

FEASIBILITY OF FLUID EVENT-DRIVEN SIMULATION FOR ATM NETWORKS

G. Kesidis, A. Singh, D. Cheung and W.W. Kwok
 E.& C.E. Dept, University of Waterloo
 Waterloo, ON, Canada, N2L 3G1
 email: {g.kesidis,asingh}@eandce.uwaterloo.ca

Abstract— We describe an ATM network simulator that uses Markov-modulated fluid models for the sources as well as fluid leaky buckets and fluid bandwidth schedulers. The generalized processor sharing (GPS) and “idling” bandwidth schedulers are described. We argue for the use of the idling scheduler over GPS. Based on this fluid model, a simulator for ATM networks has been developed. The simulator employs the well-known discrete event-driven approach. Finally, simulation results are given that, in particular, compare the performance of fluid and “cell-level” simulators. The experimental results indicate that while the fluid simulator is much faster for ATM networks with certain characteristics, some key issues still need to be addressed to widen the applicability of this approach.

Subject Areas: Modelling and Simulation Techniques, ATM Systems and Networks

I. INTRODUCTION

B-ISDN based on ATM is mandated to provide individual quality of service (QoS) guarantees to a wide variety of traffic types, e.g., video, voice and data. The precise QoS received by certain connections is not presently calculatable even if Markovian models of the traffic sources are given. Accurate estimates of, for example, the precise traffic capacity of a particular network, can be obtained by simulation.

ATM networks are generally simulated using the cell-level approach whereby the simulator keeps track of each cell emitted by the traffic sources [7]. This paper deals with an alternative approach to ATM network simulation that is based on “fluid” source models (see, e.g., [3]). Instead of keeping track of a potentially enormous number of cells in the network (as in a cell-level simulator), a fluid simulator keeps track of the *rate* at which each source is currently emitting cells. The advantage of the fluid approach is that a potential reduction in the number of events that must be handled would result in a faster simulation as compared to the cell-level approach (source rates typically change over a longer time scale than a cell transmission time). However, fluid models for both traffic sources and the bandwidth schedulers used at the switches are required (like the GPS fluid analogue of the PGPS bandwidth scheduler [9]).

This paper is organized as follows. In §II, we describe the dynamics of the objects involved in a fluid ATM network model. In §III we justify our choice of an idling bandwidth scheduling policy (over GPS). In §IV, we specify performance quantities to be measured. In §V, the structures of our fluid and cell-level simulators are discussed. In §VI, simulation results are given that compare

the performance of cell-level and fluid simulations. The strengths and weaknesses of the two types of simulators are discussed in the light of these results. Finally, §VII presents some of the research directions that could be pursued to make the fluid model suitable for simulation of a wider variety of ATM networks.

II. FLUID DYNAMICS OF NETWORK DEVICES

Our model of an ATM network consists of output-buffer switches connected by links of various lengths. The common bandwidth of all the ATM network links is c cells/s. A fluid-based bandwidth scheduling policy is used at each output port of each switch where a processor sharing node (PSN) resides. We assume that the switch fabrics are nonblocking (no fluid is lost therein) and that they have constant propagation delay. At the user-network interface (UNI) a fluid version of a leaky bucket (FLB) policer/shaper may be present. The sources are Markov-modulated fluids (MMF) [2], [3].

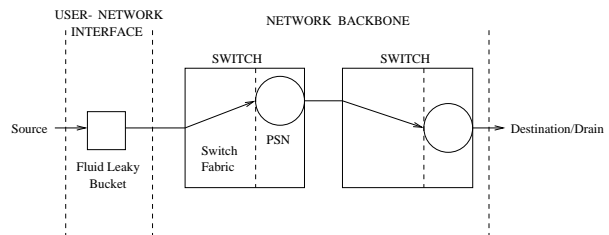


Fig. 1. ATM Network Model

Consequently, from a connection’s point of view, the network consists of a FLB followed by a number (possibly zero) of fluid bandwidth schedulers all separated by constant propagation delays (link or link plus switch fabric), as shown in Figure 1. These tandem devices make up the virtual circuit of the connection. In this section, the dynamics of these devices are described. We assume that only one connection uses a particular FLB.

A. Markov-Modulated Fluid Sources

Associated with each MMF source are a stationary, continuous-time Markov chain, $\{s(t) \mid t \geq 0\}$, with $n \times n$ transition rate matrix Q on the state space $S = \{1, 2, \dots, n\}$, and a rate function $R : S \rightarrow [0, \infty)$. The rate at which fluid is emitted from the source at time t is $R(s(t))$ cells/s. The rate function R need not be one-to-one; so the source may change its state without changing

the rate at which it emits cells. We assume that $R \leq c$ for all sources.

For an example we give a two state MMF model for voice (see, e.g., [8], p. 19,20):

$$Q = \begin{bmatrix} -1.54 & 1.54 \\ 2.84 & -2.84 \end{bmatrix}$$

$$R = \begin{bmatrix} 0 & 62.5 \end{bmatrix}.$$

A MMF model for a video teleconferencing source is given in §VI of [3].

B. Fluid Leaky Buckets

A FLB, handling a single source, is located at the UNI. A FLB has two functions: to act as an enforcer of stated traffic descriptors and/or to act as a traffic shaper.

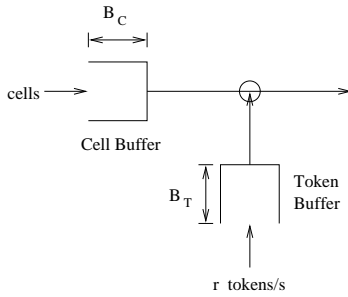


Fig. 2. A Fluid Leaky Bucket

A schematic of a FLB is given in Figure 2. There are three parameters associated with a FLB: the cell buffer capacity of B_C cells, the token buffer capacity of B_T tokens, and the token arrival rate r tokens/second¹. Let $a(t)$ be the arrival rate of fluid to the cell buffer, $d(t)$ be the departure rate of fluid from the leaky bucket, $X(t)$ be the occupancy of the cell buffer, and $T(t)$ be the occupancy of the token buffer.

In order for fluid to leave the cell buffer, it must “consume” an equal amount of fluid from the token buffer. Thus, fluid accumulates in the token buffer only if the arrival rate to the cell buffer is $< r$. Also, fluid accumulates in the cell buffer only if the arrival rate to the cell buffer is $> r$. The FLB operates so that $X(t)T(t) = 0$ for all t [1].

More precisely, the dynamics of the FLB are given by the following equations:

$$\dot{X} = (a - r)\mathbf{1}\{0 < X < B_C\} + (a - r)^+\mathbf{1}\{X = 0\} + (a - r)^-\mathbf{1}\{X = B_C\} \quad (1)$$

$$\dot{T} = (r - a)\mathbf{1}\{0 < T < B_T\} + (r - a)^+\mathbf{1}\{T = 0\} + (r - a)^-\mathbf{1}\{T = B_T\} \quad (2)$$

$$d = a\mathbf{1}\{X = 0\} + r\mathbf{1}\{X > 0\} \quad (3)$$

where the dependence on time of the quantities involved is not shown, $(y)^+ := \max(0, y)$, $(y)^- := \min(0, y)$, and $\mathbf{1}$ is the indicator function.

¹In the leaky bucket literature, the variables σ and ρ are commonly used instead of B_T and r respectively.

C. FIFO Queues of a Generalized Processor Sharing Node

See Figure 3 for a schematic of a switch output port. GPS [9] divides the output link bandwidth (of c cells/s) among a set of N FIFO queues according to positive bandwidth partitioning parameters $\{\phi^1, \phi^2, \dots, \phi^N\}$ satisfying $\phi^1 + \phi^2 + \dots + \phi^N \leq 1$. Each FIFO queue typically handles a group of connections. GPS is work-conserving and operates so that the bandwidth allotted to FIFO queues that are currently idle is distributed among nonidle queues *in proportion to their bandwidth partitioning parameters*.

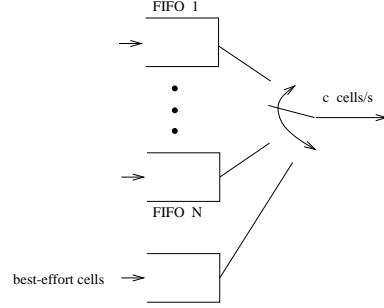


Fig. 3. A Processor Sharing Node

More precisely, the dynamics of a GPS node are describe by the following equations. For all $i, j \in \{1, 2, \dots, N\}$, consider one FIFO queue of a GPS node and let $a^i(t)$ be its arrival rate, $d^i(t)$ be its departure rate, $X^i(t)$ be its contents in cells, and B^i be its capacity in cells.

$$\dot{X}^i = (a^i - d^i)\mathbf{1}\{0 < X^i < B^i\} + a^i\mathbf{1}\{X^i = 0\} + (a^i - d^i)^-\mathbf{1}\{X^i = B^i\} \quad (4)$$

$$\frac{d^i}{d^j} = \frac{\phi^i}{\phi^j} \text{ if } X^i > 0 \text{ and } X^j > 0 \quad (5)$$

A statement equivalent to Equation (5) is: for all $i \in \{1, 2, \dots, N\}$,

$$d^i = \frac{\phi^i \mathbf{1}\{X^i > 0\}}{\sum_{j=1}^N \phi^j \mathbf{1}\{X^j > 0\}} c \quad (6)$$

taking $\frac{0}{0} = 0$.

D. FIFO Queues of an Idling Node

We now describe the dynamics of a non-work-conserving or “idling” bandwidth scheduler. For this scheduler, Equation (4) continues to hold but the “idle bandwidth” (i.e., the bandwidth due to FIFO queues being temporarily idle) is not distributed to nonidle FIFO queues.

Consider an idling node FIFO queue with occupancy X cells, capacity B cells, total arrival rate a , total departure rate d , and bandwidth partitioning parameter ϕ . We simply get that

$$\dot{X} = (a - d)\mathbf{1}\{0 < X < B\} + (a - d)^+\mathbf{1}\{X = 0\} - d\mathbf{1}\{X = B\}$$

$$d = \phi c \mathbf{1}\{X > 0\}.$$

Now let n be the number of connections sharing this FIFO queue. For connection j let: a_j be its arrival rate, d_j be its departure rate, and L_j be its total cell loss to date. The following relationships clearly hold:

$$a = \sum_{j=1}^n a_j$$

$$d = \sum_{j=1}^n d_j \leq \phi c.$$

Assume that at time τ the arrival rate changes for a connection and that $X(s) > 0$ for all $s \in [\tau, t := \tau + X(\tau)(\phi c)^{-1}]$. Therefore, this change in the arrival rate at the input will take $X(\tau)(\phi c)^{-1}$ seconds to propagate to the output of the FIFO queue:

$$d_j(t) = \frac{a_j(\tau)}{a(\tau)}.$$

III. CHOICE OF THE IDLING BANDWIDTH SCHEDULER

We give two reasons for choosing the idling bandwidth scheduler over GPS.

A. Ripple Effect

Consider a network FIFO queue denoted by \mathbf{F} . A change in an arrival rate to \mathbf{F} may cause its departure rate to change which may, in turn, cause some of the departure rates of FIFO queues downstream to change as well. Under the idling bandwidth scheduler, only those downstream nodes for which \mathbf{F} is a tributary are affected by changes in the departure rate of \mathbf{F} . Thus, a change in arrival rate of a source will cause a “ripple effect” in the network.

Under a work-conserving scheduler like GPS, when \mathbf{F} becomes empty (or its departure rate changes when $d < \phi c$), the departure rates of all nonidle FIFO queues *at the node containing \mathbf{F}* will change. Consequently, all upstream FIFO queues which handle connections that use the node containing \mathbf{F} may be affected by changes in the arrival rate to \mathbf{F} . So there will be a greater ripple effect in a network of work-conserving nodes than in one consisting of idling nodes. Consequently, a network of idling nodes will have faster execution time.

B. Interpreting Fluid Simulation Results

Given delay measurements from a network of GPS nodes, one can find bounds on delay of a “corresponding” PGPS network via Theorem 1 of [9]. These delay measurements are not as meaningful to a network of, for example, VirtualClock [10] nodes. However, using the “minimum-bandwidth property”² [4], one can use the delay measurements of a fluid network of idling nodes to find bounds on delay through a cell-level network using a variety of bandwidth schedulers [5].

²A minimum-bandwidth property can be found for a variety of bandwidth scheduling policies [4].

IV. PERFORMANCE MEASURES

We describe both customer and network oriented performance measures.

A. Measures of Received Quality of Service

Consider a connection passing through H network nodes (Idling FIFO queues or FLBs). Let $L^h(t)$ be the total cell loss experienced by this connection at node $h \in \{1, 2, \dots, H\}$ in $[0, t]$. Define $L(t) = \sum_{h=1}^H L^h(t)$. Also for this connection let $R(t)$ be the amount of fluid received at the destination and $T(t)$ be the amount of fluid transmitted by the source in $[0, t]$.

A.1 Average End-to-End Cell Loss Probability

Clearly the average end-to-end cell loss rate over $[0, t]$ for this connection is $L(t)/T(t)$.

A.2 Average End-to-End Delay

Define $\hat{T}(t) = T(t) - L(t)$. The average end-to-end delay of the connection over $[0, t]$ is given by

$$\frac{1}{R(t)} \int_0^{R(t)} \left(R^{-1}(s) - \hat{T}^{-1}(s) \right) ds$$

V. STRUCTURE OF CELL-LEVEL AND FLUID SIMULATORS

For our studies, two simulators, based on cell-level and fluid techniques, have been implemented. At a higher level of abstraction, both simulators use the same discrete event-driven approach. However, the two simulators differ drastically in terms of the nature of events involved and their processing.

In a discrete event-driven approach, the simulator uses an event list which is simply a stack of jobs that the simulator must execute. For the cell-level simulator, examples of jobs are: “cell is transmitted by source to node”, “cell joins buffer”, and “buffer transmits a cell onto link”. The jobs have an associated time at which they must be executed; in the event list, jobs are stacked in increasing order of their time of execution. The program functions by simply popping jobs from the event list and executing them; note that the execution of a job may spawn more jobs that must then be inserted into the event list.

In the fluid model, the simulator does not keep track of individual cells. Rather, it keeps track of the rate at which cells flow out of the sources and various nodes. Therefore, examples of events in this case are: “the source S changes its output rate from X to Y ”, “the buffer of FIFO queue F is full”, etc. The processing of such events is more complicated than the events in the cell-level simulator. Execution of a single event at a node may cause spawning or cancellation of several other events. For example, a change in the cell arrival rate to a FIFO queue may alter the cell arrival rates to upstream FIFO queues. In addition, it may spawn events related to “boundary conditions,” e.g., the time when the queue would become full or empty. Spawning of such an event for a queue may also require cancellation of any similar boundary condition event for the

same queue, if one exists on the event list. For the sake of keeping this paper within reasonable size, we do not go into the description of all the events necessary for implementing a fluid simulator. However, a complete catalog of events required for our fluid simulator is given in [6].

Both the cell-level and the fluid simulator allow the user to specify the structure of the ATM network to be simulated, and characteristics of its components (delays, output rates, buffer sizes, etc.). The simulators have been implemented using the C++ language and operate under the Unix operating system.

VI. PERFORMANCE EVALUATION

One of our chief objective was to assess the relative efficiency of the fluid simulation model. Experiments were conducted to assess the efficiency of the fluid and cell-level simulators. In each of the experiments, conducted using a Sun Sparcstation, source traffic for a fixed duration was simulated. The CPU time required to simulate this traffic was measured. Figures 4, 5, and 6 show the networks that were used for these experiments.

Figure 4 shows a simple network consisting of a single FIFO queue connected to a voice source and drain. The results of experiments using this network are shown in Table 1. For each experiment, the source generates traffic for a fixed duration (3600 seconds in this case). The matrices Q and R , given in section 2.1, describe the characteristics of this source. The bandwidth of the FIFO queue was varied to change its utilization (for a FIFO queue, utilization is defined as the fraction of time that the queue is not empty.).

Experiments similar to the above were done using the network shown in Figure 5 where the FIFO queue is connected to four voice sources and four drains. The Table 2 shows the results of these experiments.

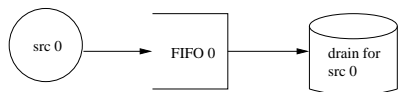


Fig. 4. The Example Networks 1

The results indicate that while the number of events processed by the cell-level simulator is fixed, the number of events processed by the fluid simulator decreases with the increasing bandwidth. In the case of the cell-level simulator, the number of events processed is roughly determined by the number of cells emitted by the sources, and the number of components visited by cells before arriving at their respective drains. However, for the fluid simulator, the number of events processed depends on the source rate transitions as well as events indicating transitions at various components, such as the FIFO queues and drains. Consequently, the cell-level simulator has a fixed but high CPU time requirement irrespective of the bandwidth of the FIFO queue. On the other hand, for higher bandwidths of the FIFO queue, fewer state transitions occur at

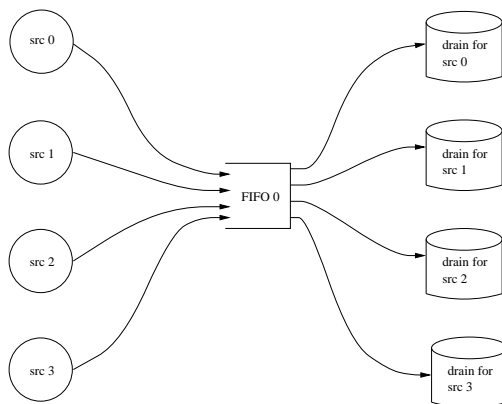


Fig. 5. The Example Networks 2

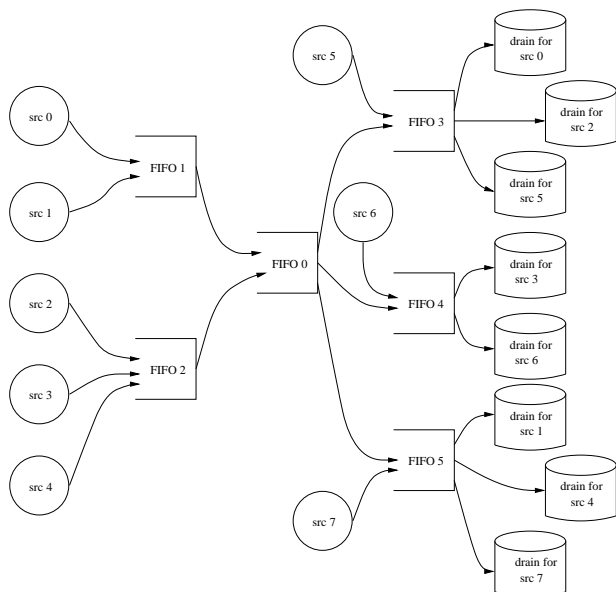


Fig. 6. The Example Network 3

the FIFO queue. This results in a decrease in the overall number of events processed by the simulator and a smaller CPU time requirement. For the network in Figure 4, the fluid simulator is about 2.99 to 4.26 times faster than the cell-level simulator depending upon the bandwidth of the FIFO queue. For the network in Figure 5, the speed up achieved by the fluid simulator over the cell-level simulator ranges between 3.24 to 5.26.

FIFO BW	# of Events		FIFO Use	CPU Time	
	Fluid	Cell		Fluid	Cell
8000	33875	317508	0.99631	3.71	11.10
12000	32490	317508	0.92123	3.73	11.10
15000	31473	317508	0.81749	3.62	11.10
20000	30224	317508	0.64119	3.37	11.10
30000	28949	317508	0.42896	3.22	11.10
50000	25019	317508	0.25737	2.61	11.10

FIFO BW	# of Events		FIFO Use	CPU Time	
	Fluid	Cell		Fluid	Cell
42000	162083	1280000	0.999266	18.81	60.9
48000	160215	1280000	0.986854	18.6	60.9
52000	158788	1280000	0.952458	18.45	60.9
56000	157642	1280000	0.898724	17.97	60.9
80000	136285	1280000	0.631527	14.35	60.9
100000	134094	1280000	0.505232	14.73	60.9
200000	100197	1280000	0.252616	11.58	60.9

Further experiments using the network shown in Figure 6 were conducted to compare the efficiency of the two simulators. Table 3a, and 3b show the results of these experiments. In these tables, the FIFO queue 0 is designated as WANFIFO, whereas FIFO queues 1, 2, 3, 4, and 5 are designated as LANFIFOs. In all these experiments, the source traffic generated during a fixed duration (3600 seconds) is simulated. The Table 3b shows the results when the output data rate of all the sources is increased 10 times. To allow the FIFO queues to function meaningfully, their bandwidths are also increased by 10 times.

Bandwidth		# of Events		CPU Time	
LAN	WAN	Fluid	Cell	Fluid	Cell
23744	42400	958826	4240501	135.31	187.44
33920	84800	496654	5428307	90.25	228.35
42400	84800	607617	5790225	82.00	246.11

Bandwidth		# of Events		CPU Time	
LAN	WAN	Fluid	Cell	Fluid	Cell
237440	424000	909007	40847715	169	2004

As shown in Table 3a, while using the network in Figure 6, the speed up of the fluid simulator over the cell-level simulator lies in the range of 1.39 to 3.00, which is smaller than the speed-ups achieved for the previous example networks. This can be easily understood on the basis of the ripple effect. This network differs from the earlier examples in the sense that cells from each source go through three FIFO queues before arriving at their respective drains. Any transition in the output rate for a source causes events at the LANFIFO queue connected to this source. Execution of this event would spawn other events for the WANFIFO which in turn spawns events for the LANFIFO queues connected to the drains. This causes an overall increase in the number of events processed by the fluid simulator, resulting in a less pronounced speed-up of the fluid simulator.

The impact of higher output data rate on simulation speed is illustrated by the results in Table 3b. An increased data rate results in an approximately proportional increase in the number of events that need to be processed by the cell-level simulator. However, in the case of a fluid

simulator, it does not result in any significant change in the number of events since the number of rate changes remain the same. Therefore, in this case, the fluid simulator is much faster than the cell-level simulator.

In summary, the fluid simulator performs better if the sources maintain relatively constant and high data output rates. For networks where a large number of components are connected in series, the fluid simulator's performance advantage over a cell-level simulator may erode due to the ripple effect. For such networks, the cell-level simulator may perform better than the fluid simulator.

VII. FUTURE WORK AND CONCLUSIONS

Researchers have previously investigated the fluid-based techniques for the purpose of analytical modeling of communication networks. To the best of our knowledge, this is the first study that experimentally assesses the relative pros and cons of using fluid simulators for ATM network simulations.

The fluid model provides a new approach to simulation which in some cases could result in a significant speed-up. Further research is required to enhance the applicability of the approach. Approximate fluid models that effectively contain the ripple effect but at the same time provide useful first order performance estimates for ATM networks need to be investigated.

ACKNOWLEDGEMENTS

We thank Anthony Hung for his helpful comments during this work.

REFERENCES

- [1] V. Anantharam and T. Konstantopoulos. Optimality and interchangeability of leaky buckets. In *32nd Allerton Conference, Monticello, IL*, pages 235–244, October 1994.
- [2] D. Anick, D. Mitra, and M. M. Sondhi. Stochastic theory of a data-handling system with multiple sources. *Bell Sys. Tech. J.*, Vol. 61, No.8:pp. 1871–1894, 1982.
- [3] A.I. Elwalid and D. Mitra. Effective bandwidth of general Markovian traffic sources and admission control of high speed networks. *IEEE/ACM Trans. Networking*, Vol. 1, No. 3:pp. 329–343, June 1993.
- [4] A. Hung and G. Kesidis. Bandwidth scheduling for wide-area ATM networks using virtual finishing times. *IEEE/ACM Trans. Networking*, Vol. 4, No. 1:pp. 49–54, Feb. 1996.
- [5] A. Hung and G. Kesidis. End-to-end delay bounds and buffer sizing in ATM networks. Technical Report 95-08, E&CE Dept, Univ. of Waterloo, June 1995.
- [6] G. Kesidis. Specification of a fluid ATM network simulator. Technical Report (in preparation), 1995.
- [7] G. Kesidis and A. Singh. An overview of cell-level ATM network simulation. In *High Performance Computing Systems Conf., Montreal, Canada*, pages 202–214, July 1995.
- [8] R. Nagarajan. *Quality-of-Service issues in high-speed networks*. PhD thesis, CS Dept, University of Massachusetts at Amherst, 1993.
- [9] A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single node case. *IEEE/ACM Trans. Networking*, Vol. 1, No. 3:pages 344–357, June 1993.
- [10] L. Zhang. VirtualClock: A new traffic control algorithm for packet-switched networks. *ACM. Trans. Comp. Sys.*, Vol. 9, No. 2:pp. 101–124, May 1991.