$ represents twenty consecutive empty slots, i.e., no arrivals. A simple indication of the efficiency can be calculated based on the following observation: suppose the average ON period of a source is $E[ON]$ cells long, then on the average for each $E[ON]$ cell arrivals only one tuple need to be generated for the burst level description. Hence the description is particularly compact when compared to a traditional cell-level description. An average of $(E[ON] + E[OFF])$ time slots are represented by just two $\langle s_i, d_i \rangle_S$ tuples.

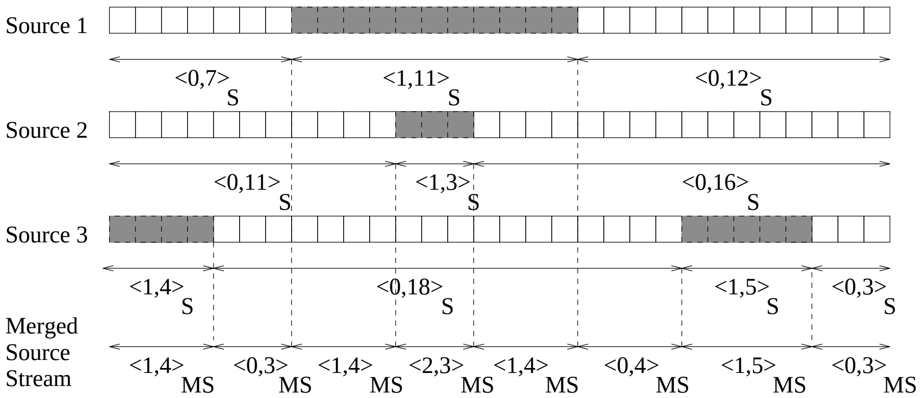


Figure 3: Example of a source traffic merger.

Multiplexers are fed by a number of input streams. From the $\langle s_i, d_i \rangle_S$ description of these streams it is necessary to derive a description of the merged traffic of N streams. In a traditional simulation, this is accomplished by ensuring the time-ordered arrival of the cells from the different streams. However, in our scheme we describe the merged traffic in terms of $\langle a_j, m_j \rangle_{MS}$ tuples, where a_j is the number of sources that are active for m_j slot times. The subscript MS indicates that the tuple describes merged traffic. Consequently, the total number of arrivals represented by a $\langle a_j, m_j \rangle_{MS}$ is $a_j \times m_j$. The relation of the constituting $\langle s_i, d_i \rangle_S$ and the $\langle a_j, m_j \rangle_{MS}$ is depicted in Figure 3. The generation of merged tuples $\langle a_j, m_j \rangle_{MS}$ is obtained by a parallel merge-sort operation of the $\langle s_i, d_i \rangle_S$ tuples. Note that we assume that arrivals from the different source traces are aligned at slot boundaries. By aligning the cell arrivals in this way, the contention for the buffer space is more pronounced and cell losses are more likely to happen. Hence the simulated behavior represents the worst case in terms of cell losses of the actual system.

Consider now the case of the departures from a multiplexer with output link capacity C. An output link capacity of C implies that C cells can be transmitted on the output link in one cell time of the input link. If the queue of the multiplexer is empty and less than C connections are currently active (underload), then arriving cells can immediately depart on the output link. Since the new arrivals are less than C, the queue will still be empty at the end. The same periodic form of departures continues for as long as the number of active sources remains the same. Also, as long as the underload persists, so will the periodic form of departures. For example, in Figure 4(a), all $\langle a_j, m_j \rangle_{MS}$ tuples are underloads. The departure stream can then be described as a sequence of tuples of the form $\langle b_k, c_k \rangle_P$ where b_k is the length of the periodic bursts and c_k the number of cycles. The subscript P indicates that the tuple describes departure traffic. Let us consider another situation shown in Figure 4(b) where an initial queue backlog of four cell exists. Then, for a number of cycles the departures continue at their peak rate. When finally the queue becomes empty, the same scenario as the previous case can apply. Furthermore, departures at

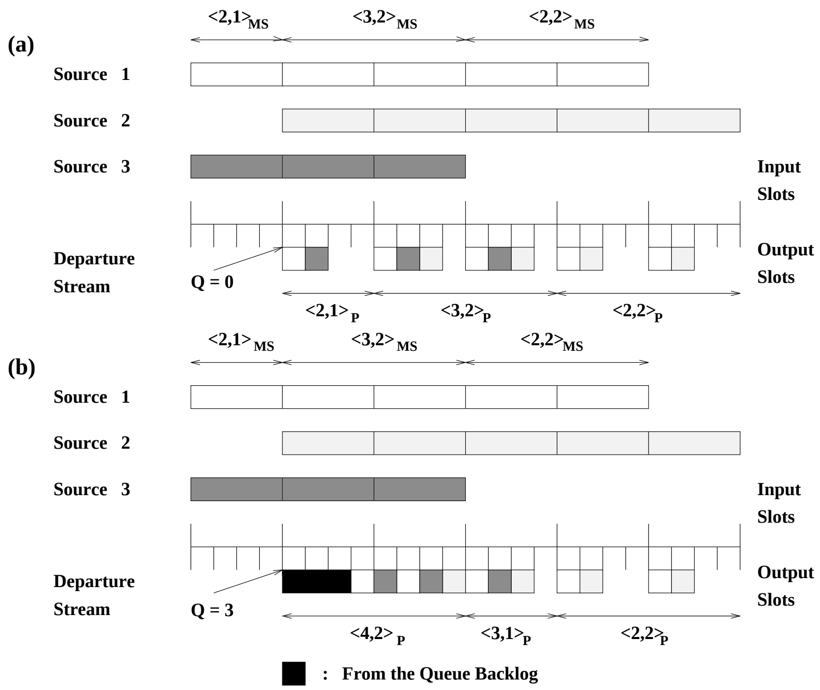


Figure 4: Example of the departure process when C = 4. when (a) the initial queue length is zero. or when (b) the initial queue length is 3.

the peak rate from the backlog or overload can be described as periodic with $b_k = C$.

2.2.2 Switch Level

As shown in Figure 1, at each switch node additional connections may either join or leave the switch node to form an interference traffic. The interference traffic consists of joining and leaving cross traffic. Joining cross traffic is further classified as external (traffic just enter the network) and internal arrivals (traffic relayed from other network nodes). It is reasonable to assume that all external sources are mutually independent except for the particular cases like videoconferencing, which will be reserved for the future work. Similar assumption can be applied for traffic arriving from different switch nodes as well. Therefore, this kind of traffic is also modeled by a superposition of a number of ON/OFF streams, just like the multiplexer traffic by using the same technique as in Section 2.2.1. Additional multiplexers will be needed to generate the joining cross traffic separately from the regular traffic.

On the other hand, leaving cross traffic arises when incoming traffic to the switch is split into several output links shown in Figure 2(b). Except for the traffic forwarding to the next switch in the end-to-end path, all diverted traffic can be regarded to as leaving cross traffic. There are two types of incoming traffic to the switch, one for regular traffic and the other for joining cross traffic.

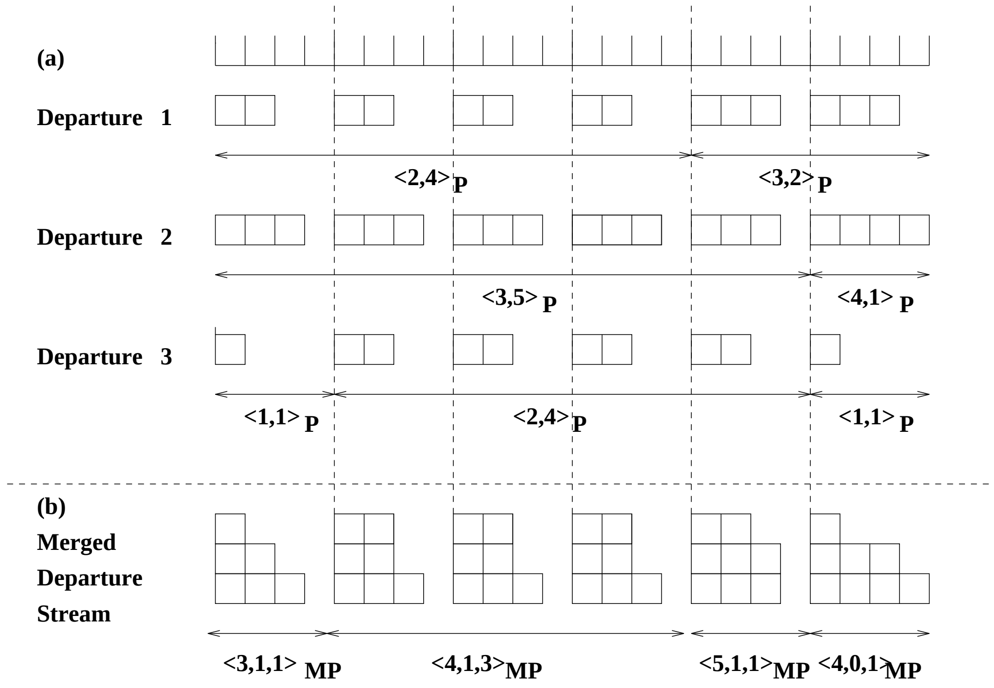


Figure 5: Example of a departure stream merger when the link capacity, C, is 4.

Before merging, the incoming traffic must be split first. For simplicity, we assume a probabilistic splitting of the traffic. That is a certain portion of the incoming traffic to be “leaving” on an average basis. For example, if the splitting ratio is 0.3, average 30% of each type of the incoming traffic forwards to the next stage in the path and the rest 70% is diverted to form a leaving cross traffic. Recall that the incoming traffic is described as departure streams with the tuple format of $\langle b_k, c_k \rangle_P$. Since b_k actually indicates the number of active sources and different sources may have different destinations, the split is performed by assuming a logical separation of the sources into two groups, those who remain with the flow and those who depart. If, for example, the splitting ratio is 0.3, the sources are divided by multiplying b_k with a random variable Y which is obtained from the simple transformation of a uniformly distributed random variable X from the interval [0,1]. Specifically, the random variable X is transformed to Y by multiplying it by 0.6 in order to produce a uniformly distributed Y with an average value of 0.3.

After splitting, departure streams from the previous stage will be merged with a joining cross traffic just like the merge operation at the multiplexer. The merged traffic will then be processed and the new departure traffic generated. An important aspect is the alignment of the periodic streams. The alignment we choose is the one where all the periodic departure bursts start at the same point in the cycle. Figure 5(a) depicts such an alignment. This alignment results in the worst case losses than any other possible alignment.

The resulting merged departure tuples are shown in Figure 5(b). From the periodic property of merged departure tuples, these tuples are expressed in the form of $\langle I_l, D_l, p_l \rangle_{MP}$, where I_l is the overall number of arrivals in excess of the link capacity C at the beginning of the cycle, D_l is the overall available departures (i.e. "slack" capacity) at the end of the cycle, and p_l is the number of cycles during which the same I_l and D_l continue. The subscript MP indicates that the tuple describes merged departure traffic.

2.3 Calculation of the Performance Measures

2.3.1 Calculating the Multiplexer Performance Measures

In order to calculate the cell level statistics of the multiplexer, a set of recurrences is evaluated for each tuple. These recurrences provide the aggregate lost and serviced cells, while keeping track of the queue length of the multiplexer. First, we need to distinguish between $\langle a_j, m_j \rangle_{MS}$ tuples that may result in an increase of the buffer contents, and $\langle a_j, m_j \rangle_{MS}$ tuples that my result in a decrease. We call the former *overload* tuples and the latter *underload* tuples. Specifically:

- A tuple $\langle a_j, m_j \rangle_{MS}$ is an overload tuple if $a_j \geq C$
- A tuple $\langle a_j, m_j \rangle_{MS}$ is an underload tuple if $a_j < C$.

where C is the normalized output link speed. Suppose that the tuple $\langle a_j, m_j \rangle_{MS}$ spans the time between t_j and t_{j+1}, i.e., $t_{j+1} = t_j + m_j$. Then the queue size, Q_{j+1}, at time t_{j+1} and the queue size Q_j at time t_j are related as follows:

$$Q_{j+1} = \begin{cases} min[Q_j + (a_j - C)m_j, K], & a_j \geq C \\ max[Q_j - (C - a_j)m_j, 0], & a_j < C \end{cases} \tag{1}$$

Equation (1) captures the dynamics of the underload and overload conditions as an increase or decrease (respectively) of the accumulated backlog in the buffer. Incoming cells are either serviced or lost. Losses can only occur during overload tuples. The number of lost cells will be the excessive arrivals that cannot be accommodated in the queue. Hence, the losses L_{j+1} at time t_{j+1} can be calculated as:

$$L_{j+1} = \begin{cases} L_j + max[(a_j - C)m_j - (K - Q_j), 0], & a_j \geq C \\ L_j, & a_j < C \end{cases} \tag{2}$$

In any $\langle a_j, m_j \rangle_{MS}$ tuple, m_j represents the time in terms of input time slots. In the same time, Cm_j departures can occur on the output link. While in overload, departures occur back to

```
for each ⟨a_j, m_j⟩_MS do
    for i = 1, ···, m_j do
        l = 1;
        while (q + i) ≤ K and l ≤ a_j do
            Delay[q + l] + +;
            l + +;
        endwhile
        q = min(K, q + a_j);
    endfor
endfor
```

Figure 6: Algorithm for the delay calculation given a sequence of $\langle a_j, m_j\rangle_{MS}$ tuples and an initial queue size of q.

back. Hence, exactly Cm_j cells depart at the output, i.e., as many as output link time slots. In underload, at least $a_i m_j$ cells will be serviced plus any additional number of cells, reflecting the case that the spare output link slots $(C-a_i)m_j$ can clear part of the queue backlog. Consequently, the serviced cells S_{j+1} at time t_{j+1} are:

$$S_{j+1} = \begin{cases} S_j + Cm_j, & a_j \geq C \\ S_j + a_j m_j + min[(C-a_j)m_j, Q_j], & a_j < C \end{cases} \tag{3}$$

The initial conditions are: $S_0 = 0, L_0 = 0$ and $T_0 = 0$. The queue length is also typically set to $Q_0 = 0$. At the end of the simulation of n consecutive $\langle a_j, m_j\rangle_{MS}$ tuples (indexed from 0 to $n-1$), the cell loss ratio, CLR, can be calculated by:

$$CLR = L_n/(S_n + L_n) \tag{4}$$

On the other hand, each cell may experience some delay at the buffer of the multiplexer. Since we assume finite buffers with FCFS discipline, the delay for each cell is totally dependent on the buffer position where the cell is inserted. The buffer position is determined, based on the current queue length and the order of active sources. In particular, when the number of active sources is greater than the output link speed, a special consideration needs to be taken. In this situation, cells will accumulate by the difference between the output capacity and active sources at a time. Eventually, buffer overflow might happen when the accumulating cells exceed the buffer size. At this point, the buffer delay should be accounted up to the buffer size. The delay for each cell at the multiplexer can be calculated with the algorithm shown in Figure 6.

2.3.2 Calculating the Switch Performance Measures

In each cycle, the switch experiences a surge of arrivals totaling I_l cells. During this surge losses may occur. While at the end of the cycle, a backlog of D_l tuples can be serviced. We can use the relation between I_l and D_l to define the regions of operation of the switch node. For example, if $I_l \geq D_l$, then in the long term we expect accumulation in the queue.

The queue size can be described as the following recurrence:

$$Q_{l+1} = \begin{cases} max[0, min\{K - D_l, (I_l - D_l)p_l + Q_l\}], & I_l \geq D_l \\ max[0, q - (D_l - I_l)(p_l - 1)], & I_l < D_l \end{cases} \tag{5}$$

where q is an auxiliary variable that describes the queue length at the end of the first cycle of the periodic behavior. Specifically:

$$q = max\{0, min(K, Q_l + I_l) - D_l\} \tag{6}$$

The recurrence for the calculation of Q_{l+1} captures both the overall trend that the queue will decrease/increase as well as the transient surge at the beginning of each periodic cycle. Note also that the buffer size K plays an important role. For example, if $D_l \geq K$ then at the end of the cycle the queue will be zero, because there is enough slack capacity left to clear the maximum queue backlog. Although from a design standpoint small sizes for K are unlikely, our recurrences allow K to be arbitrarily small.

Similarly, the cell level losses are calculated using:

$$L_{l+1} = \begin{cases} L_l + max[0, \{I_l - min(D_l, K)\}p_l + Q_l + min(D_l - K, 0)], & I_l \geq D_l \\ L_l + max(0, Q_l + I_l - K) + max(I_l - K, 0)(p_l - 1), & I_l < D_l \end{cases} \tag{7}$$

The serviced cells during the p_l cycles are given by:

$$S_{l+1} = \begin{cases} S_l + C_p C_n p_l + min(K - D_l, 0)p_l, & I_l \geq D_l \\ S_l + C_p C_n p_l + min[min(0, K - D_l), Q_l + I_l - D_l] & \\ -max[max(0, D_l - K)(p_l - 1), (D_l - I_l)(p_l - 1) - q], & I_l < D_l \end{cases} \tag{8}$$

The calculations for the cell loss ratio CLR and delay can be performed as in the case of the multiplexer.

2.3.3 Calculating the End-to-End Performance Measures

The ultimate goal of this study is to predict the end-to-end network performance so that the estimation can be used to provide guidelines for call admission control, resource allocation, and buffer dimensioning. There are two important parameters we should obtain in terms of end-to-end standpoint. The first one is the end-to-end CLR for the entire path between the multiplexer and demultiplexer to which the source and the destination station belong respectively. Even if we can easily calculate the total CLR by dividing the total number of lost cells by the total number of cells simulated at the multiplexer and switch nodes each, it might be meaningless in the sense that the total CLR cannot describe the end-to-end characteristic well enough. In fact, the end-to-end CLR is characterized by the worst value among all CLR's from each network element in the path. The modeled end-to-end traffic streams traverse the network element with the worst CLR, so the end-to-end CLR cannot be better than this worst value. For the demultiplexer, since the output link capacity can be represented as the sum of all output links and consequently the total capacity of the output link is always larger than that of the input link, we may expect that there is no possibility of losing cells. Let us assume that N intermediate switch nodes are involved in the end-to-end connection of our concern. The *End-to-End CLR* is computed from:

$$CLR_{End-to-End} = max\{P_{mux}, P_1, \cdots, P_N, P_{demux}\} \tag{9}$$

where:

- P_{mux}: CLR at the multiplexer
- P_{demux}: CLR at the demultiplexer
- P_i: CLR at the intermediate switch node i

Another important parameter is the *End-to-End Delay*, $D_{End-to-End}$, which can be approximated by adding each delay for the multiplexer/demultiplexer and switch nodes. That is, it can be produced as the convolution of the delay distributions along the path of the connection.

$$D_{End-to-End} = D_{mux} + D_1 + \cdots + D_N + D_{demux} \tag{10}$$

where:

- D_{mux}: Delay at the multiplexer
- D_{demux}: Delay at the demultiplexer
- D_i: Delay at the intermediate switch node i

3 The Parallel Simulation Approach

The previous section describes the specific system structure and the relation to the queue dynamics at the each stage. In this section, we determine the conditions that enable the construction of a *time-parallel* simulation. Previous *time-parallel* algorithms described in [8] require an initial guess of the queue length, therefore, requiring subsequent fix-up phases. As the following section explains, fix-up phases are avoided altogether by observing the behavior of the queues.

3.1 The Time-Division Technique

3.1.1 Multiplexer Level

The state of a multiplexer is its queue length. During the merge operation at the multiplexer, two subsets of the time division (TD) points can be distinguished.

- *Guaranteed Overflow* tuples, where:

$$a_j > C \text{ and } m_j \geq K/(a_j - C) \tag{11}$$

- *Guaranteed Underflow* tuples, where:

$$a_j < C \text{ and } m_j \geq K/(C - a_j) \tag{12}$$

That is, overload tuples last for a sufficiently long duration to fill up the multiplexer buffer even if it was empty at the time point just prior to the $\langle a_j, m_j \rangle_{MS}$ tuple. Underload tuples last for a sufficiently long duration to clear the multiplexer buffer even if the backlog in the queue was the entire buffer K just prior to the $\langle a_j, m_j \rangle_{MS}$ tuple.

Once such conditions above are identified, separate and independent simulations can execute in parallel by simply separating the sequence of tuples with TD points and assigning each portion to each processor. Specifically, after each guaranteed overflow, we can start a simulation with the assertion that $Q_0 = K$. Similarly, after a guaranteed underflow we can start a simulation with the assertion that $Q_0 = 0$. If there are n guaranteed overflow/underflow points, we may execute n independent simulations in parallel.

3.1.2 Switch Level

Guaranteed overflows/underflows also exist for intermediate switch nodes based on the $\langle I_l, D_l, p_l \rangle_{MP}$ tuple definition and in a slightly different form. The two sets are:

- *Guaranteed Overflow* tuples, where:

$$D_l = 0 \ and \ p_l \geq K/I_l \tag{13}$$

- *Guaranteed Underflow* tuples, where:

$$I_l < D_l \ and \ p_l \geq K/(D_l - I_l) \tag{14}$$

Similar assertions for Q_0 as in the case of the multiplexer can be applied. In this case, separate and independent simulations can also be performed in parallel, based on the previous Equations (5)-(8) in Section 2.

Let TD density be the probability of finding an overflow or underflow tuple. Even a very small TD density (e.g., as low as .01%) is sufficient for effective use of our method. To illustrate the point, consider a tuple trace of 1,000,000 tuples, which is quite practical and reasonable. If the TD density is just .01%, we can expect on the average .01% * 1,000,000 = 100 TD points in the tuple trace. Hence, for this specific example, we can obtain 100-fold TD parallelism. In the experiments conducted here and in previous performance studies of this algorithm [5, 6], it was observed that with the exception of a very few cases the limitation to parallelism is not the number of TD points, but rather the number of available processors.

Note that the TD density at the switch nodes is a separate issue from the multiplexer because it depends on the specific configuration of the system. For example, if a number of heavily utilized links are merged on a common link of the same capacity as the input links, chances are that many guaranteed overflows exist. Even if the incoming links are not heavily utilized, if there exist sufficiently many of them, the possibility of guaranteed overflows is still high.

3.2 The Algorithm

The entire simulation procedure as seen from the perspective of an LP that operates on a specific time slice can be summarized in the following steps:

1. Generate source traffic for each input link to a multiplexer.

2. Merge all the input sources to the multiplexer.

3. Calculate performance measures at the multiplexer.

4. Generate departure traffic from the multiplexer.

5. Repeat steps 1 through 4 for the joining cross traffic multiplexers.

6. Split the departure traffic of the previous node and join the remaining with the generated cross traffic.
7. Calculate performance measures at the switch.
8. Generate departure traffic from the switch.
9. Repeat steps 6 through 8 until the final switch.
10. Divide the incoming traffic into multiple output links at the demultiplexer.
11. Calculate performance measures at the demultiplexer.
12. Goto 1.

The need for synchronization with the other LP is minimized due to the time division points. Each LP is essentially moving a time slice of the incoming traffic (including the one from the interference sources) through the stages of end–to–end model to the end of the connection, thus forming a "pipeline".

4 Experimental Results

As an experimental setup, one multiplexer, two switches and one demultiplexer are connected serially to establish an end-to-end connection. In addition, two more multiplexers are used to produce the joining cross traffic, one for the multiplexer at the front and the other for the switch nodes. For all switch nodes, we use the same joining cross traffic generated from a multiplexer. The number of input sources to the multiplexer is fixed to 32 and the output link speed of the multiplexer to 16, assuming that the sources are no more than 50% utilized on the average. In order to obtain some basic performance results for the parallel simulator as well as the ATM end-to-end connection, each source of either the observed or the interfering traffic is first modeled as a hyper-geometric ON/OFF source traffic with the following parameters: the average burst period $E[ON] = 28$ cell slots, the burstiness $b = 3.4$ where ($b = 1 + E[OFF]/E[ON]$), and the squared coefficient of variation equal to 4 for both $E[ON]$ and $E[OFF]$.

The speed of the simulation program is measured as the number of cell arrivals to the model in a second of wall-clock time. A highly optimized sequential version of a single multiplexer simulator has achieved in the past a performance of approximately 140 thousand cells per second on a Sun Sparcstation 10. The implementation in the current study has reached 300 thousand cells per second on a four processor Sun Sparcstation under interfering workload from other users. Since the focus of the current study is not on the simulator performance but on the produced results, it is briefly indicated, that as noted in past studies [5, 6], the limit on the parallelism was not the existence of enough time division points, but rather the number of available processors.

The simulator provides two important simulation results about the CLR and delay associated with the end-to-end problem, which can be used as engineering guidelines. The first can be seen in Figure 7 where it is shown how CLR varies with the the speed of the trunk links between the switches when the network access multiplexer buffer size and its multiplexing speed remain constant. According to the definition of the CLR that we proposed earlier, for our reference connection which spans over two switches in the interior of the network, the CLR experienced by the user is the highest value of the three curves shown in Figure 7. Hence, for a normalized trunk speed less than 19, the dominant CLR is that at the first switch. Moreover, systematically, the switch closer to the interior of the network (the second switch in the example) demonstrates a better CLR performance than the one closer to the access point. This indicates that if we engineer the trunk speeds such that the first switch does not exceed the CLR at the access multiplexer, then the switches towards the interior of the network are guaranteed not to exceed this CLR as well.

By increasing the trunk speed in the interior of the network in comparison with the access multiplexer link speeds, the CLR figures can be drastically improved. What is really important is that only a modest increase in the trunk speeds brings the desirable result. In the case of Figure 7, increasing the normalized trunk speed from 16 to 19 improves the CLR several orders of magnitude (note that the y axis is logarithmic). Figure 8 provides a view at another interesting point. Namely, it shows that the increase of the buffer at the access multiplexer brings a minuscule improvement relative to the improvement brought by increasing by the same amount the buffer size in the internal switches of the network.

Given the attention on the issue of self–similar traffic it is interesting to note the counter–intuitive results that Figure 9 presents. In Figure 9, the hypergeometric ON/OFF source processes are replaced by ON/OFF processes where the ON period is taken from a Pareto distribution but whose mean is the same as in the case of the hypergeometric model. The OFF period is geometrically distributed with the same mean as in the previous model. Thus, the utilization of each input source link remains at roughly 30 % as in the previous experiments. One would expect that these Pareto-ON sources would aggregate asymptotically to an FGN (Fractional Gaussian Noise) and hence the particularly bad performance of queues under self–similar traffic would develop. However, the examples presented herein indicate that one must be carefully with the fact that the result has value as an *asymptotic* result. For the experiments considered here, no more than 32 sources multiplex at any multiplexer. As a result, the CLR performance at the first switch presented in Figure 9 is almost identical (if not slightly better for the Pareto-ON) for the two source models.

It has to be underlined that the multiplexers operate at a small buffer regime. The small buffers result as "shock absorbers" to the occasional arrival of a lengthy ON period from the Pareto-ON sources which, if not for the small buffer, would result in the gradual queue buildup which is typical feature of self–similar traffic. To add more insight to this point as well as to the difference between the CLR performance at the different hops of the connection, we consider the delay distributions as presented in Figures 10 and 11.

Figure 10 is produced using the familiar hypergeometric ON/OFF model. It can be seen that the deeper a connection goes into the network (that is, the more the switches) the less the variance in the tail of the delay distribution. The observation is supported by the fact that the switches in the inside of the network are primarily fed by traffic which has already passed from a constant service time server. Hence, the arrivals have been effectively "serialized" one after the other and their burstiness has been smoothed. Only traffic joining the switch from its access multiplexer is still more bursty but such traffic is rarely added directly to a switch in the core of the network, i.e., the switch almost always received traffic that has gone through a few steps of switching.

Finally Figure 11 serves as the explanation to why the heavy tailed ON periods do not add any exaggeration to the CLR for the particular scenario. What is happening is that when lengthy bursts arrive at the multiplexer, a good portion of them is dropped. Hence, the bursts that arrive after a lengthy burst and after some time has elapsed (and some of the queue backlog has been serviced) are unlikely to be as big as the previous one due to the fact that the Pareto distribution has its bulk close to the origin (accounting for the shift). The new arrivals find the multiplexer away from an overflow state and are thus dutifully admitted to the queue. Note that this particular behavior is dependent on the Pareto distribution but it nevertheless illustrates that the consideration for a self–similar model must be made with care both in recognizing that it is an asymptotic behavior from the aggregation of heavy tailed processes and in appreciating that the operating buffer regime can produce results surprisingly similar to those derived from previously used models.

5 Conclusions

A simulation approach has been presented for the end-to-end performance evaluation of an ATM network consisting of a potentially large number of multiplexers and switch nodes. The key to the simulation technique is the of a *burst-level* tuple description of the arrival traffic and a *time-parallel* technique for parallelizing the execution. Implementations of this technique on a SUN Sparcstation of four processors show a promising speed-up by allowing massively parallel simulation for ATM networks. The performance results obtained for an ATM end-to-end connection demonstrate

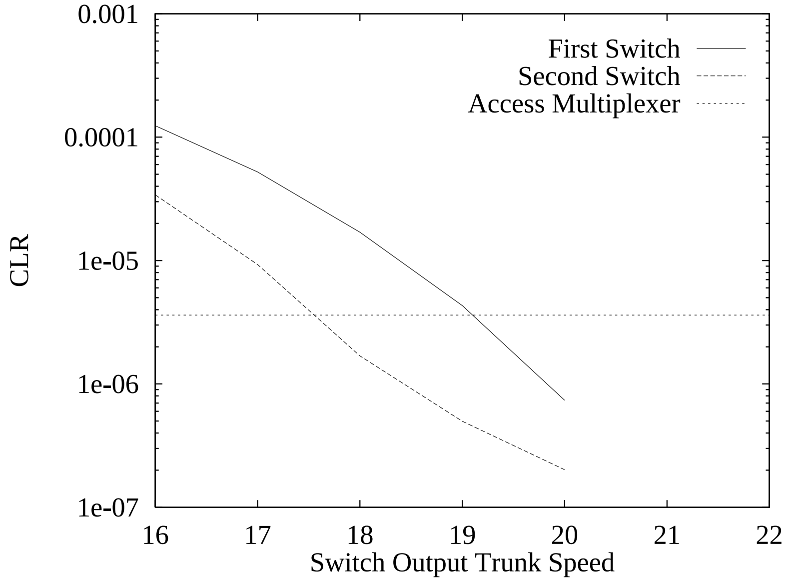


Figure 7: CLR of the first and second switch for variable switch trunk speed versus the CLR at the access multiplexer. The buffer size is 100 cells at each multiplexer and each switch.

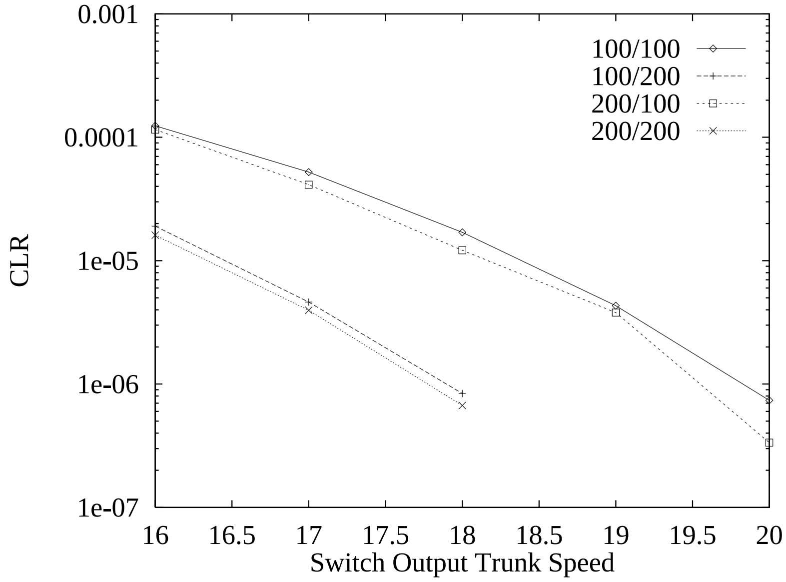


Figure 8: CLR for different buffer size selection. In every configuration X/Y the X is the access multiplexer buffer size and Y is the interior switch element buffer size.

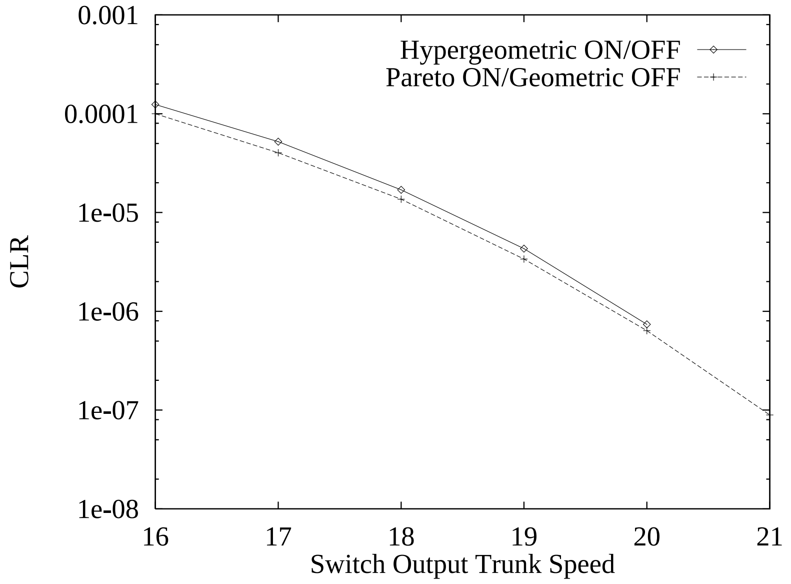


Figure 9: CLR for two different arrival traffic models. The buffer size at the switches and the multiplexers is equal to 100 cells.

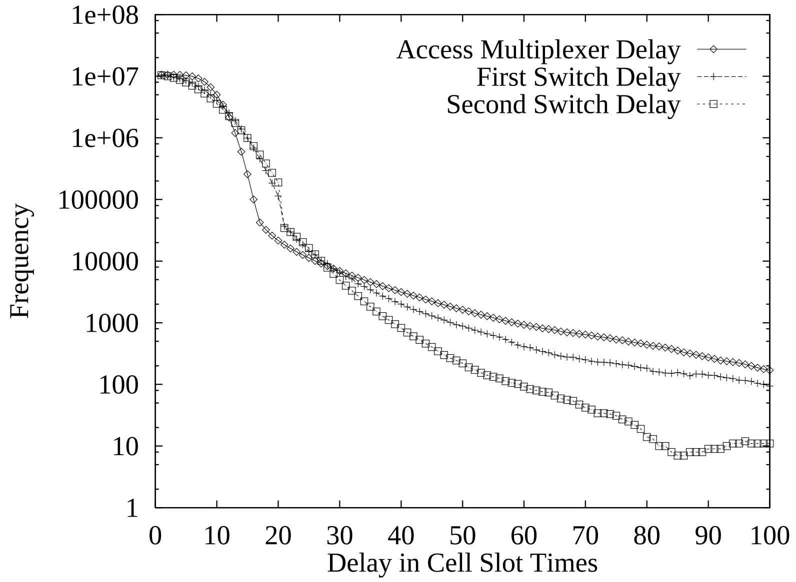


Figure 10: Delay distribution frequency histogram at different hops of a connection. The buffer size at the switches and the multiplexers is equal to 100 cells. The multiplexer speed is 16 and the trunk speed is 20.

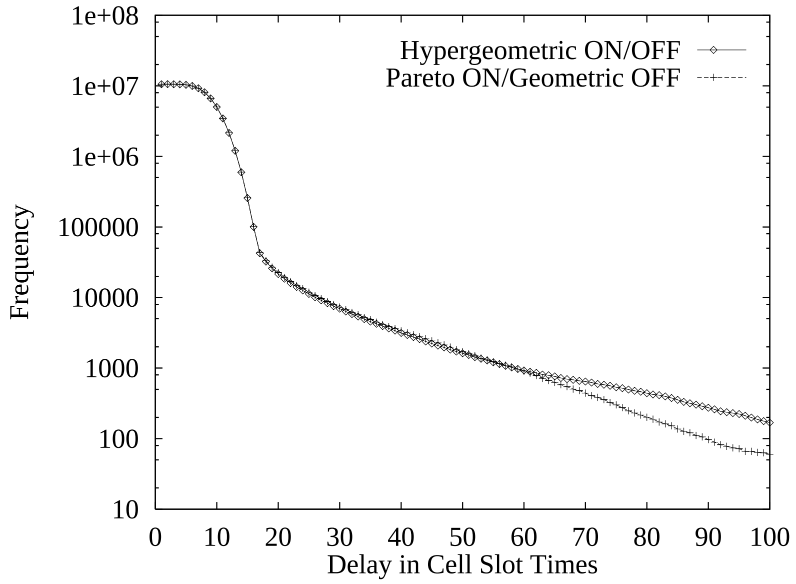


Figure 11: Delay distribution frequency histogram for two different arrival traffic models. The buffer size at multiplexer is equal to 100 cells. The multiplexer speed is 16.

the impact of the number of hops traversed by a connection on the end-to-end CLR and delay distribution. It is also demonstrated that the heavy tailed traffic processes do not necessarily result in substantially different performance from previous models in certain contexts, and namely when operating in the small buffer regime.

It is demonstrated that there is always benefit in the proper dimensioning of the trunk speeds and the buffer sizes in the interior of the network that surpasses by far any similar efforts at the boundary of the network. In the future, we plan to extend the model to support cross–correlation between traffic connections in order to capture the performance of multi-party communications like videoconferencing. Any such result will be of high value because of the unmanageable complexity of any known analytical approach. The impact of such correlation on the end-to-end performance must be clearly identified because the high bandwidths involved (for video communication), can significantly influence the performance of an entire network.

References

[1] I. Stavrakakis, "Efficient Modeling of Merging and Splitting Processes in Large Networking Structures," *IEEE Journal on Selected Areas in Comm.*, vol. 9, no. 8, Oct. 1991.

[2] V. J. Friesen and J. W. Wong, "The Effect of Multiplexing, Switching and Other Factors on the Performance of Broadband Networks," *IEEE INFOCOM*, pp 1194-1203, 1993.

[3] B. Maglaris *et al*, "Performance Models of Statistical Multiplexing in Packet Video Communications," *IEEE Trans. on Communications*, vol. 36, no. 7, pp. 834-844, July 1988.

[4] H. Saito *et al*, "An Analysis of Statistical Multiplexing in an ATM Transport Network," *IEEE Journal on Selected Areas in Comm.*, vol. 9, no. 3, pp 359-367, April 1991.

[5] I. Nikolaidis *et al*, "Parallel Simulation of High-Speed Network Multiplexers," *Proc. of the 32nd IEEE Control and Decision Conference*, pp. 2224-2229, 1993.

[6] I. Nikolaidis *et al*,"Time-Parallel Simulation of Cascaded Statistical Multiplexers," *Proc. of the ACM SIGMETRICS*, pp. 231-240, 1994.

[7] H. Heffes and D. M. Lucantoni, "A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance," *IEEE Journal on Selected Areas in Comm.*, vol. 4, no. 6, pp. 856-868, Sep. 1986.

[8] Y. B. Lin and E. D. Lazowska, "A Time-Division Algorithm for Parallel Simulation," *ACM Trans. on Modeling and Computer Simulation*, vol. 1, no. 1, pp. 73-83, Jan. 1991.

[9] W. E. Leland *et al*, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," *IEEE/ACM Trans. on Networking*, vol. 2, no. 1, pp. 1-15, Feb. 1994.

[10] W. C. Lau and S. Q. Li, "Traffic Analysis in Large-Scale High Speed Integrated Networks: Validation of Nodal Decomposition Approach," *Proc. of the IEEE INFOCOM*, pp. 1320-1329, 1993.

[11] J. F. Ren *et al*, "End-to-End Performance in ATM Networks," *Proc. of the ICC*, pp. 996-1002, 1994.

[12] Y. Ohba *et al*, "Analysis of Interdeparture Processes for Bursty Traffic in ATM Networks," *IEEE Journal on Selected Areas in Comm.*, vol. 9, no. 3, pp. 468-476, April 1991.