TECHNICAL RESEARCH REPORT

ATM Network Simulation

by V. Peris

CSHCN TR 92-1/ISR TR 92-127

CENTER FOR SATELLITE &
HYBRID COMMUNICATION NETWORKS

A NASA CENTER FOR THE
COMMERCIAL DEVELOPMENT OF SPACE

University of Maryland Institute for Systems Research

ATM Network Simulation

Vinod Peris*
Institute for Systems Research, and
Center for Satellite & Hybrid Communication Networks
University of Maryland
College Park, MD 20742
peris@src.umd.edu

Abstract

ATM is the switching and multiplexing technique chosen by CCITT for broadband ISDN. Since no ATM networks are fully operational yet, very little is known about the traffic patterns that may prevail in the networks of the future. Thus, statistical models of traffic are difficult to justify, thereby complicating the theoretical analysis of such networks. Furthermore, the role of satellite links in ATM networks has also to be examined. An ATM network simulation would make it possible to study some of the issues confronting the design of such high speed networks. In this paper we describe our efforts at building an ATM network simulation package. Some of the issues regarding the development of this package, as well as the problems encountered are described herein.

1 Introduction

In the last few years there have been rapid developments in the area of high speed networking. Part of this is spurred on by the rapid progress made in optical fiber technology which allows data to be transmitted at the rate of gigabits per second, with very high reliability. Also the last couple of years has seen a lot of cooperation in the communications industry to arrive at uniform standards in order to make these networks interoperable. This has resulted in what are known today as ATM

*The work of this author was supported by NASA Grant #NAGW-2777S.

networks. ATM is the switching and multiplexing technique chosen by CCITT for broadband ISDN. The last couple of years have seen a lot of action in both the communications and the computer research communities on various aspects of the ATM technology. A large amount of research has been done regarding congestion control for these networks. However since very little is known about the traffic patterns that may prevail in the networks of the future, it has been rather difficult to justify the various kinds of assumptions that are needed for a theoretical analysis. There are a relatively large number of vendors offering ATM switches today, but very little is known about the performance of these switches in either local or wide area networking environments. While a handful of testbeds are taking shape in various parts of the country, very little work has been done in terms of simulating these networks. In this paper we describe some of our efforts at developing a simulation tool for an ATM network. In the next section we outline a brief introduction of the ATM technology. We refer the reader to [1] for a comprehensive tutorial on ATM networks. In section 3 we provide some motivation for the simulation of ATM networks. The following section examines some existing simulation infrastructure at the Institute for Systems Research. Section 5 describes the object oriented design of the ATM simulator. A rudimentary estimate of the simulation time is considered in section 6. Finally, we draw some important conclusions in section 7.

2 What is ATM ?

ATM is an abbreviation for *Asynchronous Transfer Mode*[1, 2]. Data (voice, video, etc.) is packaged into little packets each of size 53 bytes, called *cells*. Each cell has a 5 byte header followed by a 48 byte payload. These cells are transmitted from source to destination, in a connection oriented[1] manner. Along a path from source to destination the cells are routed by means of ATM *switches*. Before a call is made (during call setup) a *virtual connection* (VC) is established between the source and the destination. This sets up all the necessary protocol for switching the cells appropriately so that they reach their destination. The term virtual connection is used to emphasize the fact that the correct sequence of cells belonging to the same VC is maintained. Since the ATM cell header is rather small, only the VC identification, and not the entire source-destination address is carried in the header. However cells of many different VC's are multiplexed together onto the same transmission link.

[1]The cells arrive in proper sequence at the destination.

The key aspect of the ATM protocol is the limited amount of processing required at each node in the network. With link speeds in the order of gigabits per second, there is very little time to process these cells. Thus the ATM standards do not propose any end-to-end error control, but rely on the relatively error-free transmission capability of the fiber optic links. In order to maintain quality of service, it is important not only to have error free transmission capabilities, but also to make sure that queueing delays (or losses due to buffer overflows) are minimal. The CCITT standards recommend policing cell traffic at the User Network Interface (UNI), to ensure that users do not send more cells into the network than the amount negotiated during call setup.

3 Why Simulate ?

The ATM protocol is designed to be very simple to allow for the high transmission speeds, which are made possible by today's single-mode fiber optics. While there are a few vendors making ATM switches, there is very little known in terms of the performance of these switches or of the queueing delays and buffer occupancies at various nodes in the network. Since most of the technology is still in development and there are no large scale network services offered to date, simulation is an important way to analyze the performance of these high speed networks. With the rapidly changing technologies of today, a prototype testbed might become obsolete before it is even operational. Furthermore, as terrestrial networks evolve to high speed digital networks (Broadband ISDN), the role of satellite communications in the years to come needs to be addressed. We wished to examine the interworking of satellite links as part of ATM networks. With these ideas in mind we decided to build an ATM network simulator.

4 Existing Simulation Tools

There are a large number of existing discrete event simulation packages made by many different software companies. Since the problem at hand would require a large amount of computational resources we decided to use Sim^{++}[2], a distributed simulation tool [8]. This would allow us to run the simulation in parallel on many different workstations. The simulation would advance through virtual time through a mechanism known as Time Warp [5]. This basically made sure that the simulation would be oblivious to the fact that it was running on many different processors

[2]Sim^{++} is a trademark of Jade Simulations International Corp.

and the result would be the same if it were run on a single processor. There was also a Network Simulation Testbed being developed on top of Sim^{++}, by Scott Corson and some students of Dr. A. Ephremides. This was written in C^{++} and was completely object oriented. An important reason to go along with an object oriented development is the fact that it is easier to maintain and augment, than regular programming paradigms. With the rapid advances being made in the ATM technologies of today, coupled with the rapidly evolving ATM standards, it is important to be able to easily incorporate new objects into the simulation.

We contacted Dr. Brad Makrucki of BellSouth, who is a BellSouth representative at the CCITT ATM standards meetings. He had built a rudimentary ATM simulation tool, in FORTRAN, using a simulation package called SIMAN [6]. He kindly accepted to share his expertise with us and help in the design phase of the project.

5 Design of the ATM Simulator

In the summer of 1992, I made a trip to Atlanta and spent six weeks at BellSouth. The first week was spent in discussing some of the ATM features that it would be necessary to simulate in order to analyze the performance of the system as a whole. After that I looked at Dr. Brad Makrucki's simulator code which was to be the starting point for our simulation tool. Finally I developed an object oriented model for an ATM simulation, within the framework of the Network Simulation Testbed (Simnet) which was almost complete at that time [7]. There was already work underway to simulate satellite channels (SatSim) using Simnet. This would allow us to easily simulate satellite links, specifically the nature of fading channels, into the simulation.

Briefly, the design was as follows: The basic building block of the simulation was an OSI node. An OSI node consisted of 7 layers, and the ATM protocol would fit somewhere in layer 2. The fiber links would be modelled as point-to-point links with no errors. Since optical fibers are very reliable and we did not wish to investigate the effects of small losses in the fiber, the lossless assumptions were made. The satellite links would be derived from broadcast links, which would allow for noise corruption due to fade, and inclement weather conditions. Each call would be modelled as a call-setup phase followed by the actual call, and terminated by a call-teardown phase. The call-setup would involve the setting up of the VCI tables all along the established path. It would also involve the checking of required bandwidth to make sure that the call can be accepted. Furthermore, all nodes at the periphery of the network, would have the additional functionality to police

the incoming cell stream to make sure the negotiated bandwidth characteristics are maintained throughout the connection. This was to be performed by a leaky bucket, which is the method suggested by the CCITT [3].

6 Time considerations

Before actually setting out to write code for the simulator we decided to check out some of the computational resources that would be needed for the simulation.

1. A typical link in an ATM network operates at 155 Mb/s. The size of a cell being 53 bytes = 53×8 bits, a typical transmitter can send up to 365,566 cells a second.

2. Sim^{++} takes about 5 ms per event on a SPARCstation IPC.

3. At the very least we would need one event per cell transmission (typically there will be a lot of other processing that will go on as well, but we will ignore it for now). This would translate to 365,566 $\times 5$ ms $\approx \frac{1}{2}$ hour to simulate 1 sec of real time.

The above calculation is far lower than what we would typically need, because in addition to some processing for each cell transmission, we would have more than one output port (transmitter) simulated on one workstation, thereby slowing down the process even further.

The above news was rather disappointing, as it meant that Sim^{++} while giving us the advantage of added parallelism, was too slow to be of any use in any remotely realistic setting, i.e., any simulation involving more than a few seconds of real time. I contacted Jeff Allen from Jade Simulations International, to make sure that I had not missed something in coming up with these numbers and this was his response:

> I think that 5 ms/event is for remote events in parallel Sim^{++}. The figure for local events is closer to 1 ms, I believe. This may still be too long for your purposes - it has been for ours ...

We then started looking at other simulation packages to see if they were significantly faster. It turned out that most of them took on the order of a millisecond to process an event. Specifically, I contacted BONeS[3] and made them run an M/M/1 example on a SPARCstation 10 – the fastest of the Sun workstations at that time –

[3]BONeS is a registered trademark of Comdisco Systems, Inc.

and was told that it took around 100 secs, for a total of 100,000 events. This again works out to 1 ms per event for the simplest of queueing models. Opnet is another simulation tool developed by MIL 3. While this is slightly faster than BONeS, it is still not fast enough to simulate an entire ATM network. Furthermore, it is not object oriented (the user can link only C code into the simulation), thereby making it a little harder to incorporate evolving ATM standards and technologies.

7 Parallel Simulation

Since simulation on a single workstation or a collection of loosely coupled workstations (with Sim^{++}) was rather slow, we turned our attention to tightly coupled parallel machines. The University of Maryland Institute for Advanced Computing Studies (UMIACS) has recently acquired a connection machine CM-5 computer. The CM-5 computer offers scalable parallel computing. A CM-5 system contains tens or hundreds or thousands of parallel processing nodes. Nodes can execute the same instruction (SIMD style) or can execute independent (MIMD style) instructions. Each node consists of an industry standard RISC microprocessor (SPARC) with 8, 16 or 32 Mbytes of memory. Node design is centered around a standard 64-bit bus. Each node may optionally contain an arithmetic accelerator (vector unit) allowing 32 MFlops of 64-bit floating point performance, or 32 Mops peak 64-bit integer performance. The processing nodes are supervised by a control processor, which runs CMOST, an enhanced version of the UNIX operating system [4].

The connection machine communication library CMMD, provides traditional message-passing functions that facilitate the porting of MIMD-style code to the CM-5. It provides point-to-point message passing functions that require coordination only between the sender and the receiver nodes. Both blocking and non-blocking functions are provided. One way of implementing a simulation is to treat each node as an independent ATM switch in the network, and allow the CM-5 data network to actually carry all the ATM cells being transmitted. While this type of simulation would be very efficient in terms of speed, it may seriously differ from the simulation of a real ATM network. This is because an interrupt occurring at any node in the CM-5 would throw the whole system out of synchronization.

Another approach would be to use the concept of Time Warp [5] to make all the processors simulate ATM nodes in virtual time. This allows each of the nodes to send time-stamped messages to each other, and whenever there is a conflicting message – *viz* time stamp earlier than the current time – the system is rolled back to the earlier time. However this approach would significantly slow down the ATM simulation as nodes are typically sending a large stream (very many cells) of

messages to each other, and so a rollback of thousands of messages will be rather expensive. Thus while a parallel simulation affords greater computational power it also brings with it the burden of synchronizing the many parallel nodes.

8 Conclusions

So we asked ourselves "*Do we really need to simulate at the cell level ?*" The answer is an emphatic *Yes*. There are two major levels of operation in an ATM network. One of them is at the cell level, and the other is at the connection level. The cell level is the basic level of operation of the ATM switch. The source packages data into cells and transmits them to the UNI. Cells are queued up either at the input or the output ports of the switch, depending on the switch architecture. Cells are routed based on the VCI carried in their header, and cells are preferentially dropped based on certain cell loss priority bits in the header. Furthermore, congestion control is done on a cell by cell basis at the network interfaces, using a leaky bucket or some other equivalent mechanism. Thus a cell level simulation would be appropriate for many quantities of interest. At the connection level, connection parameters (like peak rate, average rate) etc., and decisions regarding the routing, are made. However the present proposal to use in-band signalling (like that in SS7), closely links the switch performance at the cell level with that of the connection level operations. The result of all this is that a cell level simulation is necessary to study network performance. However with the present computing technologies, a simulation could take a few days for a few seconds of simulated traffic.

Acknowledgments

We would like to thank Dr. A. Makowski for many of the fruitful discussions that led to this work. His constant support and guidance is gratefully acknowledged. We also thank Dr. Brad Makrucki, who very graciously spent a lot of time, explaining the various ATM standards' issues as well as his simulation work.

References

[1] J.-Y. Le Boudec. "The Asynchronous Transfer Mode: a tutorial". *Computer Networks and ISDN Systems* **24** (1992), 279–309.

[2] CCITT Revised Recommendation I.361 "B-ISDN ATM layer specification", Geneva, June 1992.

[3] CCITT Recommendation I.371 "Traffic control and congestion control in B-ISDN", Geneva, June 1992.

[4] *Connection Machine CM-5, Technical Summary*, Thinking Machines Corp., Cambridge MA, Nov 1992.

[5] D. R. Jefferson. "Virtual Time". *ACM Transactions on Programming Languages and Systems* **7**,3 (July 1985), 404–425.

[6] C. D. Pegden, R. E. Shannon and R. P. Sadowski. *Introduction to simulation using SIMAN* McGraw-Hill, New York, 1990.

[7] *Simnet: A distributed network simulation testbed. Programmer Reference Manual*, Release 1.0, July 1992.

[8] *Sim^{++}, A discrete-event simulation language*, Release 3.3, Jade Simulations International Corp., 1991.