INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Monitoring and Modelling Multimedia Traffic over TCP/IP/ATM Networks

by

Weimin Huang, B. Eng.

Southeast University, P.R.China

A thesis submitted to the
Faculty of Graduate Studies and Research

in partial fulfillment of
the requirements for the degree of

Master of Engineering

Ottawa-Carleton Institute of Electrical Engineering

Faculty of Engineering

Department of Systems and Computer Engineering

Carleton University

June 1, 1997

Ottawa, Ontario

© copyright

1997, Weimin Huang

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-26996-5

The undersigned recommend to the Faculty of Graduate Studies and Research
acceptance of the thesis

**Monitoring and Modelling of Multimedia Traffic over
TCP/IP/ATM Network**

submitted by Weimin Huang, B. Eng.

in partial fulfillment of the requirements for

the degree of Master of Engineering

Thesis Co-Supervisor

Thesis Co-Supervisor

Chairman, Department of Systems and Computer Engineering

Carleton University

Abstract

The high speed nature of ATM networks makes it difficult to capture and analyze traffic at the ATM layer, especially for real time multimedia applications. In this thesis, measurements are taken on both the application layer and the ATM layer while running Multimedia-News-on-Demand application over OCRInet (an ATM network). The measurements demonstrate a sequence of problems, which include the large cell delay variation introduced by the network, deadlock occurring at the TCP layer when the user message size is not selected carefully, the traffic burstiness caused by the CPU sharing and the application layer protocols. The tight relationship among the traffic at the different protocol layers is also shown by the measurements.

The studies in this thesis also include an OPNET model simulating the implementation of running TCP/IP over ATM in IBM AIX 3.2.5 operating system. The modelling result matches the one obtained from the measurements statistically, and it also verifies the deadlock problem shown by the measurements.

Acknowledgments

“Monitoring and Modelling of Multimedia Traffic over ATM network”, is a joint project of MCRlab (Multimedia Communications Research Lab) at the University of Ottawa and BBNlab (Broadband Networks Lab) at Carleton University. I would like to express sincere appreciation to all of the people who worked collaboratively to establish an ATM network loop connection between the University of Ottawa and Carleton University, in particular, Jim Yuan (OCRInet), Keith Smith (University of Ottawa) and Dave Sword (Carleton University) deserve special recognition. I am extremely grateful to the Multimedia Communications Research lab of the University of Ottawa, who provided me the measurement source, the “News-on-Demand” multimedia stream of the MCRIlab CITR project of Prof. Georganas “Synchronization of Multimedia Data for presentational Applications”. Special thank is due to Jeff Brinskle, the manager of MCRIlab, who has been very supportive of the project. I particularly want to thank GN Navtel for providing the Nettek InterWatch 95000 for the measurement over ATM layer, so I have the chance to observe the traffic at cell level.

I would like to thank Prof. Orozco-Barbosa, Prof. Georganas, Prof. Devetsikiotis, Prof. Lambadaris and Prof. Kaye for their patient direction and warm support for this project. Their guidance and encouragement is always important during the course of the thesis.

I wish to express appreciation to my husband, Shiyuan Qi for his support and encouragement during this research. Special thanks are due to my family in China for their constant care.

I gratefully acknowledge the funding provided by the Telecommunications Research Institute of Ontario.

Table of Contents

Acceptance Sheet	ii
Abstract	iii
Acknowledgments	iv
Table of Contents	v
List of Tables	ix
List of Figures.	xi
List of Acronyms	xv
 1 Introduction	
1.1 Background	1
1.2 Related Research	5
1.2.1 Measurement at ATM Layer	5
1.2.2 Performance of Running TCP/IP over ATM	8
1.3 Motivation and Contributions of This Thesis.	9
1.4 Thesis Organization.	12
 2 Application Software and Network Protocols	
2.1 Multimedia News-on-Demand Application.	13
2.1.1 Introduction	13
2.1.2 Scenarios and Material Segmentation.	13
2.1.3 Multimedia Communication Protocols at Application Layer.	14
2.1.4 Network Protocol Layers	17

2.2	TCP/IP	18
2.2.1	TCP Operation Principle	18
2.2.2	Internet Protocol	20
2.3	ATM and its Flow Control Mechanisms	20
2.4	Performance of Running TCP/IP over ATM Networks.	22

3 Measurement Environment

3.1	OCRInet Connections.	25
3.1.1	Introduction of OCRInet	25
3.1.2	OCRInet Connection between Two Universities	26
3.2	Running TCP/IP over ATM Networks in IBM AIX 3.2.5 Operating Systems Environment	28
3.2.1	Overview	28
3.2.2	Socket Layer	29
3.2.3	TCP Layer.	34
3.3	ATM Interface Layer	36
3.3.1	Introduction	36
3.3.2	Support IP/ATM.	39
3.3.3	Traffic Control Function	39
3.4	GN Navtel Interwatch 95000	39
3.4.1	Introduction	39
3.4.2	Measurement Connections	40
3.4.3	Measurement Functions.	40
3.4.4	Drawbacks	42

4 Multimedia Traffic Measurement and Analysis

4.1	Measurement Points.	43
4.2	Overview of Measurement Experiments	44
4.3	Data Processing and Information Extraction	47
4.4	Measurement Results and Data Analysis	47
4.4.1	Delay Variation Introduced by Network	48
4.4.2	Low Traffic Rate with SOCK_PAC_SIZE of 1024 bytes	52
4.4.2.1	The Causes of Low Throughput	52
4.4.2.2	Verify the Problem from Cell Interarrival Time of M2610, M2611 and M2612	55
4.4.2.3	Verify the Problem with Cell Rate of M2610, M2611 and M2612	56
4.4.2.4	Demonstration of the Problem from ACK Traffic Of M2710, M2711 and M2712	58
4.4.2.5	Comparison with other Tests	60
4.4.2.6	Relationship to Previous Study and Lessons Learned	63
4.4.3	Traffic Burstiness Contributed by Application Layer.	65
4.4.4	Relationship between Traffic at Protocol Layers	77
4.4.5	Positive Autocorrelation in Traces of M2610, M2611 and M2612.	78
4.4.6	Other Results	83
4.5	Summary	84
 5 Modeling		
5.1	Purpose	88
5.2	Modules	88
5.2.1	Application Layer.	89
5.2.2	Socket and TCP Layer	91

5.2.3	ATM Layer	94
5.2.4	Fidelity Level	96
5.3	Comparison of Modeling Result and Measurement	96
5.4	Further Information Provided by Simulation Modeling	96
6	Conclusions and Recommendations for Further Study	
6.1	Conclusions	100
6.2	Recommendations for Further Study	102
	Appendix A: The Scenario of Multimedia News	104
	Appendix B: The Software Structure of Multimedia News-on-Demand Application	107
	Appendix C: Rate-based and Credit-based ATM Flow Control Schemes	113
	Appendix D: Function of IP over ATM Supported in Fore Systems Card	116
	References	118

List of Tables

Table 4.1:	General Information about Measurements on MM Traffic	45
Table 4.2:	General Information about Measurements ACK Traffic	45
Table 4.3:	General Information about Measurements at the Application Layer .	47
Table 4.4:	Percentage of Cells at a rate over PCR set on Fore Card	48
Table 4.5:	Distribution of Cell Interarrival Times of M2610	50
Table 4.6:	Distribution of Cell Interarrival Times of M2611	50
Table 4.7:	Distribution of Cell Interarrival Times of M2612	50
Table 4.8:	An Example of Stop-and-wait Manner.	54
Table 4.9:	Distribution of No of Cells in 1ms of M2610, M2611 and M2612 . . .	57
Table 4.10:	Distribution of ACK Cell Interarrival Times of M2710	59
Table 4.11:	Distribution of ACK Cell Interarrival Times of M2711	59
Table 4.12:	Distribution of ACK Cell Interarrival Times of M2712	59
Table 4.13:	An Example of Interaction between the Server and the Client when SOCK_PAC_SIZE=4096	52
Table 4.14:	Operating Order of Three Servers in Test 3	67
Table 4.15:	Distribution of Audio Message Interdeparture Times of sa11	71
Table 4.16:	Distribution of Audio Message Interdeparture Times of sa12	71
Table 4.17:	The Distribution of Video Message Interdeparture Times of sv11 . . .	73
Table 4.18:	The Distribution of Video Message Interdeparture Times of sv12 . . .	73
Table 4.19:	The Activities at the Application Layer of M2611.	80
Table 5.1:	Level of Model Fidelity	96
Table A.1:	Audio Segments	104

Table A.2:	Video Segments	^x 105
Table A.3:	Text Segments.	106

List of Figures

Figure 2.1	Example of the Structure of a Temporal Scenario	14
Figure 2.2	The Communication Sequences between Client and Server	16
Figure 2.3	Protocol Layers involved in the Application	17
Figure 2.4	Basic TCP Traffic Control Procedures	19
Figure 2.5	The Principle of Leaky Bucket	21
Figure 3.1	OCRInet Topology	25
Figure 3.2	OCRInet Physical Connection Between OU and CU.	26
Figure 3.3	The Assignment of PVCs of the Connections	27
Figure 3.4	Flow Control involved in Multimedia Application over ATM	28
Figure 3.5	The Incremental Data Operation at Each Layer	29
Figure 3.6	Data Distribution in Clustered MBUFs	31
Figure 3.7	Transmit Socket Layer Data Copy Strategy in IBM AIX 3.2.5.	32
Figure 3.8	TCP Window Update Algorithm	34
Figure 3.9	TCP Segmentation and Nagle's Algorithm	35
Figure 3.10	The Protocol Configuration with ATM and Ethernet Interface	37
Figure 3.11	Fore System Card Advanced Cell processor Architecture	38
Figure 3.12	LB Implemented on ATM Fore Card	39
Figure 3.13	The connection of GN Navtel and Newbridge Switch at Carleton U.	40
Figure 3.14	Cell Capturing Function of GN Navtel	41
Figure 3.15	An Example of "Decode Display"	41
Figure 4.1	Measurement Points.	43

Figure 4.2	Store-and-forward Mechanism Indicated in M2610 traffic	49
Figure 4.3	A Typical Segment of Cell Interarrival Time of M2611	49
Figure 4.4	An Example of Stop-and-wait Manner	55
Figure 4.5	Influence of Nagle's Algorithm and Delay-ACK on MM Traffic when SOCK_PAC_SIZE = 1024 bytes	56
Figure 4.6	A Segment of Sample Path of Number of Coming Cells in 1 ms of M2610	56
Figure 4.7	A Segment of ACK Cell Interarrival Time of Trace of M2710.	58
Figure 4.8(a)	Number of Coming Cells in 1 ms of M2641 in the first 300ms	61
Figure 4.8(b)	Number of Coming Cells in 1 ms of M2610 in the first 300ms	61
Figure 4.9	The ACK Cell Interarrival Time of M2741	63
Figure 4.10	Audio Traffic Defined in the Scenario	65
Figure 4.11	Video Traffic Defined in the Scenario.	66
Figure 4.12	Text Traffic Defined in the Scenario	66
Figure 4.13	CPU Sharing and Servers' Status	68
Figure 4.14(a)	A Segment of Audio User Message Interdeparture Times Trace of sa11	70
Figure 4.14(b)	A Segment of Audio User Message Interdeparture Times Trace of sa11	70
Figure 4.14(c)	The Reflection of Nagle's Algorithm and Delay-ACK on sa11	71
Figure 4.15	The Video Message Interdeparture Time of Trace sv11	73
Figure 4.16	The Audio Message Interdeparture Time of Trace sa41	74
Figure 4.17	The Audio Message Interdeparture Time of Trace sa81	74
Figure 4.18	The Audio Message Interdeparture Time of Trace sa91	75
Figure 4.19	The Video Message Interdeparture Time of Trace sv41	75
Figure 4.20	The Video Message Interdeparture Time of Trace sv91	76

Figure 4.21	The Video Message Interdeparture Time of Trace sv81	76
Figure 4.22	The Video Message Interdeparture Time of Trace sv82	77
Figure 4.23	Data Movement Path and Feedback in Traffic Pattern	77
Figure 4.24	Autocorrelation of Multimedia Cell Interarrival time of M2611: Positive Peak at Lag 195	78
Figure 4.25	Number of Cells between Two Cell Interarrival Time Peaks (M2610, M2611 and M2612	79
Figure 4.26	The Text Message Interdeparture Time	89
Figure 5.1	Modules Included in Simulation Models	90
Figure 5.2	State Transmission Diagram of the Audio Server Model.	90
Figure 5.3	State Transmission Diagram of the Audio Player Model.	90
Figure 5.4	The State Transmission Diagram of the Model of Socket/TCP Layer at Server Site	92
Figure 5.5	The State Transmission Diagram of the Model of Socket/TCP Layer at Client Site.	94
Figure 5.6	The State Transmission Diagram of the ATM Layer Model	95
Figure 5.7	Comparison of Simulation Result and Measurement	97
Figure 5.8	Comparison of Simulation Result and Measurement on Histogram . .	98
Figure 5.9	Comparison of Simulation Result and Measurement on Autocorrelation .	98
Figure 5.10	Cell Interarrival Time with Respect to Simulation Time when SOCK_PAC_SIZE = 1024 bytes	99
Figure 5.11	Cell Interarrival Time with Respect to Simulation Time when SOCK_PAC_SIZE = 935 bytes	99
Figure B.1	The Architecture and Data Flow in the Multimedia News-on-Demand Application.. . . .	108
Figure B.2	The Introduction of Synchronization Delay	109
Figure B.3	The Schedule is Shared by Three CMSCs	110

Figure B.4	Structure of the Shared Buffer Used by Socket_MSC & FIFO_MSC .	112
Figure C.1	Rate-based Congestion Control	113
Figure C.2	Credit-based Congestion Control Applied to Each Link of a VC . .	114

List of Acronyms

AAL	ATM Adaptation Layer
ABR	Available Bit Rate
API	Application Programming Interface
ARP	Address Resolution Protocol
ATM	Asynchronous Transfer Mode
BECN	Backward Explicit Congestion Notification
CBR	Constant Bit Rate
CCS	Common Channel Signaling
CMSC	Client Media Synchronization Controller
C-Plane	Control Plane
CPE	Customer Premises Equipment
CPU	Central Processing Unit
EPD	Early Packet Discarding
FECN	Forward Explicit Congestion Notification
FIFO	First In First Out
FR	Frame Relay
FTP	File Transfer Protocol
GUI	Graphical User Interface
IP	Internet Protocol
IPC	Inter-Process Communications
ISDN	Integrated Services Digital Network
LAN	Local Area Network

LB	Leaky Bucket
MPEG	Moving Picture Experts Group
MMDB	Multimedia Database
MSC	Media Synchronization Controller
MTU	Maximum Transfer Unit
OC	Optical Carrier
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
PPD	Partial Packet Discarding
PVC	Permanent Virtual Circuit
QoS	Quality of Service
RM	Resource Management
SCR	Sustainable Cell Rate
SMDS	Switched Multimegabit Data Service
SMSC	Server Media Synchronization Controller
SNMP	Simple Network Management Protocol
SSP	Stream Synchronization Protocol
SVC	Switched Virtual Circuit
TCP	Transmission Control Protocol
UBR	Unspecified Bit Rate
UDP	User Datagram Protocol
UNI	User Network Interface
VBR	Variable Bit Rate
VC	Virtual Circuit
VP	Virtual Path
WAN	Wide Area Network

Chapter 1

Introduction

1.1 Background

Considerable research and development efforts are underway in the design and deployment of high-speed networks. ATM (Asynchronous Transfer Mode) network is one popular example. It is a connection-oriented network with a fixed cell size of 53 bytes, which composes 48 bytes of payload and 5 bytes of header. Cell Routing is accomplished through packet switches over VCs (virtual circuits) set up between endpoints. By using of proper scheduling algorithms in the packet switches, ATM is able to handle multiple classes of traffic. In the current ATM Forum UNI 4.0 draft proposal, 5 classes of service - UBR (Unspecified Bit Rate), ABR (Available Bit Rate), CBR (Constant Bit Rate), Real-Time Variable Bit Rate and Non-real-time Variable Bit Rate have been defined to be supported by ATM networks[1]. For each of these traffic classes, separate traffic management mechanisms have to be designed to deal with different types of QoS (quality-of-service) requirements.

One type of target applications for ATM networks is multimedia communications. In these applications, multimedia traffic --- the mixture of media streams with various network QoS requirements, is much more complex than pure data/voice traffic in conventional networks. Furthermore, in order to achieve real-time data delivery and keep synchronization among the media streams, the multimedia applications usually have their specific communication protocols at application layer. Such protocols are independent of network ones. Because of the diversity of goals, these protocols may be different from application to application, and their effects over the multimedia traffic are distinguishable each other. Therefore the analysis of multimedia traffic is a complicated work that includes not only the research on the media sources but also the influence from application and network protocols.

TCP/IP (Transmission Control Protocol / Internet Protocol) is a collection of network protocols widely used in data communication networks. TCP is a reliable, connection-oriented, end-to-end transport layer protocol that provides flow-controlled data transfer, while IP is a network layer protocol supplying connectionless datagram services. Because it allows user-to-user communication for hosts connected to many different kinds of subnetworks, TCP/IP is also called the internet protocol suite.

A great deal of attention has been paid to the performance of TCP/IP over ATM. ATM is undoubtedly a promising technology for broadband networks in the future. However, in order to protect the large investment spent on the Internet Protocol Suite, ATM is expected to be able to support TCP/IP as its upper layer protocols. Moreover, in ATM networks traffic with no real-time deadlines is intended to be transported in a “best-effort” mode, which provides no guarantees on the available bandwidth, delay or cell loss rate. This is similar to the service provided by the current Internet, except that connections still need to be set up at the ATM layer over virtual circuits between endpoints. Because of this similarity, TCP is likely to be used as the transport protocol to support these “best effort” services over ATM. Therefore, it is important to study the operation of TCP traffic control mechanisms on ATM networks, especially when congestion control schemes are introduced at the ATM layer.

The huge marketing potential of ATM greatly promotes the development of ATM products in the industry. While ATM traffic management mechanisms are still in the process of being standardized, network equipment manufactures have actually developed ATM equipment providing gigabit channels. Although there has been a debate in the research community regarding the suitability of TCP/IP for applications over ATM networks, supporting TCP/IP has become a basic function of many ATM products. Currently, a user who connects to an ATM network can run existing applications using TCP or UDP (User Datagram Protocol) even without modification.

Advances in ATM networking and distributed applications suggest that high data rates will be available to support applications with various QoS requirement. However, appropriate operating

systems support and efficient protocol implementation are also essential to the high-bandwidth applications. These applications require not only high-speed networks but also carefully engineered end-to-end protocols, which can be implemented efficiently within the constraints of various operating systems and host architectures. In a real life network, the protocols have to be operated in a complex environment comprised of various operating systems, host architectures, and rapidly growing and evolving Internet connection of several heterogeneous subnetworks. Thus, evaluation of the protocols is quite a challenging task that can not be achieved by studying the protocols in isolation.

Traffic monitoring provides an appropriate solution on traffic observation and protocols evaluation. In fact, proper monitoring of network traffic levels and usage are always basic to the operation and traffic engineering of all packet networks, including X.25, ISDN (Integrated Service Digital Network), CCS (Common Channel Signaling), FR (Frame Relay), SMDS (Switched Multimegabit Data Service) and ATM networks[2]. If it is done at multiple protocol layers simultaneously, the measurements will provide an overall image of the traffic behavior and protocol operations. The traffic analysis at one layer can be done with the information collected at the other layers. so it can reveal the relationship between the traffic at different protocol layers. There are some other purposes of traffic measurements summarized as follows:

- Traffic Measurement is an essential input to performance models and traffic engineering algorithms;

In network engineering, performance of a network and its traffic control functions are tightly related to the characteristic of the traffic running over it. Different conclusions may be drawn when different traffic inputs are tried upon a given network model. Thus, valuable simulation results can not be guaranteed if the input does not describe real traffic closely enough. Traffic measurements in high resolution supply reliable inputs to network models and traffic engineering algorithms and prevent the performance evaluation from “Garbage in and Garbage out”.

- Traffic measurement can validate simulation models.

Validation of simulation models with traffic measurement is needed in order to get reliable simulation, especially when mathematical models of these networks are not available [4]. With real life traffic as input to the model, the variance between simulation result and measurements could reflect how well a model describes the real system and whether the model is acceptable.

- Traffic Measurement can depict operation of protocols.

Some measurement mechanisms can describe the operation of protocols by collecting packets flowing between hosts. For instance, TCP_DUMP offers detailed information of each TCP packet for one communication session. Thus, the peer-to-peer dialogue at the Transmission layer can be shown clearly.[5]

- Measurements of traffic levels are necessary for triggering congestion controls and operator alarms;

This kind of measurements is implemented in performance monitoring in the field of Network Management. For example, on one interface when the number of dropped packets is over a threshold, an alarm will be sent to a manager workstation, and an error will be reported to a log file.[6]

- Measurement of traffic can detect throughput bottlenecks in a network;

In network administration and maintenance, traffic measurement is often used to uncover the bottlenecks in a network. This sort of diagnosis helps users analyze the cause of the bottlenecks and keep the network in order[7]. Most of protocol analyzers are designed for this purpose.

Traffic Measurement can be made over many time intervals - from milliseconds to hundreds of seconds[3]. The choice of time intervals depends on the purpose of the measurement. For instance, traffic measurements that serve as triggers for congestion controls need to be done on finer time intervals than measurements in support of traffic engineering. For the research on network traffic, some phenomenon (such as burstiness of the traffic) might be missed if the measurement time interval is not fine enough.

Measurements can usually be done on either the workstation running the examined application or another machine (monitor) that is independent of the application. When both the application and the measurement programs run over the same machine, the measurement overhead should be considered, for, too many CPU cycles dedicated to the measurement would not only influence the application under study but also introduce errors in the measurement. Thus, the interval of such a measurement can not be very fine, and high measurement resolution can not be expected. An isolated traffic monitor seems ideal to handle the measurement. However, to provide precise and reliable information, the captured traffic traces need to be complete and long. Therefore, an ideal monitor for a high-speed network should at least satisfy two requirements: high operating speed and a large memory. The high operating speed is required in order to capture each packet, for, incomplete traces convey uncertainty to researches. Furthermore, since certain phenomena in a data network can only be observed within a long time span, a large memory is needed to keep the long trace. Finally, the higher speed of a network, the greater the difficulty is on traffic monitoring. To keep the trace of high speed traffic for a reasonably long duration is very challenging to commercial monitors.

1.2 Related Research

1.2.1 Measurement at ATM Layer

Measurements always accompany the development of data networks. They have greatly contributed to the research in this field [8-12]. In recent years, significant progress in the area of traffic measurements starts to occur. Traffic levels used to be monitored by measuring either packet counts or switch element occupancies. These measurements were typically over intervals of 15 to 30 minutes, even for high-speed data networks. The measurements at Bellcore[3] were done at packet level with the interval of one second, and they demonstrated high burstiness of packet traffic which had never been shown in the traditional traffic measurement. Therefore, a finer-time-interval monitoring was suggested to indicate fractal nature of packet traffic and resources usage[2]. Some other measurements tried to analyze network performance with some

specific applications. For instance, the performance of SUNOS IPC (inter-process communications) over TCP/IP/Ethernet was studied by taking measurements at queues of different protocol layers[13]. The measurement obtained the information about each queue at various protocol layers. As a result, a bottleneck of throughput was detected, and schemes to improve performance were also proposed. By capturing timestamp of each packet and monitoring network throughput, the behavior of TCP protocol in both lightly loaded networks and busy segments of the Internet were analyzed respectively in [5][14]. The studies discovered the throughput problems relating to both operating systems as well as the real-life traffic.

The measurements mentioned above used various methods, among which the timestamp for each packet provided the most primitive and complete information. To enhance the measurement function at TCP layer, a timestamp option for TCP was even proposed. However, the implementation of timestamp is very tough for cell-oriented measurements in ATM networks. First of all, there is no room in the ATM cell header to place a timestamp to perform the round-trip delay measurement. Secondly, software-based per-cell processing at a high cell rate such as OC-3 is doubtedly practical. Finally, to some media streams like video, certain traffic characteristics can not be observed unless the captured trace is long enough. Large memory is required for this kind of measurement. Due to lack of qualified ATM traffic monitors, research on ATM traffic had to be done only through measurements at the higher protocol layers (such as TCP layer). When this project started, there was no literature available on measurements at the ATM layer. Methodologies for implementing a traffic monitor for ATM networks is still a challenging task for protocol analyzer designers.

However, ATM networking is a dynamic field, and a few breakthroughs on measurements happened in 1996[7][15]. A method that allowed accurate measurement of the delay incurred by individual cells in real traffic streams has been developed in [15]. This scheme was implemented on a traffic analyzer called Attila[16] which was developed at MCNC under an ARPA project. Instead of using timestamps, this scheme used the cell interarrival time measured by two Attilas, each at the source and destination to compute cell jitter. Cell delay was obtained through a simple

manipulation of cell jitter and the baseline delay, which was the round trip delay of the first captured cell in the stream. A HP 75000 tester (series 90) was used to generate a traffic stream and measure baseline delay of the whole stream. So the measurements were done with three machines together, one HP 75000 tester and two Attila traffic analyzers. The traffic analyzer was reported to work at a speed as high as OC-3 rate, while the length of traffic trace was not available in the paper. One major result from the measurements was a surprising delay variation introduced by the networks. More delay variation was found as the traffic flow passed through more network nodes, even when the network is extremely lightly-loaded. Unfortunately, the testing traffic flow from HP machine were cell streams at constant rates, which were too simple to represent the real-life traffic. The measurements only obtained essential performance of the networks without considering effects from both traffic sources and upper layer protocols. Therefore, this study could not evaluate the real life network performance.

An approach to analyze the performance of distributed applications in a high-speed WAN is described in [7]. It was designed to identify all of issues that impact performance and isolate the causes due to the related hardware and software components. A distributed parallel data server was used as a network load generator. Several monitoring spots were set intentionally to probe various aspects of high-speed distributed systems from top to bottom of the protocol stack and from end to end in the network. Thus the information provided by the approach is very comprehensive. The traffic behavior at application layer is recorded by the traffic generator when messages enter and leave all parts of the user-level system, and timestamps are taken and logged. Other monitoring about Operating Systems and network performance, such as TCP retransmits, Central Processing Unit (CPU) usage, CPU interrupts, ATM adaptation layer(AAL) information, ATM switch buffer overflows, and ATM host's adapter buffer overflow, was implemented by the modified existing tools ---- netstat, vmstat and Simple Network Management Protocol (SNMP) queries. Netstat displays the content of various network-related data structures, while vmstat reports statistics of other things, such as virtual memory, disk, and CPU activity. SNMP queries gave information about several aspects of the network devices, such as cell loss and buffer overflow in ATM switches. The measurements verified the effect of TCP window size, MTU

(Maximum Transfer Unit), and ATM congestion control over the performance of the network. However, the measurements were done over a time interval of 100ms, which might be fine enough to describe the status of network performance on a general basis, but definitely not good enough to provide detail information about traffic traces at the cell level.

1.2.2 Performance of Running TCP/IP over ATM

Even without accurate measurement at cell level, many studies have been devoted on the topic of running TCP/IP over ATM. Simulations and measurements at TCP layer are two major research efforts in this area. A major conflict between TCP and ATM was revealed from the simulations. As TCP works on the basis of positive acknowledgments, and any incompletely received packets would be regarded as being corrupted, one single dropped ATM cell would result in the retransmission of the whole TCP packet. Due to the small size of an ATM cell, the usage of bandwidth could be degraded seriously when congestion happens. Some simulation results[17][18] showed that the support to TCP/IP from a plain ATM (without any flow control) is quite poor, especially when the buffer sizes in the switches are small.

Some researches focus on how to enhance the flow control at the ATM layer and to support TCP/IP more properly. Two basic ATM flow control schemes are proposed to support ABR service: rate-based and credit-based ones. Simulation results[18][19] demonstrated that the credit-based flow control could support TCP perfectly, although it might need an incredibly large buffer space on switches and endpoints. On the other hand, intelligent cell discarding policies are designed to improve the network performance[17]. Partial Packet Discarding (PPD) and Early Packet Discarding (EPD) are two of them. In PPD, once a cell is dropped on a switch, the left cells belonging to the same IP packet will also be dropped. Whereas in EPD once the occupied space in buffer is over one specified threshold, the whole IP packet will be dropped to prevent the buffer from overflowing. EPD is shown to be more efficient than PPD and able to support TCP fairly well[18]. Another simulation[2] investigates issues on VBR support over TCP/IP/ATM networks. The results showed that interactions between TCP congestion control algorithms and leaky bucket policing can cause the average TCP throughput to be much lower than the SCR (Sustainable Cell

Rate). This is because a LB-policed TCP connection sees frequent packet losses, as the part of a cyclical loss pattern with multiple TCP packet losses at the end of every cycle. The study improved the performance by using a modified early-drop-LB policing mechanism. All these simulations not only manifested the potential problems on running TCP/IP over ATM networks but also proposed many ideas on performance improvement. However, almost every simulation model was established on the basis of operation principle of TCP described in [20][21], without considering a practical implementation in any specific operating system environment. Furthermore, real-life traffic was never used in these simulations. Therefore, it is hard for these models to describe very accurately the real network performance when running a real application. The proposed improvement might not be so evident in a true environment as proved in the simulations.

Some other performance problems relating to TCP/IP/ATM were found by measuring throughput at TCP layer [22][23]. The two studies obtained similar conclusions that socket buffer sizes and the ATM interface MTU have a dramatic impact on the throughput. Thus, performance tuning-up is necessary to gain a high throughput. Performance measurement at TCP layer also verified simulation models of ATM networks at an appropriate level of fidelity[4]. These measurements revealed the performance issues relating to TCP/IP/ATM in a real network, however, the sources used in the experiments were traffic generated by measurement tools such as `ttcp` instead of a real application. While those measurement tools are useful ones to start with, people have observed [7] many cases where `ttcp` performance is reasonable, but performance of real applications is still poor. Real distributed applications are complex, bursty, and have more than one connection in and/or out of a given host at one time; tools like `ttcp` can not adequately simulate these conditions.

1.3 Motivation and Contributions of This Thesis

While many research results in the field of TCP/IP/ATM have been listed in Section 1.2, there are still lots of work to be done. This invoked the research for this thesis.

In most cases, measurement results were seldom obtained from a real application running over a real network. Due to the tight relationship between source characteristics and network performance, protocol stacks can not guarantee the ability to handle real traffic unless they are tested with complex and bursty traffic from real sources. Though some performance problems have been uncovered in measurements with simple artificial traffic before, these measurement do not provide realistic evidence of network performance.

Although a layered protocol structure of Data Communication Systems has been used for a long time, surprisingly, there has been little published in the technical literature concerning the interaction and relationship between traffic at different protocol layers. Consequently, research has not focused to the combined effects of multiple protocols over the traffic at each layer. Although there are various causes of burstiness of the traffic, people seem to be more interested in the burstiness introduced by traffic sources than by protocols. This might be a result that few measurements have been done over multiple protocol layers simultaneously. Even in some experiments performance measurements have been done at each protocol layer[7], the large measurement intervals prevented the results from providing detailed traffic traces.

Furthermore, very limited real traffic statistics are available on ATM networks, due to the difficulty of capturing a traffic trace at high speed. While a tremendous amount of effort has been devoted to the analysis of ATM traffic characteristics, most of the existing research relies on theoretical models that may not accurately describe the operation of real-world networks. In addition, there is concern that present transport and network layer protocols (TCP/IP) will not support an adequate throughput when used to drive high-speed networks. Proper tests and measurements on traffic at ATM layer are necessary in order to address these issues regarding development on ATM and its upper layer protocols.

The problems of running TCP/IP over ATM networks have been studied intensively from measurements of throughput at TCP layer in SunOS 4.1[22][23], and some problems have been found about the influence of the operating system over the traffic. But few experiments are done in other operating systems. It is interesting to see whether the discovered problems in [22][23] are

specific to SunOS or not, and whether some general conclusions can be gained regarding the influence of operating systems on network performance.

Finally, few simulation models of running TCP/IP over ATM networks described practical implementations of a real operating system and network. What they considered was the fundamental functions of TCP (slow-start sliding window, time-out and retransmission). Other factors relating to the operating systems, which also have great effect on traffic pattern, such as socket buffer size, Nagle's algorithm and delay-ACK policy, are often ignored. It is, therefore, desirable to establish a simulation model that is closer to real life system and provides more reliable results.

This thesis presents a study that is based on measurement at both the application layer and the ATM layer when running a multimedia application over a TCP/IP/ATM network. An OPNET model was built to depict the application and network systems, and it is verified by the measurement results. The measurements and analysis exposes:

- Traffic Behavior at the application layer and the ATM layer;
- Contribution from each protocol layer to traffic shapes;
- Relationship between the traffic at multiple protocol layers;
- Practical problems when running an application over TCP/IP/ATM network in IBM AIX 3.2.5 operating systems.

The modeling work addresses:

- A simulation model of running a data transferring application over a TCP/IP/ATM network in IBM AIX 3.2.5 operating systems;
- Model verification with measurement results;
- Traffic trace which may not be gained from measurements;

The major steps of research for this thesis are summarized as:

- i Study of the application and traffic transport control mechanisms;

- ii Measurement at the application layer and the ATM layer when running a multimedia application over a TCP/IP/ATM network;
- iii Analysis of the traffic based on measurement and observation;
- iv Development and verification of an OPNET model of running the multimedia application over OCRInet.

1.4 Thesis Organization

The remaining chapters of the thesis are organized as follows:

Chapter 2: Introduces the application software: “Multimedia News-on-Demand Application” and the traffic control functions at each of these Network Protocol Layers: Application layer, TCP layer and ATM layer.

Chapter 3: Presents information about measurement environment. The OCRInet testbed is described on both configurations and detailed connections. The implementation of TCP/IP/ATM in an IBM AIX 3.2.5 operating system with Fore ATM Interface Card MCA-200 is fully described. GN Nettek, the ATM protocol analyzer used in our experiment, is also described.

Chapter 4: The measurement scheme is presented briefly. This chapter is dedicated to what is observed from measurement results. A sequence of conclusions is provided with the corresponding evidence obtained from the measurements. A summary is offered at the end.

Chapter 5: Develops an OPNET model of the application and network connections used in our experiment. The functions and fidelity of each modules are discussed. The simulation results are compared with measurement result. Finally, the model is used to demonstrate a traffic trace which has not been shown in the measurements.

Chapter 6: Presents conclusions and recommendations for further study.

Chapter 2

Application Software and Network Protocols

2.1 Multimedia News-on-Demand Application

2.1.1 Introduction

The Multimedia-News-on-Demand application used in this study is a specific presentational application [25] developed by MCRlab (Multimedia Communication Research Lab) at the University of Ottawa, under the sponsorship of the CITR (Canadian Institute for Telecommunications Research). The application allows users to browse through items of current affairs and select a specific article for viewing. The users can interact with the presentation of the document in the same way as they would when viewing tapes from a VCR. The news documents consist of multiple media: text, audio and video, which are stored in a remote multimedia database. Through multiple servers, activities such as multimedia data retrieval and document delivery are performed continuously in real time.

2.1.2 Scenarios and Material Segmentation [26]

All the documents available in the prototype used for the measurements are a “cut” version of the same news article. The original document is 173 seconds long. After segmentation, the video and audio streams are stored as objects of 10 seconds (except for the last 3-second ones), thus generating 18 video objects and 18 corresponding audio objects. The text is manually transcript by playing the audio stream using an audio editor, thus gaining 24 text objects each including at most 2 lines (160 characters). The lists of size and displaying duration of the media objects (given in Appendix A) show the difference between the three media sources. Audio objects contain the same amount of data for each 10 seconds, while MPEG-compressed video ones are of various sizes. The video and audio data are chopped into objects of 10 seconds, while the text is

segmented according to synchronization requirements with the other two streams, so the text objects are different on both size and displaying duration.

Figure 2.1. Example of the Structure of a Temporal Scenario.

Although each medium may reside on its own server, a certain relationship still exists among the various streams during the presentation. In this application, the multimedia integration is described by scenarios, which represent the temporal relationships among the data objects (small segments) involved in the multimedia document. The temporal presentation scenario is stored in the multimedia database. Figure 2.1 gives an example of its structure. The scenario consists of a list of activities each including a list of objects and relations. Those objects should be displayed in synchronization, and the intermediate time and skew tolerance parameter between them are defined in the relation. For example, the first activity of the scenario is composed of 3 objects. The first one is a video object with object ID “glob_v01”, media_id “1”(means video), size equaling 682076 bytes and duration 9.8 seconds. The other two are the first objects of audio and text streams. The relation decides how a synchronization function at the client site deals with the network delay jitter of the objects (see Appendix B).

2.1.3 Multimedia Communication Protocols at the Application Layer

Communication protocols at the application layer, such as FTP, TELNET and SNMP, are very

common in the communication network field. They are designed to implement specific communication jobs. Similarly, in order to implement real time multimedia data delivery, a certain multimedia communication protocol is also required at the application layer. Although it is independent of network protocols, it works with them together to control the multimedia traffic.

In this application, the different media are transmitted through an ATM network for presentation on the client's workstation. Since the client's workstation may not have a large disk space dedicated to the multimedia news retrieval, the multimedia news document has to be progressively transmitted over the network. There are two basic approaches for delivering multimedia data. One is real time delivery scheme, the other is a store-and-forward scheme. In the latter approach, the multimedia delivery is performed as a file transfer. The multimedia document is played back after all its components arrive. This scheme requires simple synchronization control, but a huge buffer storage. Therefore, it may not be interactive. However, for the real time approach, a real time synchronization delivery method is required. This is realized by two steps in the Multimedia News-on-Demand application. First, before transmitting the media objects, a stream delivery schedule is computed to specify the time for the independent media servers to deliver their objects in order to meet the temporal relationship requested in the scenario. Second, at the receiver site a compensation mechanism is used to deal with the random disturbance introduced by the networks.

The real-time delivery mentioned here is a little different from what is implemented in most simulations regarding the ATM applications. For example, the video data here are transferred in a way of object by object (the segment of 10 seconds), instead of frame by frame, namely, in the application one video object is sent in every 10 seconds, while in most simulations one frame of video is sent in every $1/30$ seconds. Because the time interval of the transmission is not as fine as that in the simulations, the self-similar nature of VBR video source that is introduced by the MPEG IBP frame structure can not be expected to influence greatly the traffic burstiness.

In the multimedia news application, if the user requests to view a news document, the request is transmitted to the multimedia database. The database sends the presentational scenario to the

scheduler on the client's site. With this predefined scenario, a delivery schedule is computed so that the multimedia traffic can be delivered to reach the destination in time. At the same time, SSP (stream synchronization protocol) [25], a real time recovery process at the receiving end, preserves the synchronization among the streams carried over different TCP connections (in this application, one TCP connection for each media stream). It uses skew tolerance parameters as error thresholds to guarantee the control for the different streams. The communication sequences between client and server are described in Figure 2.2

Figure 2.2. The Communication Sequences between Client and Server.

In summary, the flow control contribution from this multimedia application protocol is performed at two levels: 1) delivery schedule available at the beginning of transmission; 2) SSP implemented at the client during the whole process of the communication. The flow control function of the delivery schedule is evident as it defines when and which media object should be sent, while SSP's effect on traffic is not so obvious since the client does not directly send feedback to the server about the status of the application. However, because it is according to the synchronized schedule (gained from SSP) that the client gets the data from the lower protocol layer, the data transmission may be blocked when synchronization delay occurs and the buffers at lower layer happen to be full. That is to say, it is through the lower protocol layers that SSP may

be able to generate the effect upon the traffic.

The software structure is described in Appendix B in order to provide a clearer description of the application protocol.

2.1.4 Network Protocol Layers

Figure 2.3. Protocol Layers Involved in the Application

The network protocol layers involved in the Multimedia News-on-Demand application are depicted in Figure 2.3. The higher layers include the application layer and the socket layer. The TCP/IP facility, which is provided by the IBM AIX 3.2.5 Operating System, runs over AAL5/ATM layer. In the following sections, a description of the basic functions of the TCP/IP and ATM layers will be given. Detailed information about running TCP/IP over ATM in IBM 3.2.5 Operating Systems will be supplied in Chapter 3.

2.2 TCP/IP

2.2.1 TCP Operation Principle

TCP is widely used on the current internet. It is a connection-oriented protocol and provides a reliable end-to-end transmission. The basic components of TCP include the slow-start, congestion avoidance, fast retransmit, and fast recovery algorithms[20, 21]. TCP controls traffic injected into networks by adjusting the size of a congestion window at the sender site. There are two phases of the window-adjustment algorithm called the Jacobson-Karels (JK) algorithm. Initially, a slow-start threshold is set to be half of the receiver's advertised window size. The TCP connection begins with a slow start phase, and the current congestion window is doubled each round-trip time until the congestion window reaches the slow-start threshold. Then comes a congestion avoidance phase, and the congestion window increases by roughly one packet each round-trip time. The congestion window is never allowed to be larger than the receiver's advertised window, which is referred to as the "maximum window". In addition to using retransmit timers, the sender uses the fast retransmit procedure to detect packet loss: if three duplicate ACK packets are received acknowledging a previously-acknowledged data packet, the sender infers that packet(s) has/have been dropped.

In the earlier Tahoe version of TCP (proposed in 1988), the sender reacts to a packet loss by setting the slow start threshold to half the congestion window, decreasing the congestion window to one packet, and entering the slow-start phase. In contrast, with Reno's fast recovery algorithm the TCP sender does not slow start after inferring that a packet has been dropped. Instead, the TCP sender effectively waits for one half a round trip time and reduces the congestion window to a half. The sender retransmits the dropped packet and uses incoming duplicate ACKs to clock additional outgoing packets.[27]

The basic TCP principles explained above is also described in Figure 2.4. It is assumed that the maximum window size is eight packets with the slow start threshold of four packets. The congestion window size starts with 1 packet and is doubled every round trip time until the slow start threshold is reached. Then it can be increased by one packet every round trip time if no congestion

happens until the maximum window size is fully used. Once some packets are lost, the congestion window will either be forced to start from one packet and experience slow start phase again (Tahoe version), or wait for half a round trip time and reduces the congestion window to a half (Reno Version).

Figure 2.4. Basic TCP Traffic Control Procedures. (Time out retransmission and Reno's fast recovery is not included)

Over the years many suggestions have been proposed on making TCP more suitable for high-speed large delay networks[30], but they have not been widely implemented yet. The congestion control algorithm that is commonly deployed in commercial implementation of TCP is still the one implemented in the Tahoe Version of 4.3 BSD UNIX. However, because every operating sys-

tem has its own memory management mechanism[23][28][29], the TCP implementation in each operating system may be slightly different from each other (A detailed discussion on this point will be presented in section 3.2). Thus performance issues had better be discussed associating the specific operating system in which the experiments are done.

2.2.2 Internet Protocol

IP is a main protocol at the network layer of the Internet protocol suite. It provides an unreliable, connectionless datagram delivery service between hosts attached to an TCP/IP internetwork. In this study, the connectionless IP datagrams are carried only on connection-oriented ATM networks. Therefore, the IP routing functionality is not emphasized in this thesis.

2.3 ATM and its Flow Control Mechanisms

In traditional data communication networks, congestion control schemes are mostly reactive, namely, congestion control only operates after congestion happens. For example: if a TCP sender does not receive an ACK in time-out after sending a packet, it will assume the packet is lost due to congestion. The corrupted packet will be transmitted again and again until the corresponding ACK is obtained, and at the same time the TCP congestion window size will be decreased in order to release the load in the congested network.

An ATM network is expected to operate at a very high speed. The result of congestion can be fairly serious even it lasts only for a short time. So the goal of ATM congestion control is not only to react to congestion, but also prevent it from occurring.

Under ATM Preventive Congestion Control, each traffic source has to provide the network a traffic descriptor during connection setup. The traffic descriptor consists of three primitive parameters: peak rate, average rate and peak rate duration. A Call admission control procedure is then used to make sure that the new traffic source can be supported without degrading the QoS of other

connections. Once the connection is established, a policing function is used to ensure that the source adheres to the negotiated parameters specified by the traffic descriptors.

Possibilities for the sources violating the negotiated traffic parameters exist. This may be due to the inability of the sources to accurately estimate the traffic characteristics before the actual communications or the attempts to obtain economical advantages. In any case, excessive cell losses may occur as a result of the unexpected traffic load. Congestion may occur and lead to degradation of network performance.

The effectiveness of various congestion control approaches in ATM networks is still being debated in standards forums. The Rate-based scheme and the credit-based scheme are two major feedback mechanisms to support the ABR service [31][32].

LB (Leaky Bucket), a traffic policing mechanism supporting VBR service, is widely used in current ATM products. The principle of LB is very simple as depicted in Figure 2.5. The token is generated according to specified token rate and sent to bucket if it is not full. Normally the input cells can enter into the network only when a token is available in the bucket. Otherwise two actions might be taken: first, the arriving cell will be discarded immediately; second, the cell will be marked as a low priority cell and sent as usual, but once congestion happens, or the occupied buffer space in switches is over a certain threshold, the marked cell will be first dropped.

Figure 2.5. The Principle of Leaky Bucket.

As indicated in Figure 2.5, LB has two major parameters, token rate and bucket size, which

can be varied to tailor the policing for particular requirements. The token rate is generally set to the policed traffic rate so that LB can never allow traffic rate that is higher than the token rate to enter the network. Adjusting the bucket size from one extreme to another represents a trade-off between the sensitivity to violation and the allowable burstiness of the arriving traffic.

In the event of congestion, the ATM network can apply an intelligent cell dropping mechanism to improve the overall throughput. In its simplest form, a selective (or partial) cell discarding mechanism can be applied so that when a cell is lost, all other cells belonging to the same packet may be discarded as well. In the implementation, once the switch drops a cell from a VC, the switch continues dropping cells from the same VC until the switch sees the ATM-layer-user-to-user (AUU) parameter set in the ATM cells header, indicating the end of the AAL packet. The end-of-packet cell (EOP) itself is not dropped. Since AAL5 does not support the simultaneous multiplexing of packets on a single VC, the AUU parameter can be used to identify the end of a sequence of cell belonging to the same packet.[17]

This mechanism, however, has a limited effect, as there may be other cells of the same (long) packet already transmitted down the link. An alternative strategy, the so called EPD (Early Packet Discard), proposes a mechanism that a switch will drop an entire packet whenever the proportion of the buffer in use exceeds a fixed threshold(i.e., prior to the buffer overflows). That is to say, ATM-EPD drops the completed packets arriving after the instantaneous queue size crosses the set threshold. In a way, the ATM switch in this case behaves as a conventional router. This scheme prevents transmission of useless cells on the congested link and reduces the total network load. The difficulty is the determination of the optimal value for the fixed threshold. Also, the fairness is not guaranteed since the cells that are not on the connection which has caused the congestion may be discarded.[18]

2.4 Performance of Running TCP/IP over ATM Networks

According to some simulation studies[17][18], the performance of TCP in a plain ATM

network (with no congestion control functions at the ATM layer) can be inferior to its performance in a conventional datagram network. Although TCP is designed to work in networks with no congestion control mechanisms below the transport layer, the use of a congestion control scheme is recommended at the ATM layer. Since each TCP segment is fragmented into a large number of ATM cells, the loss of a single cell triggers the retransmission of an entire TCP packet. Even worse, when a switch discards a cell, the remaining cells of the same TCP packet continue traveling towards their destination, wasting network resources. Though this problem is not fundamentally different from the fragmentation that occurs at the IP layer in a datagram network, the effect can be very serious in an ATM network because of the small size of the ATM cell. Indiscriminate dropping of ATM cells under congestion can lead to the collapse of throughput for an application. Therefore ATM layer congestion control schemes are definitely necessary to gain desirable network performance.[18]

Some simulations have been done to evaluate the support from various ATM congestion control mechanisms to TCP/IP[17][18][19], especially to provide ABR service. Using rate-based flow control under TCP causes the problem of fairness. TCP uses the mechanism that directly controls the connection's congestion window size. Because the average throughput for a connection is equal to the ratio of its average congestion window size to its average round-trip delay, achieving an even allocation of bandwidth through the allocation of window sizes alone is difficult. To achieve a fair allocation of bandwidth, the window sizes of different connections would need to tend toward values proportional to the round-trip time for their respective control loops. It is challenging for rate-based schemes, for, the data rate is always adjusted with the value that is either linearly or exponential proportional to it. With a fair scheduling discipline at network queues, a hop-by-hop credit-based flow control can achieve the goals of an ABR service by addressing two issues: first, it allocates to each connection a window size that is large enough to sustain the connection's fair share of the throughput; second, it allocate enough buffer space for each connection to avoid losing data.

As a popular ATM flow control mechanism, Leaky Bucket has also been investigated on its performance to support TCP. Some simulation results have shown that with LB the TCP

throughput might be much lower than the SCR (Sustainable Cell Rate)[2]. This is because the TCP congestion window size always keeps on increasing until it meets the maximum size or congestion happens. If bandwidth is shared by multiple TCP connections, once congestion happens, switches overflowing may introduce cell loss on multiple VCs at the ATM layer, which then causes packet loss on multiple TCP connections. The window size will be decreased on multiple TCP connections at the same time, which decreases network throughput greatly. [1]

Furthermore, some TCP throughput measurements indicate that deadlock might happen within a data transmission application running over TCP/IP/ATM. This is introduced by several factors including large ATM MTU value, Nagle's algorithm, TCP delay-ACK mechanism, and asymmetry of socket buffers size set at sender and receiver's end.[23][24]

To improve the network performance of TCP/IP over ATM, control loops at the ATM layer needs to interact efficiently with those at the TCP layer. Currently, the only interaction between congestion control at the TCP layer and the ATM layer is the packet loss, which is regarded by TCP as the indication of the congestion. If no packet is dropped, TCP will not be able to exercise congestion control. However, the main goal of ATM congestion control is to prevent congestion (or cell loss) from happening. Thus the conflict of reactive flow control of TCP and preventive congestion control of the ATM layer may be the chief difficulty for the two to work together. Further investigation on this is necessary.

Chapter 3

Measurement Environment

3.1 OCRIInet Connections

3.1.1 Introduction of OCRIInet

Figure 3.1 OCRIInet Topology.

As shown in Figure 3.1, OCRIInet, an ATM research MAN (metropolitan area network), employs fibre optic cables to link over a dozen of initial nodes including universities, technology companies, telephone companies and government laboratories. It is a regional laboratory which allows its members to probe the future by developing and demonstrating innovative equipments, techniques and services requiring broadband networks. It also provides a platform for the study of both quality and economics of delivering various services that exploit this new exciting medium and educational institution[33]. Also, OCRIInet is a large-scale, high-speed ATM network

consisting of a heterogeneous collection of ATM switches, computing platforms, middleware (distributed services) and so on, all of which must cooperate in order to make complex applications operate at high speeds.

3.1.2 OCRInet connection between Two Universities

In OCRInet, the University Of Ottawa and Carleton University connect to the Rideau Gate-

Figure 3.2 OCRInet Physical Connection Between OU and CU

way, which is a NT Magellan Gateway switch. The path from the gateway switch to both universities is made up of OC-48 fibre. Each end of the fibre is terminated on a FNT-150 which converts fibre signals to co-axial DS-3 signals. At the Carleton University site, the DS-3 is terminated on a Newbridge 36150 ATM switch, from which there is a short length of 100 Mbit/s Multimode fibre connected to a Sparc-20 work station: HERMES. The workstation is used to remote access and control the application running at the University of Ottawa. The connection to the University of

Ottawa is a little more complex. The DS-3 is terminated on a Newbridge 36150 ATM switch at the Administration Building. There is an OC-3 Multimode fibre connecting this switch to a Newbridge VIVID LAN switch at the Multimedia Communication Research Lab. Two IBM RS-6000s, CITR_C and CITR_S, along with a Multimedia News-on-Demand System on them, are connected to the VIVID via OC-3 multimode fibre.[34]

Figure 3.2 shows OCRInet physical connections between the two universities. The VP/VCs set up for the experiment are described in Figure 3.3. CITR_C and CITR_S are connected to OCRInet by VP/VC of 0/1026 and 0/1027 respectively, and the traffic over these two VCs is exchanged by the switch at Carleton University. The VP/VCs of 0/1028 and 0/1029 carry the tel-net information between HERMES and those two machines at the University of Ottawa.

Unfortunately, almost no further information can be collected about the OCRInet connections, except for what is demonstrated in this section. Till this project started, no detailed delay measurements had ever been done on the network. Also, little is known about the internal structure of the switches, such as buffer size of each port and variation of cell interarrival time, since they are seldom exposed by the vendors. This increases the difficulty on traffic analysis and modelling. Few ATM flow control functions are known on the switches along the connections. The Leaky Bucket algorithm implemented on the Fore ATM Interface cards might be the only ATM flow control mechanism on these OCRInet connections (see section 3.3).

Figure 3.3 The Assignment of PVCs of the Connections.

3.2 Running TCP/IP over ATM Networks in IBM AIX 3.2.5 Operating System Environment

In this section, data movement paths and traffic control functions regarding running TCP/IP over ATM will be further described associating the real operating environment of the experiments. The analysis will be the foundation of the modelling presented in Chapter 5.

3.2.1 Overview

In our experiments, the software examined is the MCR lab's Multimedia-News-on-Demand application [25] using Unix 4.3 BSD stream sockets on top of TCP/IP over ATM networks. The hardware consists of two IBM RS-6000 workstations connected on the OCRInet segment via ATM Fore MCA-200 Interface cards. The operating system is IBM AIX 3.2.5. Figure 3.4 shows the protocol layers and traffic control implementations at three levels.

User processes at the application layer interface to the network through the socket layer. The protocol modules for TCP and IP are below the socket layer in the kernel of the operating system

Figure 3.4. Flow Control Involved in Multimedia Application over ATM.

(IBM AIX 3.2.5). Data are passed between user processes and the protocol modules through the socket buffers. On message transmission, the data are first moved by the host processor from the user space to a kernel memory; then they are packetized by the protocol layers; finally the packets are delivered to the data link device driver for transmission. Figure 3.5 shows where the incremental overhead for packet processing, based on the packet size, occurs in a typical BSD 4.3 distribution [28]. In our case, the IP packets are segmented into ATM cells by the ATM FORE

Figure 3.5. The Incremental Data Operations at Each Layer.

MCA-200 cards at the end.

All of the application delivery schedules, TCP flow control, and ATM congestion control have effect on the multimedia traffic. At the application layer, data are moved to the socket layer according to the delivery schedule predefined in the scenario (see 2.1). Also the data are moved to the socket layer in a unit called user message size. This unit used to be defined as 1024 bytes in the application program, which was not a reasonable value to AIX 3.2.5 MBUF's structure. Currently, the unit has been changed into a user-defined one. With the various values of the unit, the network performance may vary as it is seen from the measurements demonstrated in Chapter 4.

In order to illustrate how TCP is implemented on IBM AIX 3.2.5, in the following parts we will introduce the implementation and the data movement at both the socket layer and the TCP layer. The interface layer (AAL/ATM) is to be described in section 3.3.

3.2.2 Socket Layer

The basic TCP principle is described in 2.2, while in real operating systems, the TCP

implementation is tightly related to memory management, which is implemented at the Socket layer. The main functions of the socket layer are: 1) to identify different applications on the same host; 2) to provide send and receive buffers for data transfer. The socket interface provides sockets, which are abstract objects that the applications can create, connect, and use for sending or receiving data. The sockets are typed, with each type providing different communication semantics. For example, a STREAM socket provides a reliable connection-based byte stream which may support out-of-band data (TCP connections), and a DATAGRAM socket provides connectionless and unreliable packet-oriented communication (UDP connections).[13]

In our case, three pairs of sockets are created to present three independent TCP connections each for video, audio and text streams respectively. Thus, each medium stream is ruled by its own TCP window.

In IBM AIX 3.2.5 operating System, the main components of MBUF memory management are the MBUF data structure, the MBUF memory, and a collection of supporting routines that manipulate the MBUF memory. MBUFs always reside in the physical memory, i.e., they are never paged out. MBUFs are 256 bytes long, out of which up to 236 bytes can be used for storing data. The remaining 20 bytes contain control fields that are used to manipulate both the MBUFs and the data. These are called small (or plain) MBUFs since the amount of data stored internally is limited to 236 bytes. When more space is required, an external page from a page pool (which is part of the MBUF memory) can be attached to a plain MBUF to form a clustered MBUF. These pages can store up to 4096 bytes. MBUFs (both plain and clustered ones) contain linked list pointers so that they can be easily connected into an “MBUF chain” to contain larger packets, and the protocol headers can also be connected to the MBUF chains.[29][35]

AIX provides a pool of free plain and clustered MBUFs and MBUF management services to various layers for their buffering needs. Depending on the amount of data being copied into the send socket buffer, a socket puts the data into either a plain or clustered MBUF. Once the data is copied into the buffer, the socket layer will call the TCP layer and passes it a pointer to the linked list of MBUFs (MBUF chain).

The data copy strategy of transmit socket layer in AIX 3.2.5 can be roughly described in

Figure 3.6. Data Distribution into (clustered) Mbuf's.

Figure 3.7. Unlike SunOS, IBM AIX does not expose its source code and implementation mechanisms to users, which might be the reason why most of research on the topic of TCP/IP/ATM is done with SunOS. Therefore, the data copy strategy of transmit socket layer in AIX 3.2.5 can not be obtained directly from publicly available material. Fortunately, the AIX operating system is compatible to UNIX 4.3 BSD, and its implementation of memory mechanism is expected to be similar to that of SunOS and other commercial operating systems. What is described in Figure 3.7 is an assumption based on description provided by IBM [29] and on reference of SunOS 4.1[13] and DEC OSF/1[28]. This assumption is the foundation of the OPNET model developed in Chapter 5. Simulation results will be compared with the measurements to verify the whole model as well as the assumption.

Network performance can suffer as a result of poorly written code. Whenever possible, AIX communication programs should read and write in quantities of either less than or equal to 935 bytes, or slightly less than or equal to 4,096 bytes (or multiples thereof). A read or write of less

Figure 3.7. Transmit Socket Layer Data Copy Strategy in IBM AIX 3.2.5.

than or equal to 935 bytes will be placed in one to four plain MBUFs; a read or write of less than or equal to 4,096 bytes (or multiples thereof) will make the efficient use of the clustered MBUFs that are used for writing larger than 935 bytes[29]. If an application software writes 1024 bytes at a time to the sender socket, each write to the socket occupies 1024 bytes of a separate clustered MBUF, which wastes a lot of buffer space and results in low throughput (see Chapter 4).

Another point worth mentioning is that when available space at the socket layer is not enough for the data coming from the application layer, the application process will be put in sleep status and transmission will be blocked until enough space is available. Therefore, when the network throughput is low or congestion happens, the communication processes will have to slow down,

instead of operating according to the delivery schedule defined in the scenario. For example, when one large packet is sent and the rest of sender socket buffer space is too small to contain the application data that need to be written, if, at this time, a congestion happens and no acknowledgment arrives to clear the sender buffer, the application process will be put to sleep. Thus, traffic at application layer is not only controlled by the predefined delivery schedule but also by the network performance at the lower protocol layers (TCP and ATM).

The TCP layer allocates a new MBUF for its header information, and copies the data in the send socket buffer into the TCP header MBUF, or into a newly allocated MBUF chain if there is room. If the data being copied is in clustered MBUF, the data will not actually be copied into the new pages. Instead, a pointer field in the new MBUF header (this header is part of the MBUF structure and is unrelated to the TCP header) is set to the point of the page containing the data, thereby avoiding the overhead of a 4096 bytes copy. TCP then checksums the data, updates its various state variables, which are used for flow control and other services, and then calls the IP layer with the header MBUF now linked to the new MBUF chain (This chain is called a packet at the TCP layer).

The socket buffer size is a per-connection user-specified parameter that controls the amount of memory the kernel reserved for the socket layer buffering. There are two distinct buffers associated with each socket: one for sending and one for receiving. Like other BSD systems, AIX 3.2.5 maintains a one-to-one correspondence between the maximum TCP window size and the buffer size allocated at the socket layer in the end systems for the TCP connections. A wrong choice of window size, such as one too small, or one leading to non-balanced sender and receiver buffer sizes, can result in unnecessary blocking and subsequent inefficient use of bandwidth. In order to avoid these mistakes and gain the highest network throughput, on both CTR_C and CTR_S, the size of sender and receiver socket buffers is set to be the maximum value of 64k (65536 bytes) [36].

The IP layer determines which interface the MBUF chain should be sent to, updates and checksums the IP part of the header sent from TCP, passes the chain (IP packet) to the interface

layer, where the device driver write routine is called to send the data. At this point, the control returns up the path to the TCP output routine, which continues sending data as long as it has more to send. When all data have been sent, the control returns to the application, which then runs asynchronously while the adapter transmits the data.

3.2.3 TCP Layer

In 4.3BSD Unix, TCP sends an acknowledgment during a normal operation when the application removes the data from the receive socket buffer. More specifically, during a normal data transfer (no retransmissions) an acknowledgment is sent if the socket buffer is emptied after removing at least 2 MSS (Maximum Segment Size), or whenever a window update cleans the receiver's window by at least 35%. This point is depicted in Figure 3.8.[10]

The packets smaller than both 35% socket buffer and 2 MSS might be processed by the TCP-

Figure 3.8. TCP Window Update Algorithm.

delayed acknowledgment mechanism. This mechanism works as follows: upon receiving of a small packet, TCP sets the flag `TF_DELACK` in the transmission control block for that connection. Every 200 milliseconds, a timer process checks these flags for all connections. For each connection that has the `TF_DELACK` flag set, the timer routine changes it to `TF_ACKNOW` and the TCP output function is called to send an acknowledgment immediately. TCP uses this mechanism to minimize both the network traffic and the sender processing or acknowledgments. The mechanism achieves this by delaying the acknowledgment of received data in order to accumulate more of the forthcoming data and to send a cumulative acknowledgment. In short, if there is no retransmission, the protocol may acknowledge data in two ways: 1) whenever

(enough) data are removed from the receive socket buffer and the TCP output function is called; and 2) when the delayed ACK timer process runs. [13]

Figure 3.9. TCP Segmentation and Nagle's Algorithm

The data need to be segmented into packets at both the TCP and the IP layer. MTU is an important parameter which influences the fragmentation. It is used to describe how many data can be sent in one piece over the networking hardware, and it is different for various networks: 1500 bytes for Ethernet, 4356 bytes for FDDI and 9180 bytes for ATM. MTU decides the value of MSS (Maximum Segment Size). TCP calculates the MSS to offer by subtracting from MTU the number of bytes required for the most common IP and TCP header sizes (in our case, both TCP header and IP header are 20 bytes, so ATM MSS = 9140 bytes).[17]

In general, messages larger than the MTU must be sent in multiple pieces (fragmented and

defragmented). Fragmentation is implemented in both IP and TCP. TCP does its own fragmentation specifically to avoid IP fragmentation. TCP segmentation of the application byte stream and Nagle's algorithm are shown in Figure 3.9[23, 37].

Nagle's algorithm is introduced as a solution to the "small-packet problem". It is observed that TCP sent many small segments which resulted in the unnecessary header and processing overhead. Nagle's algorithm inhibits the sender from sending TCP segments which are smaller than the assigned MSS if any previously transmitted data on the connection remains unacknowledged. Therefore, switching off Nagle's algorithm is necessary for applications which send a small amount of data with no replies, such as a stream of mouse events, which have no data in the reverse direction. However, Nagle's algorithm is used in the default TCP configuration, and it is highly recommended on both telnet and ftp connection. In our experiment the Nagle's algorithm is turned on by default.

Nagle's algorithm attempts to reduce the congestion in the network by allowing only one unacknowledged small packet in the network at any moment. Unfortunately, this algorithm might interact badly with the delayed acknowledgment occurring at the other end of the connection. After a small packet is sent, we may have an insufficient amount of remaining data to make a "large" packet. To obey Nagle's edict, this packet can not be sent before the previous small packet is acknowledged. However, the receiving side of TCP processing delays the generation of the acknowledgment when receiving the small packet, since there is likely to be more data following soon. Therefore, after the small packet is sent, there will be a long wait before the acknowledgment is generated and the throughput will be decreased. This phenomenon may occur in a high-speed network which has a large MTU, since only the packets larger than MSS can be sent before previous packets are acknowledged.

3.3 ATM Interface Layer

3.3.1 Introduction

The network interface layer is composed of hardware device drivers and their interfaces. A

host may be connected to more than one network and thus, have more than one interface. For example, CITR_C (or HERMES) is connected to an Ethernet and an ATM network, so it has two network interfaces for the Ethernet and the ATM connection respectively. The protocols choose a network interface to output packets as part of their routing decision. To applications, the data transmission through TCP/IP over ATM is the same as that through TCP/IP/Ethernet except by selecting different network interfaces. A protocol configuration with two network interfaces each for Ethernet and ATM is shown in Figure 3.10 [13].

Figure 3.10. The Protocol Configuration with ATM and Ethernet Interfaces.

The Fore System cards, the ATM end system interfaces used in this project, provide the workstations with dedicated fiber-optic connections to ATM switches, including with the adapter — a device driver supporting the TCP/IP protocol suite as well as AAL 3/4 and AAL5.

The Fore systems card's advanced cell processor architecture provides optimized on-board cell processing functions including standard ATM cell segmentation and reassembly. The software driver supplies a high-performance packet-level interface to the cell-processing engine. Firstly it identifies the data packets to be communicated over ATM, and then it passes them to the cell processing engine. The architecture of the card's cell processor is shown in Figure 3.11 [38].

As shown in Figure 3.11, Fore card supports two routes for applications to access the ATM-

Figure 3.11. Fore System Card Advanced Cell processor Architecture

based functions. The first one is through socket layer and TCP (or UDP) /IP, which is used in the News-on-Demand application. The application makes standard UNIX socket calls to TCP. The socket is identified by the combination of port number for TCP and IP addresses, and often by an additional identifier called the protocol family. The application must first request TCP to open three connections between the client and the server before the multimedia data can be transferred. The IP traffic is passed to an IP handler, which has access to an AAL, ARP cache, and SPANS signaling processor. At the end, the traffic is carried by the assigned VP/VC at the ATM layer. As the three TCP connections are supported by one VC, the three media streams gain the same QoS from the network.

The second route is API (Application Program Interface), a path enabling an application program to run directly over the ATM-based capabilities. It is critical in closing the gap between the

ATM-base capabilities and the application. The ATM API for an existing device driver would ideally present the same software interface, therefore requiring no application change from one driver to another one. Unfortunately, most of the present APIs operate only in a very specific environment and support limited interpretability, and they do not either perform functions necessary in ATM, such as point-to-multipoint, specification of traffic parameters, support for connection-oriented signaling, and support for multiple Qualities of Service [39].

3.3.2 Support IP/ATM (see Appendix B)

3.3.3 Traffic Control Function

A simple leaky bucket algorithm is implemented at the Fore MCA-200 card. As indicated in Figure 3.12, the bucket size is fixed equaling to 1 cell, and the users may only be able to specify the PCR for each VC. The input buffer size is unknown to the user, but since the RAM on the card is 256 kbytes, the input buffer size can not exceed 256 kbytes.

Figure 3.12. LB Implemented on ATM Fore Card.

3.4 GN Nettek Interwatch 95000

3.4.1 Introduction

The GN Nettek Interwatch 95000 is an analyzer designed “from the ground up” to handle ATM applications. In addition to RISC processors in all card ports, parallel processing is used to share the analysis workload. It combines its architecture, software and storage capabilities to keep track of the volume and complexity of data in ATM applications. It uses SNMP and can be remote

controlled by an X-terminal. It interprets data and displays through an intuitive X-Window/Motif user interface. It has common clock synchronization across all of its ports. Events and frame contents on the LAN, WAN or ATM are identified, timestamped and correlated with microsecond accuracy.

The test application of the GN Nettetst Testset used for our experiments is called Non-intrusive Monitoring. In this type of measurement, test ports are provided by the equipment under test, so the services provided by the ATM switches are not affected by interWATCH 95000. User traffic is intact and no test traffic is introduced to the links so that ATM layer monitoring and analysis tests can be performed on the user traffic[40].

3.4.2 Measurement Connections

Figure 3.13 The Connection of GN Nettetst Testset and Newbridge Switch at Carleton U.

The measurement connection is set up as described in Figure 3.13. The GN Nettetst Testset can only handle the traffic along one direction, so the traffic that can be captured is the multimedia traffic from the server to the client with VP/VC of 0/1026 and the acknowledgment traffic from client to server with VP/VC of 0/1027.

3.4.3 Measurement Functions

The cell capturing function implemented at GN Nettetst Testset can be roughly described in Fig-

Figure 3.14. A capture buffer, which can be as large as 16 MB, is used to contain the captured cells

Figure 3.14. Cell Capturing Function of GN Nettek Testset.

and their information, such as the priority of the cells, whether the cell is congested or not, the interarrival time between cells. The information can be shown in “Decode Display” while cells are being captured. An example of the Decode Display is given in Figure 3.15. The summary of the information shown at “Decode Display” can be saved as an ASCII file, which can be used for later analysis.

FrameNo	Timestamp	Delta time	Statue	VPI	VCI	(con't)
000000001	10/17/95 11:31:43		Good	0	1009	→
000000002	10/17/95 11:31:43	0.000010	Good	0	1009	
000000003	10/17/95 11:32:17	33.966474	Good	0	1009	
(con't)						
Layer	Cell Typed					
ATM	Not Congested	High Priority				
ATM	Not Congested	High Priority				
ATM	Not Congested	High Priority				

Figure 3.15. An Example of “Decode Display”.

In Figure 3.15, FrameNo is the sequence number of each captured cell. Timestamp is the individual arrival time for each cell, and delta time is the inter-arrival time of two adjacent cells. As the figure shows, cells’ status, VPI/VCI, protocol layer and type are also recorded.

GN Nettek Testset uses filters to refine the cells that should be captured so that the users can

collect only the cells they need, such as those with certain VPI/VCI, certain PTI (user data or OAM data or Congested cells), certain CLP (high or low priority), or with HEC error. In the above test, a filter is defined as VPI/VCI of 0/1009, so the machine collects only the cells with VPI/VCI of 0/1009 and ignores all of the other cells. This enables the resource of the machine (such as memory) to be dedicated to the cell collection for special goals.

3.4.4 Drawbacks

The measurement of multimedia traffic over OCRInet exposed a few problems of GN Nettest Testset. The major one is that the current version of GN Nettest Testset can not capture every cell from a high-speed traffic stream. The problem comes from the fact that the cells are processed by a software, which can not run as fast as the hardware. During the cell collection, first the cells are time-stamped by the hardware, and then sent into a 32k buffer, where they are processed by the software at the rate of around 7,000 cells per second. If the cells arrive at a rate above the software processing rate, the coming cells will accumulate in a 32k buffer. Once the occupied space in the buffer is over a threshold, the coming cells are “dropped”. The “dropped” cells are not discarded physically, instead, they just pass through the machine without leaving any trace. Finally only those processed cells are sent to the capture buffer and kept as the measurement result.

In order to overcome the drawback, a fast capture card has been designed to capture cells at full line rate of DS-3[41]. Unfortunately, the fast capture option is not available on our machine, so the machine can only capture the cells at a comparable low rate. If the observed application operates slowly, such as the MPEG-player telnet over ATM, the machine is able to capture each cell, but to some fast applications, parts of the traces will be missed.

Chapter 4 Multimedia Traffic Measurement and Analysis

4.1 Measurement Points

As shown in Figure 4.1, in our experiment the measurements and observation are done at both the application layer and the ATM layer. At the application layer, for each media stream (one for video, one for audio and one for text) the data amount and time stamp are recorded whenever the data are moved to the socket layer. The data unit (user message size) is defined as SOCK_PAC_SIZE in the program, which is set by users before starting the application. When the server starts to send a segment of media material, the program enters into a loop in which the application data are written into a socket buffer with a unit size of SOCK_PAC_SIZE until all of the data of that segment are sent. The measurements have been done with various values of SOCK_PAC_SIZE (935, 1024, 4096 and 8192 bytes) in order to see its influence on the generated traffic.

Figure 4.1 Measurement Points.

The ATM layer observation is accomplished by using a GN Nettek testset, which records the arrival time of each captured cell during an observation interval. The analyzer has been installed at the ports of the Newbridge switch at Carleton University. The captured cells from the server to

the client (multimedia traffic) use the VP/VC 0/1026, and those from the client to the server (acknowledgment traffic) use the VP/VC of 0/1027.

In order to characterize the traffic, it is desirable to observe traffic at multiple protocol layers and at various locations simultaneously while running the application. However, for various reasons outlined below, the measurements are limited to the above two points. The traffic traces were not recorded at the application layer of the client, because the jobs at the client site are very complex (including three media players to synchronize and display the media in real time), measurement overhead might have an undesirable effect on the application. Finally, lack of an OC-3 Interface card at the GN Nettest testset makes it impossible to perform measurements at the ports of CTR_C and CTR_S.

Measurement at the TCP layer was also hard to implement in our experiment. The protocol modules for TCP and IP are in the kernel of the operating systems, so the measurement code can not be implanted into the TCP layer unless the kernel of operating system is modified. Using measurement tools, on the other hand, provides a simpler and easier alternative to the problem. Various mechanisms have been developed to do measurements at the TCP layer. The most suitable one is a tool called "TCP_dump", which records the time stamps of all packets for a given communication session. This tool is ideal for our project. However, in order to avoid the large measurement overhead, the execution of "TCP_dump" must be performed on a third computer other than the two workstations running the examined application. Therefore, the traffic traces at the TCP layer will only be obtained when both the tool and the third computer are available for our experiments in the future. [42]

4.2 Overview of Measurement Experiments

The measurements have been done with various values of SOCK_PAC_SIZE (935, 1024, 4096 and 8192 bytes) to see its influence upon the whole traffic. A list of general information pertinent to the tests performed is provided in Tables 4.1 and 4.2. For every SOCK_PAC_SIZE value, a measurement has been done at least twice, each of which includes the measurement at the

application layer (on three media streams), and observation at cell level on the multimedia and acknowledgment traffic separately.

Besides varying the value of SOCK_PAC_SIZE at the socket layer, various other setup parameters exist at each of the protocol layers. At the application layer, a predefined scenario (see Appendix A) determines the delivery schedule of the server. At the socket layer, the socket buffer size at both the server and the client is 64kbytes. At the TCP layer, Nagle's algorithm is set on by default. At the ATM layer, Peak Cell Rate is set as 5Mbit/s on the Fore ATM Interface Cards (equals to a cell rate of 11,792.5 cells/sec, or a cell interdeparture time of 84.8 μ s).

Test	SOCK_PAC_SIZE (bytes)	Data File Containing MM Cells Record	Duration of Data Collection (secs)	Number of Captured Cells	Amount of Application Data (cell)	Estimated Percentage of Cells without Record
0	1024	M2610.tst	10.45882	18500	19655	5.90%
1	1024	M2611.tst	23.95940	43818	44852	2.52%
2	1024	M2612.tst	23.12939	43044	43428	0.08%
3	4096	M2641.tst	40.14000	43153	74134	42.00%
4	4096	M2642.tst	40.10572	43291	74003	41.60%
5	8192	M2681.tst	40.14198	42953	74134	43.00%
6	8192	M2682.tst	40.51720	43556	74682	41.67%
7	935	M2691.tst	41.64600	43804	76257	42.55%
8	935	M2692.tst	47.80879	40343	88198	54.25%

Table 4.1: General Information About Measurements on MM Traffic

Test	SOCK_PAC_SIZE (bytes)	Data File Containing ACK Cells Record	Duration of Data Collection (secs)	Number of Captured Cells
0	1024	M2710.tst	188.3196	1389
1	1024	M2711.tst	120.6548	1338
2	1024	M2712.tst	118.1036	1630
3	4096	M2741.tst	171.6856	1075
4	4096	M2742.tst	171.7731	1082
5	8192	M2781.tst	171.6183	1064
6	8192	M2782.tst	171.7465	1066

Table 4.2: General Information About Measurements on ACK Traffic

Test	SOCK_PAC_SIZE (bytes)	Data File Containing ACK Cells Record	Duration of Data Collection (secs)	Number of Captured Cells
7	935	M2791.tst	175.0821	1111
8	935	M2792.tst	171.7234	1113

Table 4.2: General Information About Measurements on ACK Traffic

Note: (1) M26** files contain the data describing the multimedia traffic from the server to the client; and M27** are for the acknowledgment traffic from the client to the server. These files are under the directory /home/hwm/data/cell/;

(2) Test0 was performed at the end of 1995, when the measurement at application layer was not available yet. Other tests were done in May, 1996;

(3) Amount of User data is an estimated value of the data transmitted from the server to the client during the measurement;

(4) The percentage of cells without records is gained by comparing the amount of user data and the number of captured cells. It is calculated without considering the overhead of TCP/IP, so the actual values should be a little bit larger than those in Table 4.1.

Due to the high-speed nature of the ATM layer and the inability of the GN Nettek testset to operate at these speeds, the record of multimedia traffic at this layer is incomplete. The percentage of the cells that are not captured has been estimated for each test (the last column in Table 4.1). However, some analysis can still be done over parts of the collected data because:

- Cell record loss with the SOCK_PAC_SIZE of 1024 bytes seems to be comparably low;

The “cell loss” percentage is estimated to be as low as two to five percent. It is, thus, reasonable to assume that the data collected in Test 0, Test 1 and Test 2 can accurately describe the multimedia traffic at the ATM layer. In fact, most measurement results at the cell level presented in the following sessions come from the analysis on these three tests.

- The initial segment of traces can be expected complete since the cell record loss occurs only when the processing buffer of the GN Nettek testset is full;

Although the other test results at the cell level are incomplete (approximately half of the cells are not captured), the initial part of traffic traces still provides us with some information (see section 4.4.2.5).

- The traffic of acknowledgment is at a very low rate, so its record can be assumed complete.

The measurement results at the application layer are under the directory /home/hwm/data/

socket/. Table 4.3 contains the general information about these files.

Test	SOCK_P AC_SIZE (bytes)	Audio Traffic Data File	No of Audio Messages	Text Traffic Data File	No of Text Messages	Video Traffic Data File	No of Video Messages
1	1024	sa11t	3732	st11t	24	sv11t	11243
2	1024	sa12t	3732	st12t	24	sv12t	11243
3	4096	sa41t	938	st41t	24	sv41t	2819
4	4096	sa42t	938	st42t	24	sv42t	2819
5	8192	sa81t	470	st81t	24	sv81t	1415
6	8192	sa82t	470	st82t	24	sv82t	1415
7	935	sa91t	4095	st91t	24	sv91t	12310
8	935	sa92t	4095	st92t	24	sv92t	12310

Table 4.3: The General Information About the Measurements at the Application Layer

4.3 Data Processing and Information Extraction

The raw data obtained from the measurement is the timestamp of each captured cell at the ATM layer and timestamp of each user message at the application layer. Some programs have been developed to extract relevant information from these raw data. For instance, from the timestamp it is easy to determine cell interarrival times and interdeparture times of user messages. Also, the number of arriving cells within 1 ms can be determined. In other words, the number of incoming cells are counted for every 1ms time slot since the beginning of the trace. Using analysis tools, such as Testool and GNUplot [43,44], to examine the cell interarrival times, user message interdeparture times, and cell rate (based on 1ms interval), provides us with a clear view of the traffic behavior, from which several conclusions can be drawn.

4.4 Measurement Results and Data Analysis

In this section, the analysis of the measurement results will be presented along with the concluding arguments.

4.4.1 Delay Variation Introduced by Network

The large delay variation introduced by the network is indicated in the measurement result. As we know, a peak cell rate of 5Mbit/s has been set on the Fore Interface cards of CITER_C and CITER_S, which implement the Leaky Bucket algorithm to do flow control at the cell level. The cell rate coming out of the card, thus, would never exceed the PCR. However, at the switch of Carleton University, more than half of the captured cells are at a rate higher than the defined PCR (as showed in Table 4.4). We can not detect which ATM switch is responsible for this problem.

Test	Percentage of the Cells at a Higher Rate
0	83.19%
1	56.02%
2	55.91%
3	55.89%
4	55.50%
5	55.85%
6	56.00%
7	55.90%
8	55.66%

Table 4.4: Percentage of Cells at Rate over PCR Set on Fore Card.

The cell delay variation is also shown in cell level traffic in Figure 4.2 and Figure 4.3. The figures demonstrate the cell interarrival times. The x-axis is the sequence number of each cell but the first, and the y-axis is the interarrival time between each cell and the one before it. The interarrival time corresponding to 5Mbit/s is $84.8\mu\text{s}$. From the trace in Figure 4.2 (gained from M2610), however, it seems that the switches work in a processing mechanism similar to the store-and-forward applied over groups of 4-6 cells. The interarrival times in a group can, on average, be as small as $12\mu\text{s}$, and the interarrival times between groups are as large as $420\mu\text{s}$ on average. Such a mechanism totally destroys the flow control implemented on the Fore card and introduces unnecessarily burstiness in the traffic. The cell delay variation introduced by the network seems

switch at Carleton University has been changed from a 4-port model to a 16-port model.

Tables 4.5 - 4.7 give the distribution of the cell interarrival times of M2610, M2611 and M2612 respectively. The cell interarrival times are concentrated in the three intervals highlighted in the tables. The cause of the large cell interarrival times in the groups of 1 will be explained in section 4.4.2. The other two groups (group 4 and 5) of M2610 show the store-and-forward mechanism of the switches. Group 4 contains the interarrival times of each group, while the

Group	Interval of Cell interarrival times (ms)	Number of Cell interarrival times	Percentage	Average (ms)
<i>1</i>	<i>[103.67, 200.44]</i>	<i>52</i>	<i>0.28%</i>	<i>169.64</i>
2	[20.01, 54.95]	4	0.02%	35.42
3	[1.28, 9.96]	5	0.03%	5.70
<i>4</i>	<i>[0.32, 0.584]</i>	<i>3049</i>	<i>16.48%</i>	<i>0.42</i>
<i>5</i>	<i>[0.01, 0.053]</i>	<i>15389</i>	<i>83.19%</i>	<i>0.012</i>

Table 4.5: Distribution of Cell interarrival times of M2610.

Group	Interval of Cell interarrival times (ms)	Number of Cell interarrival times	Percentage	Average (ms)
<i>1</i>	<i>[106.09, 238.76]</i>	<i>118</i>	<i>0.27%</i>	<i>165.78</i>
2	[25.53, 93.87]	9	0.02%	55.39
3	[1.02, 7.43]	15	0.03%	2.19
<i>4</i>	<i>[0.1, 0.97]</i>	<i>5028</i>	<i>11.47%</i>	<i>0.114</i>
<i>5</i>	<i>[0.02, 0.095]</i>	<i>38647</i>	<i>88.2%</i>	<i>0.085</i>

Table 4.6: Distribution of Cell interarrival times of M2611

Group	Interval of Cell interarrival times (ms)	Number of Cell interarrival times	Percentage	Average (ms)
<i>1</i>	<i>[100.9, 223.3]</i>	<i>112</i>	<i>0.26%</i>	<i>165.99</i>
2	[21.13, 9.65]	10	0.02%	70.48
3	[1.05, 9.65]	13	0.03%	3.23
<i>4</i>	<i>[0.1, 0.92]</i>	<i>4913</i>	<i>11.41%</i>	<i>0.113</i>
<i>5</i>	<i>[0.011, 0.095]</i>	<i>37995</i>	<i>88.27%</i>	<i>0.085</i>

Table 4.7: Distribution of Cell interarrival times of M2612

interarrival times between each cell in one group are shown in group 5. The ratio of group 4 to group 5 is around 1:5, therefore most of the cells are sent in a group of 6, which matches the result given in Figure 4.2. The traffic seems smoother in M2611 and M2612 than in M2610, because the difference of the average of group 4 and of group 5 is much smaller than that in Table 4.5. Also, the average of group 5 is $85\mu\text{s}$, which is much higher than that of M2610 and is identical to PCR set on the Fore Cards. This consistence is also found in the traces captured in other tests. These two groups of cell interarrival times will be used as sample data to describe the service rate at the ATM layer for designing the simulation model in Chapter 5.

Switches may be one of the causes of the higher cell rates observed in the experiments. In packet switches, buffers are required to decrease cell loss probability. However, the use of the buffers will introduce the variation on the cell interarrival time since cells may experience different delay. At the same time the scheduling mechanism implemented on switches, such as store-and-forward may introduce large cell interarrival time variation. The variation of cell interarrival time, as such, can not be avoided for ATM switches. In fact, it is an important parameter describing the performance of an ATM switch. A sequence of tests have been done to compare various ATM switch performance under several kinds of network conditions [46], in which the Newbridge switch 36150 is reported to have a very high value of this parameter.

Obvious delay variations have been observed from the measurements in [15], in which the testbed consists of Fore ASX-100 switches, which have better performance in terms of the variation of cell interarrival time than the Newbridge switches. The measurements also showed that the more network nodes the traffic flow passes through, the more variation in cell interarrival time would be introduced. Similar to our measurements, the experiments in [15] are done over a lightly-loaded network (with almost no other traffic over the network). Thus, the problem would be expected to become more serious when additional traffic is introduced into the network.

As the Newbridge 36150 and Fore ASX-100 are switches widely used in most of the ATM testbeds, the problem discovered in [15] and this thesis may not be unique for these two testbeds, though no other work has been reported addressing this kind of problems on other ATM networks.

There is a concern regarding the effect of switches on traffic streams. Since the congestion control at the ATM layer is preventive instead of reactive, the mechanisms that it uses, such as CAC (Call Admission Control) and UPC (Usage Parameter Control), are supposed to keep the network from becoming congested. They would work efficiently only if: firstly, the traffic descriptors can fully describe the traffic; secondly, traffic is successfully controlled at the interface; and finally, the traffic in the network retains the characteristics it had at the UNI, or is controlled successfully at each switch. In most of the research efforts reported, network delay variation is seldom considered and traffic is expected to be identical anywhere in a network. If the CAC and UPC are implemented very well at the UNI, the traffic is assumed to be policed. However, if the traffic in the network is drastically different than where it entered to the network, congestion may still occur no matter how perfectly the flow control has been implemented at the UNI. The demonstration in this section indicates that the network delay variation is so serious that it can not be ignored. The traffic in the network may be different from one place to another in the network due to the delay variation that is possibly introduced by the switches. Therefore, flow control at one spot is hardly able to guarantee the satisfactory performance for the whole network. Efficient congestion control and QoS support are definitely needed at various nodes in a network.

4.4.2 Low Traffic Rate with 1024 bytes as SOCK_PAC_SIZE

4.4.2.1 The Cause of Low Traffic Rate

As we know, the GN Nettek testset is unable to operate at high speed, thus, if the sustained rate of the captured traffic is high, the trace recorded may be incomplete. Table 4.1 indicates that in most of the measurements around 50% cell are “lost” in the captured traces, except when SOCK_PAC_SIZE is 1024 bytes. The traffic rate of the first three tests is low enough for the GN Nettek testset to capture most passing cells.

Two factors contribute to the low traffic rate with a SOCK_PAC_SIZE of 1024. The first one could be the structure of Mbuf of the AIX 3.2.5 operating system. AIX 3.2.5 provides two kinds of Mbufs, plain Mbuf, which can store data up to 236 bytes, and cluster Mbuf, which can store as many data as 4096 bytes. Due to the specific nature of Mbuf memory management in AIX 3.2

(see 3.2.2), whenever possible, AIX communication application programs should read and write in quantities of either less than or equal to 935 bytes, or slightly less than or equal to 4,096 bytes (or multiples thereof). A read or write of less than or equal to 935 bytes will be placed in one to four Mbufs; and a read or write of less than or equal to 4,096 bytes (or multiples thereof) will make efficient use of the clusters that are used for writes larger than 935 bytes. When application software writes 1024 bytes at a time to sender socket, each write to the socket occupies 1024 bytes of a separate cluster, in which 3072 bytes of space is wasted. The application can fill the socket buffer (64 kbytes in our case) with just 16 writes of 1K each. Thus, most of time the 64-kbyte socket buffer serves as a 16-kbyte one. This would waste time as well as memory.

Another cause of the low traffic rate is due to the use of Nagle's algorithm and delayed acknowledgment mechanism. In 4.3BSD Unix, during normal data transfer (no retransmissions) a TCP acknowledgment is sent if the socket buffer is emptied after removing at least 2 MSS, or whenever a window update cleans the receiver's window by at least 35%. Packets smaller than both 35% of the socket buffer and 2 MSS might only be acknowledged by a delay-ACK generated every 200 ms. On the other hand, Nagle's algorithm inhibits the server from sending TCP segments which are smaller than the assigned MSS if any previously transmitted data on the connection remain unacknowledged.

The use of Nagle's algorithm and the delay-ACK have an impact on the traffic rate, depending on the packet size being used. For the case of a packet size of 1024 bytes, after the sender starts to write 1024 bytes of data to the socket layer, it will form a packet of 1024 bytes and send it to the receiver. Since the packet of 1024 bytes is smaller than 35% of the receiver socket buffer ($64k \times 0.35 = 22938$ bytes), it will be treated as a small packet and will not be acknowledged immediately. The sender continues to copy the data into the socket until the data unsent is large enough (over MSS 9148 bytes) to be sent before the first small packet is acknowledged. However, even the packet of 9148 bytes is too small to trigger a normal ACK. At this time, though the data can still be copied into the socket buffer, the left buffer space is not enough to form another packet of MSS. This means that the sender has to wait for the delay-ACK before sending any data. Now a deadlock occurs because the packets received at the client side are not large enough to invoke a

normal ACK and the packet at the sender side is too small to be transmitted before the ACKs of the previous packets are received. The transmission operates in a stop-and-wait manner and forces the throughput to be very low. An example trace of this mode of operation is given in Table 4.8 and Figure 4.4.

Step	Write data	Data in Sender Socket Buffer (bytes)	No of Unacked Bytes	Packet on the Wire	Data in Receiver Socket Buffer (bytes)	ACK or not	Comments
1	write(1k)	1024	1024	>1024>	1024	No	too small to trigger an ACK
2	write(1k)	2048	1024		1024		continue to copy data
3	write(1k)	3072	1024		1024		
4	write(1k)	4096	1024		1024		
5	write(1k)	5120	1024		1024		
6	write(1k)	6144	1024		1024		
7	write(1k)	7168	1024		1024		
8	write(1k)	8192	1024		1024		
9	write(1k)	9216	1024		1024		
10	write(1k)	10240	10172	>9148>	10172	No	Form a packet large enough to be sent
11	write(1k)	11264	10172		10172		continue to copy data
12	write(1k)	12288	10172		10172		
13	write(1k)	13312	10172		10172		
14	write(1k)	14336	10172		10172		
15	write(1k)	15360	10172		10172		
16	write(1k)	16384	10172		10172		Socket buffer is full, no more data copied before the delay-ACK

Table 4.8: An Example of Stop-and-wait Manner.

Note (1) After 16 writes, the sender socket buffer (64k) is full and no more data can be written into it. Thus the next packet can only be sent after the delayed acknowledgment is received;

(2) According to the design of 4.3 BSD Unix, even after the delay-ACK cleans the 10172 bytes space in the sender buffer, no more data can be copied into sender buffer before the left 6212 bytes data is sent. These 6220 bytes will introduce another stop-and-wait cycle.

Figure 4.4. An example of stop-and-wait manner.

Strictly speaking, the word “deadlock” used in this thesis is different from the one usually defined in operating systems, which can only be solved by resetting the systems. Here “deadlock” is used to describe the situation where both the sender and receiver must stop to wait for the response from the other side. This sort of deadlock can be broken by delayed ACKs generated every 200 ms. In the following sections, the deadlock will be verified from the various traces.

4.4.2.2 Verify the Problem from Cell interarrival times of M2610, M2611 and M2612

The deadlock can be verified by the measurement at the ATM layer shown in Figure 4.5. Most cell interarrival times are as small as shown those in Figure 4.2 and 4.3, while occasionally a few appear to be as large as over 0.15 (see Figure 4.5), which indicates that the connection keeps idle for quite a long time. Because the deadlock forces the application to execute in a stop-and-wait manner, the cell interarrival time between the first cell of a packet and the last cell of the previous packet can be as large as approximately 0.2 seconds (the cycle of generating a delay-ACK) if their transmission are separated by a deadlock. These large cell interarrival times are gathered in the group 1 in Tables 4.5–4.7. Group 1 in Table 4.5 contains 52 cell interarrival times, and the record duration is 10.45 seconds ($(52 \times 0.2 = 10.4)$). Thus, each large cell interarrival time in group 1 can be regarded as an indicator of breaking a deadlock.

Figure 4.5. Influence of Nagle's Algorithm and Delay-ACK on MM Traffic when SOCK_PAC_SIZE is 1024bytes.

4.4.2.3 Verify the Problem by Cell rate of M2610, M2611 and M2612

The problem can also be shown by using the cell rate. As indicated before, the cell rate is calculated over the time interval of 1 millisecond. Figure 4.6 is a typical segment of the cell rate

Figure 4.6. A Segment of Sample Path of Number of Coming Cells in 1 ms of M2610.

trace of M2610, and it looks very similar to those of test 1 and 2. The x axis indicates time in units of 1 ms, and the Y axis shows the number of coming cells during 1 ms. The effect of Nagle's algorithm and delayed acknowledgment may be seen from this graph: the connection is busy only for the first 30-40 ms and then deadlock occurs, so the connection remains idle until the delay-ACK breaks the deadlock and the transmission continues. Table 4.9 gives the distribution of cell rate in each of the 3 tests. The average data rate required by the application is 0.705 Mbit/sec, so around 80% of the bandwidth of 5Mbit/s is not used. This point is verified by the percentage of the time slots in which no cell arrives. The 11 or 12 cells arriving in 1 ms correspond to the bit rate of 4.67 ~ 5.08 Mbit/s, which is close to the PCR set on the Fore Card. As the cells are aggregated on the basis of each 1ms, the cell rate obtained is equal to what we will get when doing a performance measurement over time intervals of 1ms. From Table 4.9 the traffic seems to correspond to the PCR set on the Fore card very well because the cell rate seldom exceeds 5 Mbit/s (i.e., 12 cells/ms). In such a coarse measurement, no indication can be found about the cell delay variation demonstrated in section 4.4.1: the traffic burstiness caused by the network delay variation. In conclusion, the misbehavior at the ATM layer may be very subtle and may only be detected by high-resolution measurements on a fine time interval.

No of Cells	No of time slots (M2610)	Percentage (M2610)	No of time slots (M2611)	Percentage (M611)	No of time slots (M2612)	Percentage (M2612)
0	8933	85.42%	19945	83.24%	19204	83.03%
1	2	0.02%	32	0.13%	21	0.09%
2	4	0.04%	25	0.1%	33	0.14%
3	2	0.02%	26	0.11%	22	0.1%
4	25	0.24%	50	0.21%	28	0.12%
5	0	0%	31	0.13%	33	0.14%
6	29	0.28%	21	0.09%	26	0.11%
7	1	0.01%	29	0.12%	28	0.12%
8	1	0.01%	25	0.10%	30	0.13%
9	1	0.01%	25	0.10%	30	0.13%

Table 4.9: Distribution of No of cells in 1ms of M2610, M2611, M2612.

No of Cells	No of time slots (M2610)	Percentage (M2610)	No of time slots (M2611)	Percentage (M611)	No of time slots (M2612)	Percentage (M2612)
10	33	0.32%	96	0.40%	33	0.14%
11	32	0.31%	2271	9.48%	2283	9.87%
12	1065	10.18%	1384	5.78%	1358	5.87%
13	105	1%			1	0.004%
14	86	0.82%				
15	85	0.81%				
16	50	0.48%				
17	4	0.04%				

Table 4.9: Distribution of No of cells in 1ms of M2610, M2611, M2612.

4.4.2.4 Demonstration of the Problem from ACK Traffic of M2710, M2711 and M2712

The Acknowledgment Traffic at cell level M2710, M2711, M2712 yields an identical conclusion on deadlock. Figure 4.7 is a typical segment of acknowledgment cell interarrival times

Figure 4.7. A Segment of ACK Cell interarrival times Trace of M2710.

Group	Interval of Cell interarrival times (ms)	Number of Cell interarrival times	Percentage	Average (ms)
1	<i>[101.92, 225.86]</i>	940	67.72%	196.65
2	[14.05, 88.98]	69	4.97%	48.86
3	[1.14, 9.77]	16	1.15%	2.17
4	<i>[0.135, 0.886]</i>	360	25.94%	0.174
5	<i>[0.01, 0.02]</i>	3	0.22%	0.014

Table 4.10: Distribution of ACK cell interarrival times of M2710.

of test0, which is very similar to those of test 1 and 2. Tables 4.10 - 4.12 give the distribution when cell interarrival times of M2710, M2711 and M2712. Again, the deadlock is indicated by

Group	Interval of Cell interarrival times (ms)	Number Of Cell interarrival times	Percentage	Average (ms)
1	<i>[100.07, 238.24]</i>	550	33.78%	0.178818
2	[10.2, 90]	308	18.92%	0.055459
3	[2.02, 9]	29	1.78%	0.004033
4	<i>[0.135, 0.989]</i>	442	27.15%	0.000167
5	<i>[0.01, 0.032]</i>	298	18.30%	0.000012

Table 4.11: Distribution of ACK cell interarrival times of M2712.

Group	Interval of Cell interarrival times (ms)	Number of Cell interarrival times	Percentage	Average (ms)
1	<i>[100.89, 234.12]</i>	585	43.75%	188.23
2	[10.1, 96.1]	191	14.29%	53.96
3	[1.01, 9.61]	31	2.32%	5.20
4	<i>[0.135, 0.99]</i>	338	25.28%	0.183
5	<i>[0.01, 0.031]</i>	192	14.36%	0.012

Table 4.12: Distribution of ACK cell interarrival times of M2711.

the large ACK cell interarrival times in group 1, which concentrate in an area around 200ms (the cycle of generating delay-ACK) in Figure 4.7.

It is in some part of the traces to observe that, more than one ACK cells are clustered together and therefore introduce many small cell interarrival times (in group 4 and 5). The causes of these small ACK interarrival times can be explained by the operation of the delay-ACK mechanism

depicted in 3.2.3. The News-on-Demand application sets up three TCP connections for the three media streams, among which the ones transmitting the video and the audio data are easy to have deadlock (section 4.4.6 will mention about the text stream). When a deadlock occurs on these two TCP connections, the TF_DELACK flags for both sides are set. Every 200 ms, a timer process checks these flags for all connections. For each connection that has the TF_DELACK flag set, the timer routine changes it to TF_ACKNOW and the TCP output function is called to send an acknowledgment immediately. Therefore, if there are two connections experiencing deadlocks, two acknowledgments will be sent almost at the same time along the two TCP connections. At the cell level, therefore, more than one acknowledgment cell can be observed for each 200 ms. When sending the multimedia segments from the server, the application is designed to send the audio material first, with the video stream following after over 0.2 seconds. For two active TCP connections, two clustered ACK cells can be seen one for each connection. The amount of audio data is much less than that of video, so its transmission can be completed much earlier. When it is finished, only the video connection is left to suffer from the deadlock, thus for each deadlock the clustering ACK cells are replaced by a single ACK cell. For the whole communication sometimes only one TCP connection is presented, and sometimes more than one stream is active, so in Figure 4.7 the segments containing the clustered ACK cells are interleaved by the ones with one ACK cell per deadlock.

4.4.2.5 Comparison with Other Tests

For tests other than test 0, 1 and 2 (i.e., when SOCK_PAC_SIZE is 4096 bytes, 8192 bytes or 935 bytes), it is hard to perform statistical analysis on the cell level measurement results since over 50% of the information is lost. However, it is still reasonable to assume the initial segment of the traces are complete, because the cell record loss occurs only when the processing buffer of GN Nettest testset becomes full. Figure 4.8(a) shows the initial 300ms segment of the cell rate (number of arriving cells in each 1ms) of M2641, which is very similar to those of M2642, M2681(2) and M2691(2).

Compared to the initial segment of 300 ms of the cell rate of M2610 (the trace in Figure 4.8(b)), the one in Figure 4.8(a) looks more continuous. It seems that the application does not

work in the stop-and-wait manner anymore. From Table 4.13 we can find the reason of this difference. SOCK_PAC_SIZE of 4096 bytes, 8192 bytes and 935 bytes makes fully use of the

Figure 4.8 (a) Number of Coming Cells in 1ms of M2641 in first 300ms.

64K socket buffer. Although the packets sent are never large enough to trigger normal acknowledgments, the large socket buffer always allows two more packets that are as large as MSS to be sent before receiving the ACK for the previous packets. Those two large packet will

Figure 4.8 (b) Number of Coming Cells in 1 ms of M2610 in first 300 ms.

Step	write data	data in sender socket buffer (bytes)	No of unacked Bytes	Packet on the wire	data in receiver socket buffer (bytes)	ACK or not	Comment
1	write(4k)	4096	4096	>4096>	4096	No	too small to trigger a normal ACK
2	write(4k)	8192	4096		4096		keeps copying data to the socket buffer.
3	write(4k)	12288	4096		4096		
4	write(4k)	16384	13244	>9148>	13244	No	forms a large packet, the data of 13244 bytes are still too small to trigger an ACK
5	write(4k)	20480	13244		13244		
6	write(4k)	24576	22392	>9148>	22392	Yes	22392 is large enough to trigger a normal ACK
7			2184	>2184>	2184	No	send the left data in the socket buffer first
8	write(4k)	6280	2184		2184		
9	write(4k)	10376	2184		2184		
10	write(4k)	14472	11332	>9148>	11332	No	send a large packet
11	write(4k)	18568	11332		11332		
12	write(4k)	22664	20480	>9148>	20480	Yes	send another large packet. The accumulated data trigger an ACK
13			2184	>2184>		No	continue.....

Table 4.13: An Example of Interaction between the Server and the Client when SOCK_PAC_SIZE = 4096
 accumulate in the receiver buffer and invoke a normal ACK, therefore the deadlock does not happen and the transmission proceeds as fast as it can.

Figure 4.9 shows the ACK cell interarrival times trace of the M2741. The traces of the other 5 tests (M2742, M2781, M2782 and M2791, M2792) are quite similar to the one in Figure 4.9. There are 17 peaks (large cell interarrival times) appearing in the trace. They are possibly caused by the segmentation at the application layer. Since the server usually transmits 10-second material in less than 10 seconds, it needs to wait before starting to send the next segment of material. This makes the cell interarrival time of the first cell of one segment and the last cell of the previous segment extremely large (equalling the time the server had been waiting for).

Figure 4.9. The ACK Cell interarrival times of M2741.

4.4.2.6 Relationship to Previous Studies and Lessons Learned

The deadlock described above appears identical to what has been observed in [22, 23], but the causes of the deadlock are different. In [22, 23], the socket buffer size at sender and receiver site are set to be asymmetric intentionally. The deadlock usually occurs when the socket buffer at the receiver site is much larger than that at the sender site. In such a setup the receive buffer is so large that data received is not sufficient to invoke the sending of an ACK. At the same time, the sender buffer, waiting to purge its contents with the reception of an ACK, is too small to hold a packet of a MSS which would have been sent immediately as mandated by Nagle's algorithm. This

asymmetric buffer size setup was regarded as a major reason causing the deadlock and using the same buffer size at both the sender and receiver ends was suggested as an efficient way to avoid the deadlock[22, 23].

In our experiment, the socket buffer at both server and client site have been symmetrically set to the maximum value (64 kbytes) in order to avoid the problem discovered in [22, 23] and to attain the highest throughput that the network can support. However, the deadlock still occurs. Actually, the symmetric setup of the socket buffer size for both sites has been suggested in the Fore Card manual[38], and no practical application is reported to require the setup of asymmetric buffer sizes. Compared to what is demonstrated in [22, 23], the problem found in this thesis is due to the control mechanisms used by TCP and the packet size. It arises from an unreasonable value of `SOCK_PAC_SIZE` at the application layer set by the software designer who ignored the internal Mbuf structure of the operating system. Such a problem is hard detect as common bugs are, because the application may still work very well in a favorable network environment, such as a lightly-loaded network. Once the network bandwidth is shared by many applications, the application performance could be drastically affected by the parameter that must be carefully set. Network programming is different from common coding as it accesses network communication functions. In order to gain good performance of both application and network, designers have to consider the network throughput whenever selecting any parameters, and the program has to be tested on functionality as well as on network performance.

Also, our measurement results imply that performance problems may arise when hosts with various operating systems are internetworked by a TCP/IP/ATM network. As seen from the experiment, the network performance needs to be tuned by carefully selecting parameters at the application and the socket layer. If different operating systems are used in the application, the setup has to be done more cautiously as the Mbuf structure is different from operating system to operating system. For example, in SunOS 4.1, a plain MBUF can store data up to 112 bytes, while the clustered MBUF is as large as 1k bytes. Its data copying strategy encourages SunOS communication application programs to read and write in quantities of either less than or equal to 512 bytes, or slightly less than or equal to 1,024 bytes (or multiples thereof). A read or write of

less than or equal to 512 bytes will be placed in one to five plain Mbufs; and a read or write of less than or equal to 1,024 bytes (or multiple thereof) will make efficient use of the clusters that are used for writes larger than 512 bytes. If, in the News-on-Demand application, a server installed with IBM AIX needs to respond to a client with IBM AIX and a client with SunOS, the value of SOCK_PAC_SIZE must be selected thoughtfully to avoid the deadlock on both connections. Therefore, when various operating systems are involved in a communication application, network performance tune-up becomes a very complex job, which is not desirable for ATM internetworking. This may be one of the major weakness of running TCP/IP over ATM.

4.4.3 Traffic Burstiness Contributed by Application Layer

Sections 4.4.1 and 4.4.2 have demonstrated the traffic burstiness caused by the TCP layer (deadlock) and the ATM layer (network delay variation). In this section, the traffic burstiness introduced by the application layer will be studied.

Figure 4.10 Audio Traffic Defined in the Scenario.

The traffic burstiness contributed by the application layer arises mainly from two aspects: the application protocol and CPU sharing. The traffic burstiness comes from the application protocol (described in Chapter 2) is obvious, because the media segments are only sent at the predefined

time as determined in the scenario. Figures 4.10, 4.11 and 4.12 provide the data size of each

Figure 4.11 Video Traffic Defined in the Scenario.

media segment and the time when they are supposed to be delivered.

The other source of the traffic burstiness at the application layer is CPU sharing. On a multi-tasking computer, the CPU is seldom dedicated to a single process. Instead it is shared by all processes and CPU cycles are distributed among them according to some policy. Such a mechanism enables users to run a few of processes on one machine simultaneously and use the CPU more efficiently. Usually, this multitasking runs so smoothly that users seldom experience any inconve-

Figure 4.12 Text Traffic Defined in Scenario.

nience. It does, however, impact the traffic pattern at the application layer and maybe even at other lower layers. In our case, the CPU of the server workstation is shared by at least the processes of the three media servers. That means that at any time there can only be one active media server occupying the CPU. Furthermore, because the three media streams are supported by its own TCP connection, the TCP connections also work in an alternating manner.

Table 4.14 shows the operating order of the three servers at the beginning of test3 where SOCK_PAC_SIZE is 4096 bytes. Usually one segment of video or audio is transmitted in two or

Order	Media	Sequence Number of User Messages	Duration (sec)
1	audio	1-49	0.441763
2	video	1-65	0.518771
3	text	1 (1st text segment)	
4	video	66-93	0.185449
5	audio	50-53 (the end of 1st audio segment)	0.002114
6	video	94-167(the end of 1st video segment)	0.561304
7	text	2 (2nd text segment)	
8	audio	54-107 (the end of 2nd audio segment)	0.441210
9	video	168-336 (the end of 2nd video segment)	1.260135
10	text	3 (3rd text segment)	
11	audio	108-148	0.445028
12	video	337-352	0.037464
13	audio	149-161 (the end of 3rd audio segment)	0.217171
14	video	353-497 (the end of 3rd video segment)	1.203647

Table 4.14: Operating Order of Three Servers in Test 3

three sections. Each time the assigned CPU share is not long enough for a (video or audio) server to transmit a whole segment, the CPU is taken over by the other servers before the transmission (of the segment) can be finished. although the users may not feel it, CPU sharing causes the video or audio streams to be non-continuous at the application layer and the TCP layer. The traffic at the

cell level, however, remains continuous, because the three TCP streams are multiplexed on a single VP/VC at the ATM layer. Figure 4.13 depicts an example of a CPU shared by three servers.

Figure 4.13. CPU Sharing and Servers' Status.

CPU sharing has a great impact on the traffic pattern at the application layer and the TCP layer, but in our application, from a statistical point of view, it may not change the cell interarrival times very much. To see this, we must consider how the data is converted from one layer to another. TCP packets from the three media streams arrive at the ATM layer in the order determined by the CPU sharing. Since the multitasking nature of the CPU induces a latency when switching from one TCP stream to another, there are “gaps” between the various TCP packets when they are multiplexed at the ATM layer. When these packets are then fragmented into ATM cells, the “gaps” between the TCP packets result in large cell interarrival times. However, these large cell interarrival times occur relatively infrequently compared to those controlled by the PCR

at the ATM interface card. For example, assume that when the CPU switches from the audio server to the video server, a “gap” of $300\mu\text{s}$ is generated between an audio packet and a video packet (each of 9188 bytes). Then at the ATM layer, the packets are broken into a total of 385 ATM cells, producing 384 cell interarrival times, among which 383 are $84.8\mu\text{s}$ (defined by PCR at the ATM Interface card) and only one is $300\mu\text{s}$. Therefore the “gaps” between the TCP packets do not change the statistics of the cell interarrival times very much. On the other hand, if in our application, each of the three TCP connections is supported by its own VC, the influence of CPU sharing will be very obvious on the cell level traffic. The status of the VC (busy or idle) will be identical to that of the corresponding server and its TCP connection. When one server sleeps, its VC will also be idle. Thus the flow control mechanisms at UNI need to deal with the burstiness coming from traffic sources as well as from the CPU sharing, especially when they work on a per-VC basis.

Presently, it is hard to predict the way the processor switch from one process to another, as the CPU time distribution is very complex and difficult to follow and observe. In fact, the alternating operation of multiple processes may be random in real life, since the processes can start or end at any time. Even within the same network and with the same parameters at every protocol layer, the traffic at the application layer may still be different because of the random CPU allocation.

When SOCK_PAC_SIZE is 1024 bytes, the deadlock often forces the corresponding server to sleep and release the CPU so that it can be used by the other servers. Because the deadlock occurs at both of the video and audio connections, the switching between the two servers is much more frequent (over 50 times in one segment) than that in other tests (such as test3 listed in Table 4.14). The frequent switching introduces many short active periods interleaved by short sleep periods in each of the two streams. The user message interdeparture times and cell interarrival times, therefore, remain small (too small to be visible in Figure 4.14(a) and (b)). Sometimes the process switching does not happen so often, which means, the CPU is dedicated to one media server for quite a while, and then another media server fully utilizes the CPU for another period of time.

This causes both long active periods and sleep periods, and it results in large user message interdeparture times during the transmission of one segment.

Figure 4.14(a) A Segment of Audio User Message Interdeparture Time Trace of sa11.

The traffic burstiness contributed by the application layer can be seen from the audio traffic trace of sa11 and sa12 when SOCK_PAC_SIZE is 1024 bytes. Figure 4.14(a)(b) show two parts

Figure 4.14(b) A Segment of Audio User Message Interdeparture Time Trace of sa11.

of the sample path of the interdeparture time of audio messages of sa11. Figure 4.14(c) shows the

Figure 4.14(c). The Reflection of Nagle's Algorithm and Delay-ACK on sa11.

small user message interdeparture times in sa11 caused by deadlock. The trace of sa12 is similar to that of sa11. Table 4.15 and 4.16 include the distribution of audio message interdeparture times of sa11 and sa12.

Group	Interval (ms)	Number	Percentage	Average (ms)
<i>1</i>	<i>[1000, 7125]</i>	<i>35</i>	<i>0.94%</i>	<i>3383</i>
1	[107.6, 958.3]	264	7.08%	208.07
2	[12.51, 93.5]	32	0.86%	46.35
3	[1.01, 6.12]	78	2.09%	2.86
4	[0.24, 0.99]	3322	89.04%	0.336

Table 4.15: Distribution of Audio Message Interdeparture Times of sa11.

Group	Interval (ms)	Number	Percentage	Average (ms)
<i>1</i>	<i>[1011, 7236]</i>	<i>35</i>	<i>0.94%</i>	<i>3515</i>
1	[100.99, 776.44]	251	6.73%	201.58
2	[11.12, 95.78]	39	1.05%	37.62
3	[1.01, 8.65]	74	1.98%	3.28
4	[0.233, 0.996]	3332	89.31%	0.335

Table 4.16: Distribution of Audio Message interarrival times of sa12

Figures 4.14 (a) (b), Table 4.15 and 4.16 indicate that 35 user message interdeparture times are very large (above 1.0). The traces have proved that 17 of them are due to the segmentation at the application layer. The scenario determines that the audio data are fragmented into segments of 10-second and are sent segment by segment. It usually takes less than 10 seconds to transmit one segment from the application layer to the socket layer, hence, the interdeparture time between the first message of the segment and the last message of the previous segment will be particularly large.

As to the other 18 large user message interdeparture times that occur within some segments, it is clear from the trace that they coincide with the duration that the CPU is dedicated to transmission of the video stream. That is, except for several portions of each 10 second where alternation of the audio and video server occurs very rapidly, some large portions are dedicated to the video server. Large user message interdeparture times are introduced in the audio stream because its transmission is separated by that of video. However, this does not occur in the transmission of every segment, and for the latter segments, the time dedicated to video is longer. The reason for this CPU time distribution nature has not been found.

The influence of deadlock can be seen from the segment of trace shown in Figure 4.14(c). It provides a finer view of 4.14(a) and (b). The data can not be written into the socket buffer at a constantly high speed, because the server has often to wait for a delay-ACK to clean the socket buffer.

Figure 4.15 depicts a typical segment of video message interdeparture time of trace sv11. The influence of deadlock is indicated by the large message interdeparture time (around 200 ms) which appears very frequently. The distribution of the video message interdeparture time of sv11 and sv12 are presented in Table 4.17 and 4.18. Most of the message interdeparture times belong to the group 1 and 4, and those in group 1 are caused by the deadlock. The video segments contain so many data that their transmission almost use up the 10 seconds, therefore the video message interdeparture time do not reflect the segmentation at the application layer (that is, there is no very large message interdeparture time in the traces).

Group	Interval (ms)	Number	Percentage	Average (ms)
1	[1243.8, 2519.8]	937	8.34%	196.072
2	[10, 100]	0	0	
3	[1, 4.08]	103	0.92%	1.36
4	[0.238, 0.998]	10201	90.75%	0.346

Table 4.17: The Distribution of Video Message Interdeparture Time of sv11.

Group	Interval (ms)	Number	Percentage	Average (ms)
1	[1652.3, 2261.3]	936	8.33%	196.125
2	[15.4, 15.4]	1	0.001%	15.4
3	[1, 5.37]	104	0.93%	1.39
4	[0.233, 0.998]	10200	90.74%	0.346

Table 4.18: The Distribution of Video Message Interdeparture Time of sv11 and sv12

The audio message interdeparture times of trace sa41 are shown in Figure 4.16. Similar to the data shown in Figure 4.14(b), 17 peaks separate 18 audio segments, and there are some small peaks occurring within each segment. Again the traces prove that these small peaks are caused by the CPU time dedicated to the video stream. However, it is interesting that these small peaks are mostly the interdeparture times between the 40th and 41st message. It is hard to understand why

Figure 4.15. The Video Message interdeparture times of Trace sv11.

the small peaks appear at this location and why their value tend to increase. More investigation is needed over these two points in the future.

Figure 4.17 (sa81) and Figure 4.18 (sa91) also show the 17 peaks caused by segmentation at the application layer and the small peaks introduced by the CPU Sharing.

Figure 4.16. The Audio Message Interdeparture Time of Trace sa41.

Figure 4.17. The Audio Message Interdeparture Time of Trace sa81.

Figure 4.18. The Audio Message Interdeparture Time of Trace sa91.

The video message interdeparture times of trace sv41 are presented in Figure 4.19, and that of sv91 is shown in Figure 4.20. Again 17 peaks demonstrate the influence of segmentation at the application layer. Figures 4.21 and 4.22 show the segments of video message interdeparture time of sv81 and sv82 respectively. Although the setup of two tests is the same, the application traffic

Figure 4.19. The Video Message Interdeparture Time of sv41.

Figure 4.20. The Video Message interarrival times of Trace sv91.

are quite different. It seems that the CPU alternation occurred much more frequently in sv81 than in sv82.

In conclusion, this section presented two aspects of the traffic burstiness contributed by the application: the communication protocol at the application layer and the CPU sharing. The CPU sharing is a very dynamic factor and may increase the difficulty to describe the traffic at the ATM

Figure 4.21 The Video Message Interdeparture Time of Trace sv81.

layer and implement congestion control over it.

Figure 4.22. The Video Message Interdeparture Time of Trace sv82.

4.4.4 Relationship between Traffic at Different Protocol Layers

As shown in Figure 4.23, it is after going through several protocol modules that the traffic generated by application sources arrives at the application layer of the destination workstation. Each of these modules changes the behavior of the traffic, so the traffic is, eventually, the

Figure 4.23 Data Movement Path and Feedback in Traffic Pattern.

combined result of the multiple protocols and the original sources. Obviously, the traffic at low layer has been affected by the protocols of upper layers. At the same time, the lower layer protocol also has an effect on the traffic at upper layers. For example, if there is no room in the socket layer, the data copying process has to sleep and the communication at the application layer

is blocked. Thus the transmission at the application layer can not be carried out as predefined in the scenario. Additionally, TCP is a flow control mechanism based on feedback from the network and the destination, therefore, the ATM layer can also influence the traffic at the TCP layer. The interaction between the traffic at the different protocol layers has been shown in the measurement. In Figures 4.5-4.9, the influence of the application layer protocol and TCP layer can be seen from the traffic traces at the cell level, and in Figure 4.14(c), lower layer traffic is also reflected by the traffic at the application layer.

After going through so many modules, the traffic at the ATM layer is quite different from the source at the application layer. In most recent research studies, application sources are often regarded as real life traffic sources running directly over ATM networks. From the measurement, we can see that in a true network the application sources traffic may be quite different from traffic sent to the ATM layer. The influence coming from other protocol layers could have such a large effect over the traffic at the ATM layer that it should not be ignored in performance evaluation on network systems.

4.4.5 Positive Autocorrelation in Traces M2610,M2611 and M2612

**Figure 4.24. Autocorrelation of Multimedia Cell interarrival times of M2611:
Positive Peak at Lag 195.**

Figure 4.24 gives the autocorrelation of cell interarrival times of M2611 (for lags up to 210),

which is almost the same as that of the M2610 and M2612. It is obvious that autocorrelation is nearly zero except for a positive peak at the lag of 195.

Figure 4.25. Number of Cells between Two Cell interarrival times Peaks (M2610, M2611 and M2612).

As shown in Figure 4.6, we know that when SOCK_PAC_SIZE is of 1024 bytes, deadlock occurs frequently and can only be broken by the delay-ACK generated every 200 ms. Thus, the transmission works in a stop-and-wait manner and in cycles of 200ms. Thus it is reasonable to analyze the traffic in time slots of 200 ms. To detect the reason of the positive peak of autocorrelation, the number of cells transmitted in every 200ms needs to be determined. Due to the network delay, the interarrival time between the first cells of two consecutive time slots may not be exactly 200 ms anymore. However, from the analysis in 4.4.2.2, we know that deadlock causes the large cell interarrival times. Therefore, we used the large cell interarrival times as the indicator of the beginning of each time slot and counted the number of the incoming cells for each time slot. The counting result is presented in Figure 4.25. It shows that during the first 10 seconds ($50 \times 200ms$), the number of cells sent in a time slot remains at 518 for a while, then alternates between 323 and 195, while in the following 10 seconds, the initial sequence consists of 646 and 390 alternatively and then of 323 and 195 again. Analyzing the activity on the application layer of the M2611 (from the traces obtained from the measurement), we will determine how these pieces

of data are generated:

Time Slots (200ms each)	Media Stream	Write (bytes)	Send (bytes)	Clear (bytes)	No of generated Cells	Total No of Cells	Comments
1	audio	16k	1k		23	192	start with audio,
	audio		9140		192	215	deadlock happens after sending MSS size packet.
2	audio	1k	6220	10164	130	130	Only one stream is active. Delay-ACK triggered sending the left data before refilling. Due to OS delay, the socket buffer is not fully refilled (enough for 10k)
3	audio	15k	1k	6220	23	215	sent the remaining data firstly and then fill the buffer
	audio		9140		192		sent a large packet
4	audio	9k	6244	10164	131	538	
	audio		9140		192		
	video	16k	1k		23		
	video		9140		192		video stream joined;
5	audio	15k	76	15384	3	325	
	audio		9140		192		
	video	1k	6220	10164	130		OS delay
6	audio	9k	6220	9216	131	538	
	audio		9140		192		
	video	15k	1k	6220	23		
	video		9140		192		
7	audio	15k	76	15360	3	518	start enter into a cycle
	audio		9140		192		
	video	9k	6220	10164	131		

Table 4.19: The Activities at the Application Layer of M2611

Time Slots (200ms each)	Media Stream	Write (bytes)	Send (bytes)	Clear (bytes)	No of generated Cells	Total No of Cells	Comments
	video		9140		192		
8	audio	9k	6220	9216	131	518	
	audio		9140		192		
	video	15k	76	15360	3		
	video		9140		192		The cycle ends
9-18							The cycle described in 7 & 8 repeat from 9 till 18, that is the reason for a sequence of 518.
19	audio	1584	76	15360	3	360	The end of audio 1st segment
			1584		34		The end packet can be sent at once.
	video	9k	6220	10164	131		
	video		9140		192		
20	video	15k	76	15360	3	195	Only video stream is active, enter into another cycle
	video		9140		192		
21	video	9k	6220	10164	131	323	
	video		9140		192		End of the cycle
22-51							The cycle described in 20 & 21 repeat from 22 till 51. That is the reason of a sequence of 195 & 323.
52	audio	16k	9140		192	387	2nd audio segment begin
	video	15k	76	15360	3		
	video		9140		192		
53	audio	1k	7244	9140	152	475	OS delay
	video	9k	6220	10164	131		

Table 4.19: The Activities at the Application Layer of M2611

Time Slots (200ms each)	Media Stream	Write (bytes)	Send (bytes)	Clear (bytes)	No of generated Cells	Total No of Cells	Comments
	video		9140		192		
54	audio	15k	1k	7244	23	410	
	audio		9140		192		
	video	15k	76	15360	3		
	video		9140		192		
55	audio	9k	6220	10164	131	646	A Cycle starts
	audio		9140		192		
	video	9k	6220	10164	131		
	video		9140		192		
56	audio	15k	76	15360	3	390	
	audio		9140		192		
	video	15k	76	15360	3		
	video		9140		192		End of the cycle
57-68							The cycle described in 55 & 56 repeat from 57-68, that is the reason of the sequence of 646 & 390
69	audio	9k	6220	10164	131	646	
	audio		9140		192		
	video	9k	6220	10164	131		
	video		9140		192		
70	audio	7008	76	15360	3	345	End of 2nd audio segment
	audio		7008		147		
	video	15k	76	15360	3		
	video		9140		192		
71	video	9k	6220	10164	131	323	A cycle begins
	video		9140		192		
72	video	15k	76	15360	3	195	

Table 4.19: The Activities at the Application Layer of M2611

Time Slots (200ms each)	Media Stream	Write (bytes)	Send (bytes)	Clear (bytes)	No of generated Cells	Total No of Cells	Comments
	video		9140		192		End of the cycle
73 ~ end							The above cycle repeats till the end of trace

Table 4.19: The Activities at the Application Layer of M2611

As seen from Table 4.19, in most of the time slots, the deadlock forces both audio and video servers to send only 15360 bytes and 10164 bytes of data alternatively. If there is only one stream, the number of cells in each 200ms time slot are a sequence of 323 and 195. If both streams exist, the number of cells in every time slot could be a sequence of 518 or 646 and 390, depending on whether the two servers send the same amount of data in a time slot. Therefore, the major reason of the positive peak of autocorrelation of cell interarrival times is the deadlock occurring at the TCP layer.

4.4.6 Other Results

Figure 4.26. The Text Message Interdeparture Time.

Because the text segments are always very small, the change of SOCK_PAC_SIZE does not

have any effect on the text traffic at the application layer. In fact, deadlock does not occur on the TCP connection of the text stream even when `SOCK_PAC_SIZE` is set to 1024 bytes. Each text segment is so small that it can be carried in one TCP packet, so it is the first packet as well as the last packet of the segment. The TCP/IP implementation in BSD Unix 4.3 sets the PUSH bit of the last packet of one application data segment to 1 in order to force the receiver to generate a normal acknowledgment immediately, no matter how small the packet is. This push mechanism was designed to protect the application from deadlock, because the last packet of an application data segment is possibly too small to invoke a normal ACK. In Figure 4.26 the traces of the text message interdeparture time of the 8 tests overlap exactly on one sample path.

4.5 Summary

In this section, a brief summary is outlined for the overall measurement experiments. The following conclusion can be obtained from the analysis in the previous sections:

- The delay variation introduced by the network is too large to be ignored. Efficient congestion control mechanisms are needed on UNIs as well as on each network node.

The measurement has shown that the characteristics of the cell level traffic could be quite different from the one being controlled at UNI because of the delay variation introduced by the network. This not only impacts the traffic control to be used in the ATM network, but also affects real-time applications. ATM switches become an important source of delay variation. The buffers in ATM switches are designed to keep the cell loss probability as low as possible and to use bandwidth more efficiently, but at the same time, also introduce the variation of cell interarrival time. Successful congestion control in ATM network should be implemented at both UNIs and switches.

- The traffic in the network is the result of the interaction of the traffic control mechanisms on various protocol layers. Therefore, the traffic at each protocol layer is affected by the protocols at the other layers, and it can not be analyzed in an isolated manner.

A great deal of research has been done on traffic control schemes over a single protocol layer. In these research studies, the influence coming from other layers are seldom considered in order to simplify the problem. However, the measurement in this thesis has shown the tight relationship between the traffic at the different protocol layers. For example, the deadlock at the TCP layer impacts the traffic at the application layer and the ATM layer. Therefore it is necessary to ponder the interaction among various protocol layers when analyzing traffic in a network.

At the sender site, the data are moved from upper protocol layer to lower ones, so it is easy to understand that low-layer traffic is affected by the high-layer ones. At the same time, however, the data at high layers can only be moved to the lower layer when space is available, therefore when the buffers at the socket layer are full, the application data may have to wait. Thus, the high-layer traffic is also influenced by the traffic at lower layers.

Furthermore, TCP is a protocol based on the positive ACK. Such a feedback mechanism makes the traffic at the sender site related to and dependant on that at the receiver as well. If the data is not removed by the receiver in time, the data will accumulate at the receive socket buffer and no ACK can be generated, which will block the transmission.

The large influence of a protocol on traffic has been shown in the measurements. Before the data move to the ATM layer, the traffic has been modified to a pattern which might be quite different from that at application layer. This kind of influence is dependant on each protocol stack. As the ATM technology and its top protocols are all developing at a rapid pace, the ATM congestion control schemes should be evaluated in association with the top protocols on a continuous base. At the same time, a given network protocol stack may perform differently when running various applications due to their different QoS requirement. The present ATM network has no ability to support multiple QoS, therefore the performance verification for each application is especially important.

- Performance measurement is necessary to verify and test network applications.

The measurements have shown that the value of `SOCK_PAC_SIZE` has a great influence on the traffic behavior of three layers. When its value is not suitable, the network performance may be very poor. This problem may not be detected by the designers and the users in a nice network environment without performance measurement, and serious problems may arise when the application runs in a practical network.

Network programming is different from other application layer coding as it accesses many of the functions provided by lower protocol layers. As a result, parameters within these application must be set carefully as their values may have potentially large impact on the performance of the entire network. Therefore, knowledge of the network protocol stack will be beneficial to network programming. Furthermore, the program related to network communications should be tested not only on the function but also on network performance.

- The traffic burstiness may be due to traffic sources as well as transmission protocols.

From the measurements, it is obvious that each protocol may contribute to traffic burstiness. The burstiness is generated by segmentation and the CPU sharing at the application layer, deadlock on the TCP connections, and cell interarrival time variations introduced by the network. The burstiness can be observed at different time resolutions. For example, the burstiness generated by application material segmentation can be seen even though measurement is done in a large time interval, because the segmentation is based on 10 seconds. Burstiness caused by the CPU sharing mechanism, on the other hand, should be observed on a finer time interval. Deadlock at the TCP layer can be shown when the measurement is done on a time slot of 1ms. At the ATM layer, the burstiness contributed by the network, can only be observed by recording timestamps of each cell with high resolution. Therefore, a measurement over a large time interval can only reflect an average result of a traffic trace, and may not be fine enough to demonstrate traffic burstiness completely, while a measurement done on a fine time interval is able to present the information about burstiness at different levels.

In most simulations related to flow control in ATM networks, an application source model, such as MPEG video, could be used as a traffic source directly running over the network according to a specific application protocol, such as sending a frame of video in 1/30 sec to maintain real time delivery. Traffic burstiness is, thus, usually considered from the source. From measurement result in this experiment, we know that the application protocol and CPU sharing mechanism are two major sources of traffic burstiness at the application layer. The application protocol is different from application to application, so traffic burstiness resulting from this factor should be analyzed based on each application. A network and protocol suite which can support one kind of application successfully, is not guaranteed to support other kinds of applications very well. The performance needs to be verified by reasonable simulation or measurement.

- The CPU sharing is one of the sources of the traffic burstiness at the application layer

Normally, in a multitasking operating system, the CPU is shared by all running processes with the active process potentially being interrupted, put to sleep, and made active again at a later time. Therefore the communication processes are seldom continuous as most people assume in most simulations. They are forced to sleep now and then to let other processes to use the CPU. That is an important source of burstiness as the processes themselves are invoked in a burst fashion. Also the traffic burstiness caused by the CPU sharing may be random in a real system, it is an interesting and challenging problem for the future research.

Chapter 5

Modelling

5.1 Purpose

Many simulation models have been established in order to evaluate the performance of running TCP/IP over ATM networks. Most of them only describe basic TCP functions. Although gaining some valuable results, these models can not provide reliable results on the performance of a true TCP/IP/ATM network that is running a real application.

The models presented in this chapter attempt to describe a real system as accurate as possible. This is done by using measurement results at both the application layer and the ATM layer. As described in Chapter 3, in real systems it is through socket buffers that TCP is implemented, so the MBUF structure of an operating system is closely related to the network performance. In order to obtain convincing simulation results, it is necessary to include the buffer management strategy implemented in the IBM computer systems. The work in this chapter will make an effort on this point. The purpose of this chapter is to demonstrate how measurements and analysis of a network system can contribute to develop accurate models, and how simulation can help evaluate the systems that was studied.

5.2 Modules

Figure 5.1 depicts the OPNET model that describes the network system and a part of the application we used. It consists of six modules each simulating the functionality at the application layer, the socket, the TCP and the ATM layer of both the server and the client side. In this section, the functions and modeling fidelity will be described for each of these modules.

Figure 5.1 Modules Included in the Simulation Model.

5.2.1 Application Layer

In the real system, the application protocol and CPU sharing have a significant influence on the traffic at the application layer and even at the lower layers. However, it is hard to simulate everything at the application layer in detail. The application protocol can be divided into two parts: delivery schedule at the server site and the SSP at the client site. The simulation of the delivery schedule is very simple, while the influence from the SSP to the traffic is too complex to depict (see section 2.1.4). Also, the CPU sharing is difficult to describe in the model because of its dynamic nature. As shown in Chapter 4, the CPU is shared in such a random manner that it can be distributed in different ways even the application runs over the same network and with the same parameters set up at the each protocol layer. Fortunately, since in our experiment the packets of the three TCP connections are multiplexed on one single VC, from the statistical point of view, the CPU sharing mechanism has little influence over the cell interarrival time. Therefore, there should be little difference on the statistical result of cell interarrival time when CPU sharing is ignored. As a result, the cooperation between the three media streams will not be covered in our simulation, instead a simplified model is presented, in which only the audio stream is modelled with its TCP connection over an ATM virtual circuit.

The State Transmission Diagram of the simulators of the Audio Server and the Audio Displayer are shown in Figure 5.2 and Figure 5.3. In the Audio Server model, at the predefined time, an interrupt will occur so that an audio segment of the predefined size is generated and sent to the socket layer. The job of the audio displayer is to discard the data gained from the lower

layer.

Figure 5.2 State Transmission Diagram of the Audio Server Model.

Another important issue on the application layer is the data copying rate from the application layer to the socket layer. Once the application starts to send a segment of media data, the program enters a loop in which the data continue to be copied into the socket buffer in a user-defined unit called "SOCK_PAC_SIZE". There is no measurement determining how long it would take to finish such a data movement, that is, little is known about the time spent on copying a certain size of data from the application layer to the socket layer. Although our measurements keep a timestamp of each user message, not all of the user message interdeparture times can present the

Figure 5.3. State Transmission Diagram of the Audio Player Model.

time of copying the data. When the process of copying data is forced to sleep because of deadlock or CPU sharing, the user message interdeparture time includes the time of copying the data as well as the time that time of sleeping. In order to observe the data copy rate from our measurement result, we deleted the user message interdeparture times which are influenced by deadlock or CPU sharing. The remaining user message interdeparture times will be used as the sample of the data copying time, on which a Testool model is built. The data generated by the Testool model will be used in the simulation as the data copying time.

5.2.2 Socket and TCP Layer

The socket layer and TCP layer function is simulated in high fidelity as the major contribution of the model. In order to make the model as close to the implementation in the real system as possible, the module includes the socket MBUF management strategy given in Figure 3.7, 3.8 and 3.9, as well as the functions such as delay ACK, Nagle algorithm. The TCP time-out and fast retransmission are also components of the module. This module is independent of the TCP models available in OPNET. The TCP models provided by OPNET focus on the management on TCP connections, and similar to the TCP models that are usually used, little is related to a TCP implementation in a real system. Schemes like delay-ACK and socket buffer structure are not included in the models.

The State Transmission Diagram of the Socket and TCP layer at the server site are described in Figure 5.4. The jobs include the reaction to the data arriving from the upper and the lower layers. Once a media segment arrives, it is chopped into the user messages of "SOCK_PAC_SIZE". After checking the socket buffer space, a corresponding space will be assigned to the messages if it is available, otherwise the process will be blocked. The message will form TCP packets and be sent to the ATM layer. The other part of the module is designed to deal with the acknowledgment packets from the client. The time-out and fast-retransmission will detect the corrupted packet and send them again. The major functions in the module are listed as follows:

- initial: Initialization step. It will move to the "arrival" when some data arrive at the layer;

Figure 5.4 The State Transition Diagram of Model/TCP Layer at Server Site.

- **arrival:** The step deals with the incoming data. If the data come from the application layer (i.e., an audio segment arrives), the data will be forwarded to the “mesg_gen”, otherwise (i.e., an ACK packet comes), they will be forwarded to the “ack_response”.
- **mesg_gen:** The audio segments will be chopped in to the messages of the predefined SOCK_PAC_SIZE and then forwarded to “compare”;
- **compare:** it will calculate the socket buffer space that is needed to contain the audio message and compare it with the available space. If the remaining space is enough, the message will be forwarded to “copy_start”, otherwise, the program will enter an idle state and the communication will be blocked until more buffer space is released.
- **copy_start and copy:** these two states simulate the data copying from the application layer to the socket layer. The time that the process stays in the idle state between the “copy_start” and the “copy” is the data copying time gained from the measurements. When the data in the socket buffer can be sent as a TCP packet, they are forwarded to the “packet_gen”;
- **packet_gen:** TCP packets are formed in this module and sent to the “tcp_output” directly.
- **tcp_output:** each TCP packet is kept in a list in order to check ACK.;
- **send:** TCP packets will go to the ATM layer only when there is enough space on the ATM interface card. A timer is set for each outgoing TCP packet
- **ack_response:** It releases the socket buffer space once getting an ACK from the client. Also it implements the fast-retransmission function.
- **check_ack:** It is invoked by the timer set in the “send”. When the corresponding ACK is not received in the timeout, it will invoke the “re_send” module.
- **re_send:** It will retransmit the corrupted packets.

The Socket/TCP models at the client site is described in Figure 5.5. The major job is to deal with the packets from the server and generate the corresponding ACK. The ACK will be generated only if the packet is big enough or the PUSH bit is set or the delay-ACK timer is invoked. Each module can be described briefly as follows:

- **initial:** Initialization step. It will move to the “timer” automatically;
- **arrival:** It deals with the arrival of a packet. If the PUSG bit of the packet is set, or the data accumulated in the socket buffer is large enough, the data will be sent to “data_send”, otherwise, it will set TF_DELACK and the program will enter into an idle status and no ACK is generated.
- **data_send:** It transfers the data to the application layer.
- **ack_gen:** It generates an ACK. It can be invoked by the “data_send” or delay-ACK timer.
- **timer:** It generates an interrupt every 200ms, which will check the flag of TF_DELACK.
- **check:** It Checks the flag of delay-ACK.

Figure 5.5 The State Transmission Diagram of the Model of Socket/TCP Layer at Client Site.

5.2.3 ATM Layer

The ATM interface is simulated as a single queue. The maximum queue size is 256 kbytes. Since the Leaky Bucket used on the Fore Card has a token bucket of just one cell, the Leaky Bucket model can be simplified as a server with service rate equaling to PCR. Another function

simulated at the ATM layer are segmentation and reassembly. The model does not include the switches due to the lack of the information about their structure. Another difficulty on modelling is the delay variation introduced by the network. The presentation in Chapter 4 has shown that the network makes the traffic behavior quite different from that has been defined at the UNI. The measurement results we obtained are the cell interarrival times that contain the effect from upper protocol layer, such as the CPU sharing at the application layer and the deadlock at the TCP layer, so it is hard to detect which cell interarrival times are purely affected by network factors. However, the average of cell interarrival times which are smaller than $100\ \mu\text{s}$ happens to be $85\ \mu\text{s}$ (identical to the peak cell rate specified on the Fore Card). Therefore, we assume that cell interarrival times that are larger than $100\ \mu\text{s}$ have been effected by upper protocol layers, and the group of cell interarrival times which are smaller than $100\ \mu\text{s}$ are regarded as the cell interarrival times that are only determined by the network. These small cell interarrival times are isolated and their Testool model will be used to describe the influence from the network.

Figure 5.6 The State Transmission Diagram of the ATM Layer Model.

5.2.4 Fidelity Level

The fidelity levels of the model are listed in Table 5.1

Modules	Level of Fidelity
ATM Layer	High
ATM Switches	Low
IP Layer	Low
Link	Low
TCP Layer	High
Socket layer	High
Application	Medium

Table 5.1: Level of Model Fidelity

Note: High level fidelity means that it is simulated in detailed; Medium level fidelity means that only part of functions are simulated in detailed; Low level fidelity means that most functions are ignored in the simulation.

5.3 Comparison of Results of Modeling and Measurement

The comparison on statistics of cell interarrival times can be seen in Figures 5.7, 5.8 and 5.9. The simulation result matches quite well with that from the measurement. Figure 5.7 presents the Testool presentation on sample path, histogram, autocorrelation and spectral density, and the histogram and the autocorrelation are shown again in Figure 5.8 and 5.9 respectively. The simulation result, cell interarrival time with respect to simulation time is showed in Figure 5.10. Because deadlock occurs and forces the transmission to work in a stop-and-wait manner, the cell interarrival times around 0.2 seconds can be observed. One big cell interarrival time appears at the end of one segment since the transmission can be finished in less than 10 seconds.

5.4 Further Information Provided by Simulation Modelling

A simulation is done with SOCK_PAC_SIZE of 935 bytes. In measurement, the throughput is too high to be handled by GN Navtel Interwatch 95000. The traffic trace generated by the simulation model is showed in Figure 5.11. As the deadlock does not happen any more, the

Figure 5.7 Comparison of the Results of Modelling and Measurement.

Figure 5.8 Comparison of Modelling and Measurement on Histogram of Cell Interarrival Time.

Figure 5.9 Comparison of Simulation and Measurement on Autocorrelation.

network throughput is very high, and it only takes a very short time to transfer the whole segment to the client. The large cell interarrival times are introduced as the transmission time is much shorter than 10 seconds.

**Figure 5.10. Cell Interarrival Time with Respect to Simulation Time
when SOCK_PAC_SIZE = 1024 bytes.**

**Figure 5.11. Cell Interarrival Time with Respect to Simulation Time
when SOCK_PAC_SIZE = 935.**

Chapter 6

Conclusions and Recommendations for Further Study

6.1 Conclusions

The measurements in this thesis provided a chance to observe multimedia traffic at multiple protocol layers simultaneously. The analysis of the measurements illustrated some performance issues in a TCP/IP/ATM network.

The measurements have found the important contribution from the application layer to the overall traffic behavior. Two issues can be regarded as the most important aspects: application protocols and CPU sharing. An application layer protocol defines how the sender and the receiver should cooperate during the communication session. For example, it specifies the manner of sending the data, such as the unit in which the data are moved to the socket layer, the data are sent according to a predefined schedule or based on the acknowledgment from the receiver. Moreover, the application protocols designed to keep the required QoS for the application in order to implement the real-time delivery and to keep the synchronization of the various media streams. When the network has no ability to guarantee the required QoS, the application protocol is extremely important to keep the application working in a proper manner. Therefore, the application layer protocols have a tight relationship with the traffic at the application layer and even with that at the lower layers, and it can not be ignored in traffic analysis. At the same time, application protocols are different from application to application, so their contribution to the traffic should be

considered individually based on each application.

While multitasking operating systems enable processes to use CPU more efficiently, CPU sharing makes the traffic behavior more complex and difficult to characterize. When a communication process shares the CPU with many others in an alternating manner, the data transmission of the application may be interrupted by other processes. CPU sharing does not introduce inconvenience to the users, but it makes the traffic quite different from what people usually expect and assume in simulations. The traffic is more bursty as the data transmission itself works in a discontinuous manner that is controlled by the CPU scheduling policy. It is difficult to investigate the burstiness caused by the CPU sharing, because the status of processes is very dynamic, that is, a process could be created or terminated at any time.

Cell delay variation introduced by the network is another observation from the measurements. It implies that the traffic within the network might be quite different than the ones entering the network due to the large cell delay variation caused by the network. Congestion may still happen even after the traffic has been successfully controlled at the UNI. Therefore congestion control is needed at each node in an ATM network.

The tight relationship of the traffic at the different protocol layers is also shown by the measurement result. The traffic flow arrives at the application layer at the client site after going through several protocol layers, so the traffic finally is the combined result of the protocol layers. The measurements have demonstrated that the influence from a protocol layer can be seen on the traffic at the other layers. Therefore the influence from the other layers should be considered in traffic analysis. Also, in an ATM simulation, the effect from the upper protocol layers can not be ignored.

In general, the simulation results match the measurements, although certain parts of the model are built with a low fidelity due to the lack of the information. The simulation verified that performance tuning is required when running TCP/IP over ATM network, and

such a tuning is dependent on each operating system. This result is not supportive to TCP/IP/ATM internetworking between various operating systems.

6.2 Recommendations for Further Study

- Faster Operating ATM protocol analyzers can provide more reliable results. As the function of ATM protocol analyzers will be improved in the future, more advanced machines are expected to be used for further study.
- TCP_dump() and a third machine can provide detailed information about traffic trace at the TCP layer and make the measurement more complete. The development on the OCRInet ATM network testbed will provide chances to observe the traffic at the TCP layer.
- More measurement spots along the ATM connection can not only keep trace of the traffic, but also detect the source of cell delay variation and performance bottlenecks. With a higher speed interface card, the GN Navtel Interwatch 9500 can be moved to other locations along the loop, and the measurement can be done at various spots.
- The thesis has demonstrated the CPU sharing effect over the traffic at the application layer. It is interesting to investigate the effect of CPU sharing mechanism on the burstiness of traffic, such as burstiness length, the frequency of changing, CPU holding time etc.

The research also provides some suggestion to GN Navtel, the manufacturer of Interwatch 9500:

- As an ATM protocol analyzer, high operating speed is necessary to catch up with high rate traffic. Thus a “fast capture” option should be an essential function provided by the analyzer. In this experiment the data flow is at DS-3 (45 Mbit/s), which is not a high rate for ATM networks, but the traffic can still not be handled by GN Navtel Interwatch

9500. The development of a fast capture function is definitely needed for the research in the future, because ATM networks are intended to be operate at higher speeds.

- Limited amount of memory restricted the length of the trace that can be kept for one measurement. However, some traffic characteristics can not be observed unless the trace kept is long enough. Some improvement is needed to use the memory more efficiently. For example, the machine should enable users to select the information that will be kept in the memory according to their specific purpose. In our experiment, what we are interested in is only the timestamp of each cell. Since we have defined a specific VP/VC as the filter (only the cells with the required VP/VC will be captured), the information such as the VP and VC of the cell and priority of cell is not useful for this experiment. If the machine can provide an information filter, that would enable the memory to store only the information that the user is interested in. The limited memory space can then be used more efficiently and a longer trace can be maintained.
- Although the machine supplies detailed information regarding each captured cell, the format of information is not very user-friendly. The print-out of a text file that contains the information usually takes a long time and users do not have the chance to print out only the information that they are interested in. Also more programs are needed to extract the specific information. The machine could provide more information if it contains an information filter on print out function and some basic data processing packages to show the traffic behavior graphically.

Appendix A: The Scenario of Multimedia News

Tables A.1-A.3 provide the size and duration of each segment for audio, video and text material used in Multimedia News-on-Demand Application. [45]

File Name	Data Amount (bytes)	Duration (secs)
glob_a01	215600	9.8
glob_a02	220000	10
glob_a03	220000	10
glob_a04	220000	10
glob_a05	220000	10
glob_a06	220000	10
glob_a07	220000	10
glob_a08	220000	10
glob_a09	220000	10
glob_a10	220000	10
glob_a11	220000	10
glob_a12	220000	10
glob_a13	220000	10
glob_a14	220000	10
glob_a15	220000	10
glob_a16	220000	10
glob_a17	220000	10
glob_a18	82134	3.73

Table A.1 Audio Segments

File Name	Data Amount (bytes)	Duration (seconds)
glob_v01	682076	9.8
glob_v02	688344	10
glob_v03	655996	10
glob_v04	682704	10
glob_v05	657416	10
glob_v06	663632	10
glob_v07	626744	10
glob_v08	647412	10
glob_v09	675936	10
glob_v10	645872	10
glob_v11	687760	10
glob_v12	693344	10
glob_v13	639020	10
glob_v14	617540	10
glob_v15	670348	10
glob_v16	678508	10
glob_v17	642408	10
glob_v18	246384	3.73

Table A.2 Video Segments

File Name	Data Amount (bytes)	Duration (seconds)
glob_t01	143	7.27
glob_t02	132	8.66
glob_t03	97	8.73
glob_t04	71	8.06
glob_t05	77	3.78
glob_t06	138	15.61
glob_t07	89	4.48
glob_t08	119	6.03
glob_t09	133	8.77

Table A.3 Text Segments

File Name	Data Amount (bytes)	Duration (seconds)
glob_t10	125	5.57
glob_t11	105	5.88
glob_t12	116	7.24
glob_t13	150	8.02
glob_t14	33	3.35
glob_t15	127	7.09
glob_t16	138	6.59
glob_t17	138	8.98
glob_t18	52	3.05
glob_t19	102	11.09
glob_t20	118	8.52
glob_t20	137	8.01
glob_t20	126	6.22
glob_t20	158	9.47
glob_t24	37	2.87

Table A.3 Text Segments

Appendix B: Software Structure of Multimedia News-on-Demand Application

This section will focus on the delivery schedule and on the way that each software module cooperates to keep synchronization among the multimedia streams. The software implementation, which has little to do with the flow control function contributed by the application layer, will not be mentioned.

The application has been developed by using a client-server paradigm as shown in Figure B.1. The server comprises one MMDb (multimedia database) Server and three Media Servers. On the client station, there are the GUI (Graphical User Interface) initial process, the Scheduler, the Socket_MSC (Media Synchronization Controller)'s and FIFO (First In First Out) _MSC's (the later two together being the actual CMSC's) and three Media Players. Each of them corresponds to one medium stream.

As explained in 2.1.3, the presentational scenario is firstly transmitted from database to the scheduler upon a request from the client. According to the scenario, the scheduler at the client site determines which media types are involved in the scenario and creates the corresponding CMSCs (client media synchronization controllers). The scheduler is responsible of computing a delivery scheduler for each media object involved in the document. Then, it informs the involved SMSCs (server media synchronization controllers) and triggers the document delivery. Each SMSC then delivers the data objects according to the delivery schedule. A CMSC is responsible for receiving and transferring to the user interface the objects of a particular media type arriving on a stream connection. During the data delivery, the scheduler will not have any responsibilities other than coordinating the CMSCs and the SMSCs. When commands are issued by the user, such as PLAY, PAUSE and STOP, the scheduler receives an appropriate signal from the GUI and it

Figure B.1 The Architecture and the Data Flow in the Multimedia News-on-Demand Application.

Figure B.2. The Introduction of Synchronization Delay.

Since the media servers can be distributed over the ATM network, they may not all begin executing the delivery schedule at exactly the same time. The CMSCs handle the start-out mismatching for the application. A synchronization delay is introduced before the presentation schedule is applied in order to compensate for the mismatch. If, for a specific presentation, the first objects of three media streams are to be presented simultaneously, each CMSC will wait until all other CMSCs are ready with their own object before the presentation schedule is applied. In this case, the synchronization delay is the time elapsed between the arrival of the first object of any media and the arrival of the last object. This case is depicted in Figure B.2.

Scheduling and predicting traffic are not sufficient to maintain a simultaneous multi-stream delivery since the networks introduce random network disturbances. Random network delays destroy the continuity of the data stream by introducing gaps and jitters during the data transmission process. A stream synchronization protocol describes the recovery operations of the CMSCs. The concept of an intentional delay is used by the various streams in order to adjust their presentation time for recovery from network fluctuations. The common presentation schedule is shared by the concurrent CMSCs as depicted in Figure B.3. The CMSC processes are required to start and end their operations according to the shared schedule, so the multimedia news article will be played back in

synchronization.

Figure B.3. The Schedule is Shared by Three CMSCs.

The synchronization deviations that can be tolerated by human perception vary in different application scenarios. Skew tolerance parameters determine the synchronization goal that should be achieved by the CMSC processes. Different classes of synchronization QoS values are defined and stored in the presentational scenario. The CMSCs use the skew parameters to determine the mismatching tolerance time admitted by the application before applying synchronization error recovery. Late arrivals in at least one of the objects cause gaps in the display. The CMSC associated with the video files in the gaps by repeating the last frame received in the previous activity while the other CMSCs simply pause their display on the screen.

The CMSCs communicate only to indicate the first packet of an object that arrives on a stream. In other words, when the CMSC receives the first byte of an object, it informs the other CMSCs that this object is ready to start. The CMSC verifies whether parallel streams are ready and ensures that the skew or tolerance parameter has not been exceeded. Then it proceeds to display the objects. If the next immediate object to display has been delayed, the CMSC may have to apply restricted blocking in order to filter the gap. When network delay fluctuations occur, the idea of rescheduling the presentation within or at the end of an activity allows for the synchronization to be recovered without having to discard or skip late packets as long as the skew parameters have not been exceeded. The CMSC encountering delay makes the necessary changes to the scheduled presentation start times for the other streams, according to the synchronization requirements extracted from the

document scenario. An intentional delay is placed on the presentation of the related media objects, and the delays encountered during the presentation of an object are cumulative. While the intentional delay is in effect, each affected CMSC may have to handle the extra buffering demand and may have to extend the object display by an additional period of time.

When packets exceed skew tolerance parameters, the application is aborted only when the skew between related streams has exceeded a predefined upper threshold value. If the upper threshold value has not been reached and late arrivals have exceeded the skew, the late frames, in the example of a video stream, may have to be discarded in order to recover from synchronization errors. The difference between the value of the upper threshold and that of the skew tolerance parameter is an indication of the quality required for the presentation. Once the quality of the presentation is chosen, however, the upper threshold value is imposed to the whole system.

At the end, it is worth to mention a little about the interoperation of Socket_MSC and FIFO_MSC in CMSC. As the CMSC must read the incoming data on the socket opened to the SMSC and write these data at the scheduled time to the FIFO in order to pass the media objects to the display unit, the CMSC has been split into two parts: Socket_MSC, whose role is only to go and read data from the socket, and a FIFO_MSC whose role is to perform most of the functionality of the MSC----sending the media objects to the media player at the scheduled time if possible, performing the recovery algorithm if not, unless the problems encountered are too big and the application must be aborted. For each media type, the two components of the CMSC share a common buffer. This common buffer is a set of MEDIA_MAX_BUFF (varies only with the media type) buffer units. Each buffer unit can store up to MEDIA_UNIT_SIZE bytes, which is only a variable of the media type: TEXT_UNIT_SIZE, VIDEO_UNIT_SIZE, and AUDIO_UNIT_SIZE are defined to be 160 bytes, 5 kbytes, and 1 kbytes respectively. The structure of a buffer block for Media

Objects Data is shown in Figure 2.6.

Figure B.4. Structure of the Shared Buffer Used by Socket_MSC & FIFO_MSC

Each unit is associated with a number of bytes and a flag, which are used to coordinate the operation of Socket_MSC and FIFO_MSC. The data are written into and read from the buffer unit by unit.

- The flag tells whether the unit has been written or not. The Socket_MSC will write data in a unit only if this unit is not written already, i.e., `written_flag = FALSE`. Otherwise, it will wait until the unit is emptied by the FIFO_MSC. After writing in a unit, it will set the flag to `TRUE` if this unit is full, or write the same unit until it becomes full or when the end of the media object is reached. The FIFO_MSC will read the data from a unit only if the unit is written already, i.e., `written_flag = TRUE`. Otherwise (no data are in the buffer yet) it will wait for Socket_MSC to fill up the unit. After reading a unit, it will set the `written_flag` back to `FALSE`.
- The number of bytes indicates how many data have been written in the unit. So if the Socket_MSC does not fill up a unit, it knows how many bytes are to be written from the current unit into the FIFO when it next invokes the `read()` system call; there are generally `MEDIA_UNIT_SIZE` bytes except for the last bytes of an object which might not fill up a unit.

In conclusion, the application protocol implemented at the server site is as simple as defined in the prototype scenario, while the application protocol implemented at the client site is very complex. Synchronization delay is introduced to compensate different network delay experienced by each media stream, and at the same time this influence on the traffic is transferred to the lower protocol layers by the shared buffer structure.

Appendix C: Rate-based and Credit-based ATM Flow Control Schemes

In the rate-based view the network sends information to the user to specify the bit rate at which the user should be transmitting, and control loop may extend end-to-end across the network (as showed in Figure C.1). The Forward Explicit congestion Notification (FECN) and the Backward Explicit Congestion Notification(BECN) are two examples of rate-based schemes. In the FECN, A code-point in the header of ATM data cells, as a single bit-indicator of congestion in the forward direction of connection, the cells will be marked at the point of congestion and then forwarded to their destinations nodes as usual. At regular intervals, the destination would check whether EFCN was set in the most

Figure C.1. Closed-loop Flow Controlled Communication Network.

recently received data cell and send the feedback to resource. The transmitter will adapt the traffic flow according to the feedback it received. It also allows rate of a connection would adapt between a minimum and maximum value. BECN differs from the FECN in that congestion notification are sent directly back to the source by the switches experiencing congestion. A congested point in the network periodically would generate backward congestion messages, in the form of ATM Resource Management(RM) cells.

Figure C.2. Credit-based Flow Control Applied to Each Link of a VC.

In credit-based schemes, a separate window for each connection spans each link and takes the form of a credit allocation given by the receiver at one side of a link. Each credit received for a connection by the scheduler allowed it to transmit one cell. The receiver only allocated credits when it had a corresponding amount of free buffer space per VC, so that the allocated buffer size per VC constrains throughput just as a protocol window does. Figure C.2 indicates what happens in a credit-based flow control scheme. In particular, the maximum average throughput per connection was constrained by the ratio of its allocated buffer size to its round-trip time across the link. The hop-by-hop control limited the relevant round-trip time to the fixed round-trip propagation delay for a single link, thus minimizing it and avoiding the complications that would have arisen with the variable round-trip times for wider control loops. Still the schemes requires the precise

measurement of round-trip times across each link to minimize buffer sizes without constraining throughput.

There are two basic credit-based schemes which differ in how they allocate buffer space. The static credit scheme reserves enough buffer space to support a throughput equal to the link rate at all times. This is feasible for a LAN at current rates, but not for a WAN with hundreds or thousands of connections. The dynamic credit scheme addresses WAN as well as LAN by varying the buffer allocations depending on the number of active sources and their bandwidth demands. This requires a more complex protocol since decollating buffer space needs the receiver to take back credits that it already had allocated across the link.

If no sharing of buffers is allowed between virtual circuits, the amount of buffering needed can be prohibitive in a share network where hundreds or thousands of virtual circuits may share the same link. For example, in a gigabit link with a round-trip delay of 1ms, more than 125 kbytes of buffering would be needed per virtual circuit if the peak bandwidth of each virtual circuit is set to the link bandwidth. This problem is easily solved by allowing the virtual circuits to share the available buffers. This introduces a small probability of cell loss when many VCs are congested simultaneously.[32]

Appendix D: Function of IP over ATM Supported in Fore Systems Card

When an application makes a standard socket call in an attempt to transfer data to the destination IP address, on a machine which has a Fore Systems adapter card and driver software, one or more IP packets are passed to the driver software. The TCP will send a connect packet and wait for a response before proceeding, the ARP cache defines what ATM VPI/VCI should be used for a particular destination IP address. The IP handler checks to see if the IP address is already in the ARP cache, and if so, the IP packet can be transferred using the VPI/VCI in ARP cache. Otherwise the workstation generates an ARP request which in turn is broadcast to other switches. After receiving the ARP, the destination sends an ARP response indicating that it is at the ATM address corresponding to the IP address broadcasted. If no ARP response is received, then TCP will try again to make the connection. Because the ARP request identifies the sender, the response is sent only to source IP address.

The source device can now set up the ATM connection from the origin port to the destination port using the SETUP message. The switches route this SETUP message through the network and output the message to the destination port. The destination switch also assigns the VCC for this connection. While the ARP and SVC process progresses, the adapter buffers any additional IP packets generated by the application until address resolution and signaling completes so that transmission over ATM network can occur.

The destination device driver then responds with the CONNECT signaling message to communicate the fact that it has accepted the connection. Under abnormal circumstances, the destination would reply with a RELEASE which would cause the connection attempt to fail so that the source could inform the caller. Under normal circumstances, the switch routes the VCC through the network and delivers a CONNECT message to the source de-

vice driver on the signalling VCC.

Upon successful completion of the preceding steps, IP packets can now be forwarded by the IP handler over the VCC associated with the IP source-destination address pair in ARP cache. The forwarded IP packet is segmented by AAL function, with each cell assigned the VPI/VCI value as taken from ARP cache for the source-destination pair. By virtue of the ATM call setup process, the switch switches the cells to the destination switch and port. It also outputs the cells using the translated VPI/VCI value as established at call setup time. The destination AAL function reassembles the IP packet, which the IP handler passes to the destination IP address operating system software. Currently, if there is inactivity for a long period of time (many minutes), then that ARP cache entry is deleted. If a failure occurs, or a device is disconnected from a switch, then the SPANS informs the connection endpoints to remove the entry from their ARP caches.

Reference

- [1] P.P. Mishra, "Effect of Leaky Bucket policing on TCP over ATM performance", ICC'96, Dallas, June 24-27, 1996, pp.1700-1706
- [2] A. Erramilli and J. L. Wang, "Monitoring Packet Traffic Levels", IEEE GLOBECOM 1994, San Francisco, Nov 28-Dec 2, 1994 pp.274-280
- [3] W. E. Lelan, D. V. Wilson, "High Time-Resolution Measurement and analysis of LAN traffic: Implication for LAN Interconnection", IEEE INFORCOM 1991, Bal Harbour, April 9-11, 1991, pp.1360-1366
- [4] G. Y. Iazarou, V. S. Frost, J. B. Evans, D. Niehaus, "Using Measurement to Validate Simulation Models of TCP/IP over High-Speed ATM Wide Area Networks", ICC'96, Dallas, June 24-27, 1996, pp.309-314
- [5] J. Crowcroft, I. Wakeman, Z. Wang and D. Sirovica, "Is Layering Harmful?", IEEE Network, Jan 1992, Vol 6 No.1, pp.20-24
- [6] W. Stallings, "SNMP, SNMPv2, and CMIP, The Practical Guide to Network Management standards", Addison Westley Publishing Company, 1994
- [7] B. L. Tierney, W. E. Johnson, J. R. Lee, and G. Hoo, Lawrence Berkeley National Laboratory, "Performance Analysis in High-Speed Wide Area IP-over-ATM Networks: Top-to-bottom End-to-End Monitoring", IEEE Network, May/June 1996 Vol.10 No.3 pp.26-39
- [8] J.F. Shoch, and J.A. Hupp, "Measured Performance of an Ethernet Local Network", pp. 711-721, Communications of ACM, Vol 23, No.12, December 1980

- [9] D. R. Boggs , J.C.Mogul and C.A. Kent, "Measurement Capacity of an Ethernet: Myths and Reality", pp.222-234, Proceedings of SIGCOMM'88, Computer Communications Review, Vol 18, No. 4, August 1988
- [10] R. Gusella, "A Measurement Study of Diskless Workstation Traffic on an Ethernet", pp.1557-1568, IEEE Transactions on Communications, Vol 38, No.9, September 1990
- [11] D. Feldmeier, "Traffic Measurement of a Token Ring Network", pp.236-243, Proceeding of the IEEE Computer Network Symposium, Washington, D.C, November 1986
- [12] R. Jain, and S.A. Routhier, "Packet Trains: Measurements and a New Model for Computer Network Traffic", pp. 986-995, IEEE Journal on Selected Areas in Communications, Vol SAC-4, No 6, September 1986
- [13] C. Papadopoulos and G. M. Parulkar, "Experimental Evaluation of SUNOS IPC and TCP/IP Protocol Implementation", IEEE/ACM Transactions on Networking, Vol.1, No.2, April 1993
- [14] J. C. Mogul, "Observing TCP Dynamics in Real Networks", SIGCOMM'92, Computer Communication Review, Vol 22, No. 4, October 1992, pp.305-317
- [15] Y. F. Jou, F. Gong, D. Windelstein, N. Hillery, "A Method of Delay and Jitter Measurement in an ATM Network", ICC'96, Dallas, June 24-27, 1996, pp.328-332
- [16] D. Winkerstein, "Attila: Performance Analysis System for ATM Networks", IEEE GLOBECOM '94, San Francisco, Nov 28-Dec 2, pp.1394-1398
- [17] A. Romanow and S. Floyd, "Dynamics of TCP Traffic over ATM Networks", IEEE JSAC, Vol.13, No.4, May 1995, pp.633-641, See <http://ftp.ee.lbl.gov/floyd/epd.html>
- [18] L. Kalampoukas and A. Varma, "Performance of TCP over multi-hop ATM networks: A Comparative study of ATM-layer Congestion Control Schemes", ICC'95 Seattle,

June 18-22, 1995, pp.1472-1477.

- [19] H. Kung and R. Morris, "Credit based Flow Control for ATM Networks", IEEE Network, Vol 9, mar/Apr 1995, pp.40-48.
- [20] J. Postel, RFC 793, "Transmission Control Protocol", <gopher://ds.internic.net/00/std/std7.txt>
- [21] V. Jacobson, "Congestion avoidance and control", in Proceedings of ACM SIGCOMM, Stanford, California, Aug, 1988
- [22] D. E. Comer and J. C. Lin, "TCP Buffering and Performance Over an ATM Network", Journal of Internetworking: Research and Experience, Vol. 6(1), pp.1-13, March 1995.
- [23] K. Moldelklev and P. Gunningberg, "How a large ATM MTU causes deadlocks in TCP data transfers", IEEE ACM Transaction on Networking, August 1995 Vol 3, Number 4, pp.409-422.
- [24] R. Ahuja, S. Keshav, and H. Saran, "Design, Implementation, and Performance Measurement of a Native-Mode ATM Transport Layer (Extended Version)", IEEE/ACM Transactions on Networking, Vol 4, No. 4, August 1996. pp.502-515.
- [25] L. Lamont, L. Li, R. Brimont, and N. D. Georganas, "Synchronization of Multimedia Data for a Multimedia News-on-Demand Application", IEEE Journal on Selected Areas in Communication, Vol 14, No.1 January 1996, pp.264-278
- [26] L. Lamont, L. Li, R. Brimont, and N. D. Georganas, "Implementation of a Multimedia Scheduler Prototype for a News-on-Demand Service - The Documentations", provided by MCRlab, University of Ottawa
- [27] S. Floyd, "TCP and explicit congestion Notification", Computer Communication Review, Vol 24. Number 5. October, 1995 pp.8-22

- [28]C. Chang, R. Flower, J. Forecast, H. Gray, W. R. Hawe, K.K.Ramakrishnan, A.P.Nadkarni. U. N.Shikarpur, K. M. Wilde, "High-performance TCP/IP and UDP/IP Networking in DEC OSF/1 for Alpha AXP", Digital Technical Journal Vol.5 No.1, Winter 1993.
- [29]AIX Version 3.2 System Management Guide: Communications and Networks, "Tuning TCP/IP for High Bandwidth Interface", pp.4-98 ~ 4-106.
- [30]V.Jacobson, R.Braden and D.Borman. "TCP extensions for high performance", RFC 1323, 1992
- [31]F. Bonomi and K. W.Fendick, "The rate-based flow control framework for the Available Bit Rate ATM service", IEEE Network, March/April 1995 , pp.25-39.
- [32]H.T.Kung and R. Morris, " Credit-based flow control for ATM networks", IEEE Network, March/April 1995, pp.40-48.
- [33]<http://www.ocrinet.ca/>
- [34] email from Jim Yuan
- [35]AIX Network Performance Part1, <http://pscc.dfw.ibm.com/aixtra/issues/aixtra5/10steps.html>
- [36]Network Option command "no -a" result on CTR_C and CTR_S.
- [37]R.Braden, RFC 1122, "Requirements for Internet hosts - application and support" <http://ds.internic.net/rfc/rfc1122.txt>
- [38]ForeRunner MCA-200 ATM SBus Adapter User's Manual.
- [39]D. E. McSysan,,D. L. Spohn, "ATM Theory and Application", McGraw-Hill Inc, 1994.
- [40] GN Navtel interWATCH 95000 ATM/LAN/WAN Analyer User's Manual

[41] email from Yair Bourlas

[42] TCP_DUMP(), <http://www.yggdrasil.com/bible/LSM/lsm445.html>

[43] TEstool 1.0 User Guide and Reference Manual

[44] Unix Softwares: Tools Gnuplot Version 3.5, <http://www.uni-marburg.de/hrz/unix/graphik/gnuplot-3.5.html>

[45] email from Jeff Brinskelle

[46] <http://lara0.exp.edf.fr/aristote/dtc/dtc001a.html/>

IMAGE EVALUATION TEST TARGET (QA-3)

APPLIED IMAGE, Inc.
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved