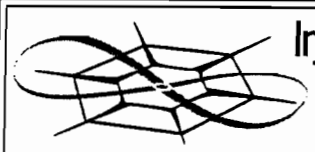
The University of Kansas



Information and Telecommunication Technology Center

AAI Performance and Congestion Management Studies: Year 3 Technical Report

Victor S. Frost, Professor, Electrical Engineering and Computer Science (EECS)
Joseph B. Evans, Associate Professor, EECS
Douglas Niehaus, Assistant Professor, EECS
David W. Petr, Associate Professor, EECS
Luiz A.P. DaSilva, Graduate Research Assistant
Roel J.T. Jonkman, Graduate Research Assistant
Georgios Y. Lazarou, Graduate Research Assistant
Soma Muppidi, Graduate Research Assistant
Beng Ong Lee, Graduate Research Assistant
C. Charalambous, Graduate Research Assistant

ITTC –FY98-TR-10980-27

January 1998

Sponsored by:
Sprint Corp.
under prime contract to
Advanced Research Projects Agency

Contract DABT63-94-C-0068

Table of Contents

List of Figures ii
List of Tables iii
Abstract v
1.0 Introduction 1
2.0 Wide Area ATM network Experiments Using Emulated Traffic Sources 4
2.1 Lessons Learned from AAI WAN Experiments: ATM WAN Performance 5
2.2 Lessons Learned from AAI WAN Experiments: ATM WAN Performance: Conducting National Scale Network Performance Measurements 7
2.3 References 9
3.0 A New Model and Performance Evaluation Methodology for ATM 11
3.1 ATM Traffic Model 11
3.2 Performance Analysis Methodology 12
3.3 Model Validation and Observations 14
3.4 Summary 16
3.5 References 16
4.0 Experimental Performance of TCP/IP Over High-Speed ATM over ACTS 18
4.1 System Configuration and Experimental Scenarios 18
4.2 Results of OC-12c Terrestrial/Satellite throughput Experiments 24
4.3 Discussion of Network Throughput and TCP Performance Pitfalls 25
4.4 Remaining Experiments Over High-Bandwidth-Delay-Product Networks 26
4.5 References 27
5.0 Overview of NetSpec 28
5.1 NetSpec Evolution 28
5.2 NetSpec Control Framework 30
5.3 Summary of NetSpec 37
5.4 References 38
6.0 Summary of 1997 AAI Traffic Flows 39
6.1 Introduction 39
6.2 Sites and Topology 39
6.3 Throughput Measurement 41
6.4 Significant Experiments 44
6.5 Initial Observations 46
6.6 NRL 47
6.7 ARL 49
6.8 NCCOSC 52
6.9 CEWES 54
6.10 NRLSSC 56
6.11 EDC 57

6.12 GSD 59
6.13 TIOC 59
6.14 KU 61
6.15 Conclusions 63
6.16 References 64
7.0 AAI Related Publications 65

Appendix A: "Wide Area ATM network Experiments Using Emulated Traffic Sources, " Beng-Ong Lee and Victor S. Frost

Appendix B: "Modeling and Analysis of Traffice in High Speed Networks," Soma S. Muppidi and Victor S. Frost

Appendix C. "Experiments and Simulations of TCP/IP over ATM Over a High Data Rate Satellite Channel," Charalambous P. Charalambos, Georgios Y. Lazarous, Victor S. Frost, Joseph Evans and Roelof Jonkman

Appendix D. NetSpec: Philosophy, Design and Implementation," Roelof J.T. Jonkman and Joseph B. Evans

List of Figures

Figure 4.1	Maximum Throughput Network Configuration	19
Figure 4.2	Congestion Free LAN and Wan Configurations	21
Figure 4.3	Stressed LAN and Wan Configurations	22
Figure 4.4	Stressed Terrestrial/Satellite Configuration	23
Figure 4.5	Aggregate Throughput Versus Offered Load in LAN, WAN, and ACTS	24
Figure 5.1	NetSpec Version 1 Architecture	28
Figure 5.2	NetSpec Version 2 Architecture	29
Figure 5.3	NetSpec 3.0 controller Architecture	30
Figure 5.4	NetSpec Block Language	32
Figure 5.5	Tree Structure of the Parsing of the Blocks in Figure 5.4	33
Figure 5.6	Phases of a network Connection	34
Figure 5.7	Command Syntax	35
Figure 6.1	Connections of the Sites and Switches for the Majority of 1997	41
Figure 6.2	SNMP Requests and Responses	42
Figure 6.3	AAI Network Data Transmitted by KU to TIOC Over an OC-12 Link for ACTS Tests	44
Figure 6.4	AAI Network Data TX by KU to TIOC Over an OC-12 Link	45
Figure 6.5	AAI Network Data Received from IOC to KU Over an OC-12 Link	46
Figure 6.6	Flow Patterns at Ports A and B Show Similar Characteristics	47
Figure 6.7	Connections to the FORE Switch aai-pop.nrl.aai.net at NRL	48
Figure 6.8	AAI Network Data Received by NRL During 1997	49
Figure 6.9	Connection to the FORE Switch aai-opo-ether.arl.aai.net at ARL	50
Figure 6.10	AAI Network Data Received by ARL During 1997	51
Figure 6.11	Connection to the FORE Switch aai-opo-ether.nccosc.aai.net at NCCOSC	52
Figure 6.12	AAI Network Data Received from nccosc.perf.ukans.aai.net During 1997	53
Figure 6.13	Connection to the FORE Switch aai-pop-ether.cewes.aai.net at CEWES	54
Figure 6.14	AAI Network Data Received by CEWES During 1997	55
Figure 6.15	Connection to the FORE Switch aai-pop-ether.nrlssc.aai.net at NRLSSC	56
Figure 6.16	Connection to the FORE Switch merlin.edc.magic.net at EDC	57
Figure 6.17	AAI Network Data Received by EDC During 1997	58
Figure 6.18	Connection to the FORE Switch aai-opo-ether.gsd.aai.net at GSD	59
Figure 6.19	Connection to the FORE Swithc hertz.tioc.magic.net at TIOC	60
Figure 6.20	HERTZ1 Traffic Flow Transmitted to KU During 1997	61
Figure 6.21	Connection to the FORE Switch spork.tisl.ukans.edu at KU	62
Figure 6.22	AAI Network Data Transmitted by KU During 1997	63

List of Tables

Table 2.1	Experiment Dates: KU-ITTC to AAI	8
Table 4.1	Host Configurations	18
Table 4.2	Results of Ten Maximum Throughout Experiments	20
Table 6.1	Switches at the Various Sites that are Monitored	40
Table 6.2	List of KU Performance Machines	43

Abstract

Wide-area Asynchronous Transfer Mode (ATM) networks are an evolving technology. New methods for performance measurements have been developed and applied in testbeds to obtain a better understanding of how advanced applications operate on high-speed wide-area networks (WANs). This report describes the development of network measurement tools, AAI network performance measurements, AAI network traffic measurements and analysis, and experiments with congestion control for the AAI.

1.0 Introduction

The ACTS ATM internetwork (AAI) is an early example of a high-speed national-scale ATM system. The thrust of the AAI effort at the University of Kansas (KU) is the measurement of AAI network performance and the use of those measurement capabilities to determine performance of the AAI. Our goal is to determine the performance of such a network under stress, as well as to profile the user applications, i.e., characterize the traffic. This research effort has yielded: 1) tools for measuring the performance of national-scale networks; 2) measurements of a national-scale Satellite/Trestresrial ATM network under unstressed and stressed conditions; 3) new mathematical models for traffic on such networks; 4) new theoretical techniques for predicting the delay and loss performance of ATM systems.

A three-year research agenda with the following tasks was mapped out to achieve the above goals.

KU's first task was measuring the performance of the AAI. Techniques to predict the performance of the AAI were developed to aid in explaining the operation of the network. The second task used AAI measurements to understand the traffic generated by various applications on the AAI. Evaluation of congestion control techniques for AAI was the third task.

During the first year of the AAI program the focus was on five areas: 1) initial network performance measurements and troubleshooting; 2) network measurement tools; 3) network traffic measurements; 4) AAI simulation development and validation; and 5) congestion control for AAI.

In the second year of the AAI program we conducted experiments on the AAI to gain an understanding of the factors affecting ATM WAN performance. Included in this endeavor was the evaluation of the peak performance of state-of-the-art workstations in firewall and router configurations. At the end of first year of the AAI, Sprint and KU collected traffic flows from SC'95. These flows were observed at 15 min. and 1 min. intervals respectively. The Sprint flows were observed on core network switches, while the KU measurements were obtained from the corresponding edge elements. These traffic flows were in the second year of the effort and analyzed in detail and compared to standard models. Also toward the end of first year of the AAI, KU put in place the automatic collection of traffic flows from AAI switches. A major goal of our AAI research is to measure the performance of the AAI under stress. To accomplish this goal a new network measurement tool, NetSpec, was developed and used in this effort. Of special note was the development of generic traffic models. These models allow the WAN to be loaded with controlled time-of-day dependent traffic flows. Such flows are representative of background traffic, and thus the impact of different levels of background traffic on target applications can now be experimentally evaluated. Also during the second year of the AAI effort, KU organized and hosted the 1996 DARPA

Workshop on Wide-Area ATM Performance in June. As a result of a successful proposal submitted by KU the results of the workshop were reported in a feature topic of the IEEE Communications Magazine in August 1997. Complete details of these activities can be found in [1.1].

Congestion studies were completed on the AAI. These studies evaluated the performance of the AAI under stress with the network possessing the following capabilities: 1) no congestion control, 2) open loop IP rate control, and 3) emulated ideal ABR closed loop control. The results and lessons learned from these experiments are presented in Section 2. The AAI provided the opportunity to observe ATM flows. A major objective of this research effort was to characterize the ATM traffic from a WAN and develop performance prediction methodologies. Traffic modeling and performance prediction in ATM systems is required for the design of efficient congestion control, routing and other network management algorithms. Here a non-Markovian phase process was developed to model the ATM WAN traffic flows. The phase process captures the macro-dynamic properties of the traffic. The traffic dynamics within each phase, i.e., the micro-dynamics, are described by a random process with finite mean and variance. A technique to predict the delay and loss performance given this traffic model has been developed and validated using AAI traffic flows. The traffic model and the corresponding performance prediction methodology is discussed in Section 3. AAI provided the opportunity to evaluate the performance of terrestrial/satellite ATM systems. Such practical experiments in a satellite network environment assist in the design and understanding of future global networks. Experiments, both simulation and AAI measurement based, were conducted to evaluated TCP/IP over ATM with a high speed satellite link. Performance comparison studies were conducted with the same host/traffic configurations over local area (LAN), wide area (WAN) networks, and a terrestrial/satellite system. It was found that the satellite systems deliver results similar to the terrestrial systems regardless of their path latencies in cases where the communication channels exhibit low bit error rates (BER). Network performance tests were carried out using application-level software (ttcp, Netspec) on OC-3 (155.54 Mbps) and OC-12 (622.08 Mbps) ATM satellite links. OC-12c experiments were limited due to ACTS HDR terminal hardware failures. Results from these experiments are presented in Section 4. A major contribution of this research was the development of NetSpec. NetSpec has been obtained by over one hundred and forty organizations for research and education purposes. This new WAN performance evaluation tool was the enabling technology for the results generated by this research. The current functionality of NetSpec will be summarized in Section 5. Throughout the past year the traffic flows in the AAI were continually monitored. Other than the flows generated by the KU performance studies and an occasional ftp experiment, there were no significant events on the AAI. A summary of these flows is provided in Section 6, and the complete set of AAI collected traffic data is included on the attached CDs.

References

[1.1] V.S. Frost, D.S. Petr, J.B. Evans, D. Niehaus "AAI Performance and Congestion Management Studies : Year 2 Technical Report", Telecommunications and Information Sciences Laboratory, Technical Report TISL-10980-21, January 1997.

2.0 Wide Area ATM Network Experiments

The idea of Asynchronous Transfer Mode (ATM) was first introduced in mid-1980s. Since then, numerous studies had been conducted to fully understand the performance and behavior of ATM wide area networks (WANs) and local area networks (LANs). As ATM standards evolve and its functionality expands and as advanced technology provides better ATM equipment and facilities at lower costs, new research issues in the area of network performance have arisen. Several national wide large scale ATM testbeds, such as the AAI, Multi-dimensional Applications Gigabit Internetworking Consortium (MAGIC) and Advanced Technology Demonstration Network (ATDnet), have been deployed to experimentally evaluate the ATM performance. One of the weaknesses of testbed networks is the lack of realistic traffic on the system. To experimentally evaluate networks, realistic traffic flows must be generated to emulate the actual traffic. Such source models are needed for network design and to evaluate traffic shaping, routing algorithm, and control mechanisms. Future broadband networks, especially wide area networks (WANs), will carry the traffic from commonly used applications, like telnet, file transfer protocol (ftp), and world wide web. Telnet is a widely common tool that allows a user at one site to establish a connection to a login server at another. Ftp is another tool that provides file access on remote machines. Traffic from video and voice services is expected to be a substantial portion of the network traffic. The exponential growth of the World Wide Web (WWW) will also have a great impact on the design and the performance of networks. Many ATM WAN experiments have been conducted on the testbeds to mea- sure point-to-point maximum throughput in bits/second [2.1-2.2]. These maximum throughput experiments do not address the network performance under cross traffic congestion scenarios. To truly understand ATM network performance from user prospective, realistic congestion experiments that include emulated user traffic must be conducted. In addition, user applications only see packet level performance, not the ATM level performance. Packet loss and packet delay jitter are the important elements in describing network performance. Here a summary of the empirically-derived analytic source models is described. Telnet and ftp connections, video streams, voice and the WWW traffic models are considered. The models were collected from various studies and notes [2.2], [2.3], [2.4], [2.5], [2.6], [2.7]. These source models have been successfully implemented in NetSpec, a network testing tool. The details of NetSpec can be found in Appendix D. Congestion experiments using these emulated user traffic models were designed and conducted on the AAI to evaluate network performance, such as packet loss and packet delay jitter on a stressed ATM WAN. In addition, effects of transport level flow control and ATM level traffic shaping were evaluated. Details of the results from these experiments can be found in Appendix A.

2.1 Lessons Learned From AAI WAN Experiments: ATM WAN Performance

This section presents the summaries of lessons learned from the AAI experiments. The lessons learned are categorized into: (1) ATM WAN performance; and (2) conducting national scale network performance measurements. Note that prior to the development of NetSpec experiments like the ones reported here were not possible. Thus, lessons learned from the experimental process are also important.

First consider the lessons learned related to the ATM WAN performance drawn from the experimental results of this study. TCP and UDP are the popular protocols being used on internet protocol (IP) networks. User applications only see packet performance, not ATM cell performance. So, it is vital to have acceptable packet level performance, especially in congested networks. The performance studies focused on measuring the throughput in b/s, IP level packet loss, and IP level received packet jitter. Here throughput measures the rate of reception of valid data and packet jitter measures the variation in packet arrival times at the receiver. The WAN experiments focused on evaluating the impact of 'background' flows on a target application. In this case the background flows were ftp, WWW, MPEG video, and teleconference video. The target flow generated a constant bit rate IP flow. Ideally the jitter of the target flow into the network was zero and the measured jitter at the receiver represented the variation caused by the network.

This study observed that the performance of packet traffic over ATM WAN in a congested and uncontrolled environment is poor. Delay jitter and packet losses are large and intolerable in non-cell level paced TCP and UDP tests. When the three multiple sources (two background traffic sources from ARL and NRL and the target flow from NCCOSC) merged together in the core network and these flows competed for the same destination, cells were dropped due to the peak rate burstiness, particularly from the bursty nature of background traffic sources (FTP and WWW). In ATM, a single 53-byte cell drop may cause an 9180-byte 1 [2.8] packet discard at the AAL5 layer due to an incomplete packet error. Network resources are wasted on delivering the useless cells belonging to the same corrupted IP packet. In TCP, retransmissions are triggered to recover the lost packets, but delays (both average and jitter) are increased and good TCP segments which followed the lost packet maybe unnecessarily retransmitted. In UDP, the discarded IP packets are unrecoverable at the receiving ends. As a result, large delay jitter and losses on packet level were observed for TCP. Traffic shaping techniques must be applied to alter the bursty IP traffic streams to have better control over network traffic. The TCP protocol may be used in conjunction with ATM traffic shaping to provide safe transmission of data.

In an ideal world, user traffic should have been shaped before submitting it to the ATM network. However we have learned that bursty traffic sources in an uncontrolled environment will adversely impact network performance. The results indicate poor network performance even though the aggregate average throughput in the worst case

scenario is only about 100 Mb/s over a 155.52 Mb/s OC-3 link. At first, cell level pacing was used on the target flows to evaluate its effects. The results of cell level paced target flows show that its effects are dependent on the direction of the flows and the network structure. In the case of the traffic flow from ITTC to the AAI cloud, better network performance was observed. Paced cells from the target flows were less likely to be dropped compared to the groups of back-to-back cells because of the large buffer and the OC-12 to OC-3 rate mismatch situation in the TIOC switch. However, target flows suffered more delay and losses when they went from the AAI cloud to ITTC. This is because the congestion point was in the AAI cloud. The congestion situation was identified as the traffic from three OC-3 links competing for the same output port which was also another OC-3 link in the core switch where they merged together. In addition, there were fewer available cell buffers in the network core as compared to the TIOC edge switch. As a result, paced cells from target flows were more likely to be discarded over a fixed period of time and worse packet performance was observed for the connections from the AAI cloud to ITTC. Adequate network performance was observed when cell level pacing was used on background traffic sources regardless of the direction of the flows. This time, the network only saw well-behaved traffic streams. The background traffic was not permitted to transmit at the full OC-3 bandwidth. Instead, the pacing rate was specified to demonstrate the potential effectiveness of ABR service. Bandwidth was reserved for the target flows. Although the target flows were still transmitted as groups of back-to-back cells, the small buffers in core switches were able to absorb the small amount of back-to-back cells and the bandwidth was available to the application. Given this situation, the network performance is substantially improved independent of the direction of the flows. The simplest traffic shaping technique, peak rate cell level pacing, was found to be an effective way to avoid severe network congestion. Note the AAI provided the opportunity to demonstrate this over a multi-host national scale ATM WAN. Previous network measurements, e.g., from the MAGIC network, focused on the evaluation of point-to-point cell level pacing.

Tests were also conducted from ITTC to the AAI cloud, and the AAI cloud to ITTC separately. Different ATM network performance was in general observed depending on direction of the traffic flows. This type of asymmetric performance was caused by the asymmetrical structure of the network in connection-oriented WANs. Note this will be a common occurrence in future networks. In the tests with the traffic flow from ITTC to the AAI cloud the congestion point was the OC-12 to OC-3 mismatch in the TIOC switch. However, when the direction of all the traffic flow was reversed, this created a congestion point within the AAI. As a result, different network performance was obtained. In designing congestion experiments and evaluating network performance, the direction of traffic flow must be taken into consideration. If not, underestimations and inaccurate conclusions might be drawn from the network experiments.

The main functionality of Early Packet Discard (EPD) is to relieve network stress by reserving queue buffer space in switches. Note EPD is another form of open loop congestion control. As the AAI core network evolved, we had the opportunity to study EPD performance. The comparison of the network delay and loss after network upgrades, including EPD, shows a substantial gain in network performance, and traffic congestion was reduced in AAI backbone once the EPD was enabled. Although bursty background traffic sources were observed to have more cells being dropped, mission-critical target flows had better performance. These improvements are attributed to the EPD functionality which in this case provides better overall network performance. Here the EPD discarded cells from the bursty background traffic leaving room for the traffic from the target flow.

2.2 Lessons Learned From AAI WAN Experiments: ATM WAN Performance: Conducting National Scale Network Performance Measurements

Although results from network performance measurements are beginning to appear in the literature, experiments with national scale high speed networks are rare. Thus, the experience base is limited for network researchers. Network investigators usually gain experience by doing experiments and by trial and error. We have learned that without careful preparation, this approach often leads to wasted human, host and network resources. This section addresses some of the observations made and issues found in this study concerning the process of conducting large national-scale network experiments.

Maintaining network connectivity, especially in a WAN testbed environment, has been a difficult task. Note that many widely geographically distributed elements must work perfectly for the experiment to be successful. In the case of AAI network, we have struggled to set up and maintain stable network connections to other sites. The core network was very stable; however the IP over ATM configuration sometimes failed. PVCs were set up to interconnect the four experiment sites (ARL, NRL, NCCOSC, and ITTC). Smart (soft) permanent virtual connections (SPVCs) were used to set up connections though multiple network switches. This was the only available mechanism for implementing cell level pacing. The SPVCs with specific pacing rates were set up before and torn down after each experiment. In addition, the workstations deployed by KU to the other AAI sites were not stable because they used beta versions of the operating systems. These early releases of the operating systems were required to obtain access to advance ATM features. Thus the geographically distributed workstations needed to be rebooted occasionally. This required coordination with the other sites. Excellent coordination was provided by the sites; however, it usually caused delays in performing the experiments. In doing the experiments, all the workstations needed to be operational and reachable through the AAI. The AAI backbone network was very reliable and stable, however the experimental nature of the ATM/IP routing software sometimes caused connectivity problems. Thus, the problem of network connectivity partly contributed to the delay in running the experiments.

Host state also plays an important role in doing network measurements. Note that in the cell level paced background traffic tests the standard deviation of UDP packet was about 3 ms higher than in the other tests. See Appendix A for details. This was because some statistic capturing scripts were running on this workstation. NetSpec is a running process at the application level. As a result, it competes with other processes for CPU resources. Thus, in this case, the results of the experiment were impacted. To accurately measure network performances using workstations, host machines must have low system load.

Table 2.1 and 2.2 are the run dates of experiments for ITTC to the AAI and the AAI to ITTC, respectively. One set of experiments took about 3 hours to complete. In other words, more than 150 hours of successful network experiment time, including the 24-hour validation experiments, were logged in this study. One of the obstacles found when conducting this study was that, as expected, congestion was induced into the network. The experiments were purposely designed to cause severe congestion in the network in order to evaluate network performance. Tests were postponed several times to evaluate the impact on the production network in the beginning of this study. Toward the end of the study, the tests were conducted on night time and weekend basis to avoid cross traffic and impacts on the production network.

A total of 772 successful national scale network experiments were conducted as part of this effort. These experiments generated over 1 Gbytes of raw data. Each packet in each test was time-stamped and recorded. Significant effort was required to create algorithms to post-process the data, and then apply these algorithms to the raw information. Appendix A contains the results of the analysis of this data.

5/22/97	UDP Target Flows UDP Target Flows with 25Mbps Background Traffic UDP Target Flows with 60Mbps Background Traffic
6/4/97	TCP Target Flows alone TCP Target Flows with 25Mbps Background Traffic TCP Target Flows with 60Mbps Background Traffic
6/5/97	Cell Level Paced UDP Target Flows
6/18/97	Cell Level Paced UDP Target Flows with 25Mbps Background Traffic Cell Level Paced UDP Target Flows with 60Mbps Background Traffic
6/22/97	Cell Level Paced TCP Target Flows
6/23/97	Cell Level Paced TCP Target Flows with 25Mbps Background Traffic Cell Level Paced TCP Target Flows with 60Mbps Background Traffic
7/3/97	TCP Target Flows with Cell Level Paced 25Mbps Background Traffic
7/4/97	UDP Target Flows with Cell Level Paced 25Mbps Background Traffic
7/9/97	TCP Target Flows with Cell Level Paced 60Mbps Background Traffic (Part I)
7/10/97	TCP Target Flows with Cell Level Paced 60Mbps Background Traffic (Part II) UDP Target Flows with Cell Level Paced 60Mbps Background Traffic

Table 2.1 Run Dates of the Experiments from ITTC to AAI Cloud

7/19/97	UDP Target Flows UDP Target Flows with 25Mbps Background Traffic UDP Target Flows with 60Mbps Background Traffic
7/20/97	UDP Target Flows alone Cell Level Paced UDP Target Flows alone
7/26/97	TCP Target Flows with 25Mbps Background Traffic TCP Target Flows with 60Mbps Background Traffic
8/4/97	UDP Target Flows with 25Mbps Background Traffic Cell Level Paced UDP Target Flows with 25Mbps Background Traffic
8/9/97	TCP Target Flows with Cell Level Paced 25Mbps Background Traffic UDP Target Flows with Cell Level Paced 25Mbps Background Traffic
8/10/97	Cell Level Paced UDP Target Flows alone TCP Target Flows with Cell Level Paced 60Mbps Background Traffic UDP Target Flows with Cell Level Paced 60Mbps Background Traffic
8/21/97	Cell Level Paced TCP Target Flows with 25Mbps Background Traffic Cell Level Paced TCP Target Flows with 60Mbps Background Traffic
8/21/97	Cell Level Paced UDP Target Flows with 60Mbps Background Traffic UDP Target Flows with Cell Level Paced 60Mbps Background Traffic

Table 2.2 Experiments from AAI Cloud to ITTC

In summary, this work demonstrated that stressed ATM WANs can be used to support high performance application with simple traffic controls, e.g., cell pacing (as would be used in ABR) or EPD.

2.3. References

[2.1] V.S. Frost, D.S. Petr, J.B. Evans, D. Niehaus "AAI Performance and Congestion Management Studies : Year 1 Technical Report", Telecommunications and Information Sciences Laboratory, TISL-10980-11, January 1996.

[2.2] V. Paxson, "Empirically Derived Analytic Models of Wide-Area TCP Connections," IEEE/ACM Transactions on Networking, VOL. 2, NO. 4, August 1994.

[2.3] R. Caceres, P.B. Danzig, S. Jamin, and D.J. Mitzel, "Characteristics of Wide-Area TCP/IP Conversations," Computer Science Department, University of Sounthern California, Los Angeles, California.

[2.4] N.M. Mara_h, Y. Zhang, and R.L. Pickholtz, "Modeling and Queueing Analysis of Variable-Bit-Rate Coded Video Sources in ATM Networks, " IEEE Transactions on Circuits and Systems for Video Technology, VOL. 4, NO. 2, April 1994.

[2.5] D. P. Heyman and T. V. Lakshman, "Source Models for VBR Broadcast Video Traffic," IEEE/ACM Transactions on Networking, VOL. 4, NO. 1, February 1996.

[2.6] M. Krunz, and J. Hughes, "A Tra_c Mopdel For MPEG-Coded VBR Streams," Department of Electrical Engineering, Department of Computer Science, Michigan State University.

[2.7] M. F. Arlitt and C. L. Williamson, "Web Server Workload Characterization: The Search for Invariants," Department of Computer Science, University of Saskatchewan, 1996.

[2.8] R.Atkinson, "RFC Default IP MTU for use over ATM AAL5," RFC 1626, Naval Research Laboratory, May 1994.

3.0 A New Model and Performance Evaluation Methodology for ATM

Broadband networks of the future will carry audio, video and data traffic from many diverse applications. Such networks will need to meet a variety of traffic and performance requirements. Asynchronous Transfer Mode (ATM) has been chosen as the technology to implement B-ISDN. Details about ATM will not be discussed here but can be found in [3.1]. ATM offers many advantages like bandwidth on demand and Quality of Service (QoS). The integration of traffic from many different types of applications is expected to generate traffic having complex characteristics. In particular, broadband traffic has a complex correlation structure that often spans a wide range of time-scales. Such long-range dependence is not taken into consideration in traditional Markovian models. Thus, conventional traffic models yield system performance predictions that are significantly different from what would actually be obtained in a real environment. Several recent traffic studies [3.2], [3.3], [3.4], have reported that the correlation in the traffic arrival process has a significant impact on the network resource management and network performance evaluation. Particularly, it has been observed that the long range dependence of the traffic process dominates queueing performance. In the context of network traffic, long-range dependence implies that even if the number of packet arrivals is aggregated over larger and larger time-scales, the aggregated process will not smooth out as is expected with finite variance processes like Poisson process or short-range memory processes like Markov-Modulated Poisson Processes. In this work a phase type arrival rate process is used to model ATM wide area network (WAN) traffic. This phase process is non-Markovian and governed by a distribution with infinite-variance. In each state a point process is assumed. The infinite variance phase process controls the large-scale or macro-dynamics of the traffic, while the point process captures the small-scale, micro-dynamics of the flow. In this study we showed that such a model can be used to accurately predict the delay and loss performance of ATM WAN traffic flows. Further this work demonstrates that the queueing delay performance is not sensitive to the specific nature of the micro-dynamics. However, these results indicate that the cell loss probability is affected by the nature of the small scale variations. Delay and loss performance is predicted and compared to trace driven simulations. These traces were collected from the ACTS ATM Internetwork (AAI). Details of the methodology and corresponding validation results are presented in Appendix B.

3.1 ATM Traffic Model

In this section the analytical traffic model is presented. Here the arrival rate process is modeled as being modulated by a phase process. Assume that the phase-process is stationary and ergodic having a finite state space $\mathbf{S}=\{x_1, x_2, \ldots x_N\}$. This phase-process captures the macro-dynamic properties of the rate process. Within each state $x_i \in S$, the arrival process is assumed to follow a point process with a distribution with a finite mean and variance. The distribution function of the rate process associated with each state defines the traffic micro-dynamics of that state. The steady state phase

probabilities of the phase process are assumed to follow a distribution with a heavy tail, i.e., infinite variance. The variables which characterize the process considered here are:

1. The steady state probability vector $\bar{\pi} = [\pi_1, \pi_2, \ldots, \pi_N]$ where $\pi_i = P(S = x_i)$.

2. The rate vector $\bar{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_{N+1}]$ where $\gamma_1 \leq \gamma_2 \leq \ldots \leq \gamma_{N+1}$ represent the boundary rates (in cells/sec) for each state $x_1, x_2, \ldots x_N$.

Here we assume that the mean sojourn time in each state of the phase process is proportional to the steady state probability of being in that state and that within each state x_i the arrival process is assumed to have *Cramer- type* [3.5] characteristics, for example, exponential, uniform etc., with mean arrival rate equal to γ_{i+1}. Note that γ_{i+1} is the upper bound on the rates for state x_i and is thus a conservative assumption for performance modeling. The state probabilities themselves are obtained from a *non-Cramer* [3.5], [3.6] type distribution having infinite variance.

In this work a Pareto distribution is the infinite variance distribution used, however, any infinite variance process may be assumed to obtain the state probabilities. In particular rate histograms estimated from measured network trace data have also been used in this work to determine the appropriate state probabilities. It should be noted that in [3.2], the authors report from empirical findings the usefulness of Pareto Distribution in modeling collected network data that is inherently long range dependent in nature. The performance predictions from trace based simulations are compared to those obtained using both the Pareto and histogram (measurement based) derived state probabilities.

3.2 Performance Analysis Methodology

An analysis technique is now developed based on the above model. The performance metrics of interest here are mean cell delays and cell loss ratio. Let Z denote a random variable associated with a performance parameter of the queue, having *R(t)* as the input process. Here Z is delay or loss probability. The objective of the performance analysis technique is to characterize the random variable Z in terms of the properties of *R(t)*. In each of the states the arrival process is assumed to follow a fixed *Cramer - type* distribution with a given mean arrival rate γ_{i+1} for state x_i i.e.,

$$E[R(t) \mid i^{th} state] = \gamma_{i+1} \tag{9}$$

Within any given state of the input process the queueing system behaves as a slotted-G/ D/ 1 queue with mean arrival rate given by Equation (9). Given the linearity

property of expected value, the expected value of the random variable Z can be written as

$$E[Z] = \sum_{i \in S} \pi_i E[Z \mid S = x_i] \tag{10}$$

Note that Equation (10) represents the effect of traffic macro-dynamics as well as the traffic-micro dynamics. The traffic macro-dynamics are described by the values of $\bar{\pi}$ and the micro-dynamics are represented by $E[Z \mid S = x_i]$. The macro-scale dynamics capture the burstiness in the input trace. The effect of micro-dynamics will be considered. Fundamentally, the average delay and cell loss given that the phase process is in state i needs to be found for a time slotted system with deterministic service. From [3.7] the average delay is given as

$$E[D \mid S = x_i] = \frac{1}{\lambda} \frac{1}{1-\rho_i} \{\frac{1}{2}\sigma_i^2 + \frac{3}{2}\rho_i^2\} \tag{11}$$

where

D is delay.

σ_i^2 is the variance of the arrival process, given S=X_i.

$1-\rho_i$ = probability that the system is empty given S=X_i and

$\lambda = \sum_{i=1}^{N} \pi_i \gamma_{i+1}$ is the mean arrival rate.

Thus the expected value of the delay is given by

$$E[D] = \frac{1}{\lambda} \sum_{i=1}^{N} \frac{\pi_i}{1-\rho_i} \{\frac{1}{2}\sigma_i^2 + \frac{3}{2}\rho_i^2\} \tag{12}$$

Note the above expression for delay is valid for any point process model for the micro-dynamics. Further, the above result shows that the average delay is not dependent upon the specific distribution of the micro-dynamics, as long as the moments are identical. In addition we can use the above expression to calculate the average delay for different cases of the micro-dynamics. Next an analytical expression for the estimating the cell loss ratio is presented. This analysis is based on [3.7]. Unlike the case for estimating the mean cell delays, the calculation of cell loss in a deterministic server queueing system is considerably more complex. The cell loss ratio in a finite buffer queueing system is approximated by the tail of the queue length distribution of the infinite buffer counterpart. Here we assume that the number of arrivals to the queueing system in a slot follows a Poisson process. Note that this theoretically limits the cell loss prediction methodology to Poisson Micro-dynamics. The practical implications of this assumption will be discussed later.

Exact expressions for cell loss in terms of the ergodic occupancy distribution of an M/ D/ 1 system are now presented. The cell loss estimate for the arrival process is then obtained as sum of loss estimates obtained from individual states weighted by the probability of being in that state. The expressions for the M/ D/ 1 queue are obtained as a special case of the M/G/ 1 analysis by using a recursive numerical technique due to Kielson and Servi and given in [3.7]. Let P_k^i denote the probability that there are k cells in the queue given that the input process is in state x_i i.e., $P_k^i = P[Q = k \mid S = x_i]$. Here the cell loss probability estimate given that the input process is in state x_i, is approximated as the probability $P[Q > K \mid S = x_i] = P_L[K \mid S = x_i]$, for a given system size of 'K'. Let the utilization of the system given that the input process is in state x_i, be ρ_i. The utilization of a queueing system is defined as the ratio of the mean arrival rate to the queueing system to the service rate. For stability we require $0 \le \rho_i < 1, \forall i$. Note that $P_0^i = 1 - \rho_i$ given in the previous section, $P_k^i, P_L[K \mid S = x_i]$ are given below as:

$$P_k^i = \frac{(1-\rho_i)p_k^i + \sum_{n=1}^{k}\{1-\sum_{m=0}^{n} p_m^i\}P_{k-n}^i}{p_0^i} \qquad k = 1,2,3,\ldots K \tag{13}$$

$$P_L[K \mid S = x_i] = 1 - \sum_{k=0}^{K} P_k^i \tag{14}$$

where

$$p_n^i = \frac{e^{-\rho_i}\rho_i^{\,n}}{n!} \tag{15}$$

The final result for obtaining the cell loss probability estimate can be stated as

$$P_L(K) = \sum_{i=1}^{N} \pi_i (1 - \sum_{j=0}^{K} P_j^i) \tag{16}$$

$$= \sum_{i=1}^{N} \pi_i (1 - \sum_{k=0}^{K} \frac{(1-\rho_i)p_k^i + \sum_{n=1}^{i}\{1-\sum_{m=0}^{n} p_m^i\}P_{k-n}^i}{p_0^i}) \tag{17}$$

3.3 Model Validation and Observations

Details of the validation of the new traffic models and performance methodology are found in Appendix B. In this section the major observations concerning the model will

be summarized. In these studies the load was varied by changing the cell service time in the simulation model. The knee of the delay vs. load curve is the region of the delay profile where the delay values start increasing sharply with an increase in the load. The performance analysis methodology developed here, captures the burstiness in the input trace and predicts delay profiles that correspond to those obtained from real network trace data obtained from the AAI high speed wide area ATM network.

Cell loss profiles from the model and simulation based on collected traces were also compared. These comparisons demonstrate that the cell loss probability estimates of up to 10^{-6} were accurately captured by the new modeled. All the cell loss results were obtained by assuming that the traffic micro-dynamics follow a Poisson process.

The effect of traffic micro-dynamics on queue dynamics has been a point of contention of many studies. In studies like [3.3], it has been pointed out that long-range dependence is the dominant traffic characteristic that determines the queuing performance. However in [3.8], it has been argued that traffic dynamics at the micro level do influence the queuing performance. In order to study the effect of micro-dynamics, different distributions of micro-dynamics are employed and the queueing performance was studied both in terms of the mean cell delay and cell loss ratio. In particular, exponential and uniform distributions are considered here for defining the micro-scale dynamics within a given phase. These results study demonstrate that the delay curves predicted by the model using different micro-dynamics are similar. It was also observed that the effect of micro-dynamics on the average cell transfer delay at a queueing system is not significant. The delay curves obtained from the simulations are similar to those predicted by the model and help establish the relative insensitivity of average cell delay to micro-dynamics in the presence of long-range dependence.

The influence of micro-dynamics on cell loss is also investigated through trace (based on AAI data) driven simulations. As employed in the case of delay results, exponential and uniform distributions are considered for the micro-dynamics. These results indicate that effect of micro-dynamics on cell loss depends on the load of the queueing system and thus unlike the case of average delay, the cell loss probability values are sensitive to the micro-dynamics, at low loads. We also observed that as the load increases, the influence of micro-dynamics on cell loss probability decreases. At high loads the cell loss in the queue is mainly determined by the bursts in the arrival process, which is modeled by the macro-dynamic structure of the input process. While at low loads, the dynamics of cell arrivals represented by the micro-dynamics of the input process dominate the cell loss in the queueing system. We also observe that the loss estimates obtained with exponential micro-dynamics are always greater than the loss probability obtained assuming uniformly distributed micro-scale dynamics.

The results above were obtained using exponential micro-dynamics with fifteen phases. The dependence of mean delay and cell loss results on the number of phases was also

considered. Performance results are estimated for N = 10; 15; 25 and 35 phases. It was found that the performance prediction results are not sensitive to values N.

The accuracy of the model in terms of the second-order statistics of the rate process was also studied. Here the second-order statistics of the rate process are represented by its auto-correlation function. The auto-correlation function obtained from collected trace data was compared with that obtained from a Pareto/Exponential model. The sojourn times in each of the phases is assumed to be proportional to the steady state probability of being in that phase. The results obtained show that autocorrelation functions of the model and real network data match reasonably well even for long lags. The accuracy of the model in capturing higher order statistics like the bispectrum and trispectrum is not studied as such statistics are shown to have a marginal influence on the queuing performance [3.9]. Also note that further work needs to be done for obtaining analytical expressions for computing the autocorrelation function.

3.4 Summary

A simple phase modulated model as well as a corresponding performance evaluation technique were developed and shown to predict the delay and loss performance of AAI ATM flows. One of the ways in which long-range dependence can manifest itself in network traffic is the burstiness associated with the traffic process. By capturing the burstiness of the network traffic in the form of traffic macro-dynamics, the model developed here can predict ATM queue performance that can be expected in future networks. Indeed, extensive trace driven (based on AAI data) simulations performed as a part of this study have shown that the simulation results match the delay and loss predictions obtained by the model. Using the developed model, the effect of traffic micro-scale dynamics on the queueing performance was investigated. The results obtained from the study of micro-dynamics indicate that the mean delay estimate is relatively invariant to the actual micro-dynamics. However, the cell loss probability is found to be sensitive to micro-dynamics at low loads. At higher loads, the bursts on the macro-scale dominate the cell loss in the queue and the micro dynamics play a minor role in the cell loss probability value obtained.

3.5 References

[3.1] Martin De Pyker, "Asynchronus Transfer Mode Solution for Broadband ISDN," Second Edition, Ellis Horwood, London, 1995.

[3.2] V. Paxson and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling," IEEE/ACM Trans.Networking, June 1995.

[3.3] A. Erramilli, O. Narayan, and W. Willinger, "Queueing Analysis with Long-Range Dependent Traffic," IEEE/ACM Trans. Networking, April 1996.

[3.4] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the Self-Similar Nature of Ethernet Traffic", Proc. Acm Sigcomm'93.

[3.5] P. R. Jelenkovic and A. A. Lazar, "On the Dependence of the Queue Tail Distribution on Multiple Time Scales of ATM Multiplexers," Proceedings of CISS, John Hopkins University, 1995.

[3.6] P. R. Jelenkovic and A. A. Lazar, "Multiplexing On-Off Sources with Subexponential On Periods: Part I," IEEE INFOCOM '97, April 1997.

[3.7] J. N. Daigle, "Queueing Theory for Telecommunications," Addison-Wesley Publishing, 1992.

[3.8] L. A. Kulkarni and S. Q. Li, "Measurement Based Traffic Modeling", TISE Technical Report [needs to be fixed]

[3.9] S. Q. Li and C. L. Hwang, "Queue Response to Input Correlation Functions: Discrete Spectral Analysis," IEEE/ACM Trans.Networking, 1993.

4.0 Experimental Performance of TCP/IP over High-Speed ATM over Satellites

Terrestrial/satellite network experiments assist in the design and understanding of future global networks. This section describes the practical experiences gained from measurements of TCP/IP on ATM network over a high speed satellite link and presents performance comparison studies of such networks with congestion and the same host/traffic configurations over local area (LAN) and wide area (WAN) networks. These performance comparison studies on the LAN, WAN, and satellite environments increase our understanding of the behavior of high bandwidth large delay networks, i.e. the results of the experiments given here provide a model for future high speed (Gb/s) global networks of the future. It was found that the high bandwidth large delay systems deliver results similar to the terrestrial fiber networks regardless their path latencies in cases where the communication channels exhibit low bit error rates (BER). NASA's Advanced Communications Technology Satellite (ACTS), with its special characteristics and high data rate satellite channels, and the ACTS ATM Internetwork (AAI) were used in these experiments. Network performance tests were carried out using application-level software (ttcp, Netspec) on OC-3 (155.54 Mb/s) and OC-12 (622.08 Mb/s) ATM satellite links. Details of these experiments can be found in Appendix C.

4.1 System Configuration and Experimental Scenarios

The goal of these experiments was to first determine the maximum attainable throughput over the ACTS elements of the AAI and then to evaluate the performance of the terrestrial /satellite system under stress, i.e., with congestion. The hosts used in these experiments and their configuration is given in Table 4.1.

Name	NIC	Architecture – Clock Speed	Operating System
faraday (KU)	OC-12c	SUN UltraSPARC 1 – 167MHz	Solaris 2.6beta
mckinley (GSFC)	OC-12c	SUN UltraSPARC 1 – 167MHz	Sp;aros 2.5.1
hartley (KU)	OC-3c	SUN SPARC 20 – 125MHz	Solaris 2.4
wiley (KU)	OC-3c	DEC 3000/700 – 225MHZ	Digital Unix 4.0A
elmer (KU)	OC-3c	DEC 3000/700 – 225MHZ	Digital Unix 4.0B
galaga (KU)	OC-3c	DEC 3000/700 – 225MHZ	Digital Unix 4.0
nrl (NRL)	OC-3c	DEC 3000/700 – 225MHZ	Digital Unix 4.0A

Table 4.1 Host Configurations

The ATM switches used in the experiments are FORE ASX-1000 and FORE ASX-200BX models. These switches provide a shared buffer space of 8192 cells for Unspecified Bite Rate (UBR) traffic for each network module (four ports for SONET OC-3c or one port for SONET OC-12c). The UBR buffer space is allocated per virtual circuit (VC) dynamically on an as needed basis. These switches also support the Early Packet Discard (EPD) algorithm, which in case of congestion, and therefore switch buffer

overflow, discards the entire sequence of ATM cells belonging to a single packet, thereby not loading the link with unnecessary cells that will be retransmitted by TCP (in the packet level).

The first experiment focused on determining the maximum attainable throughput, the configuration used is shown in Figure 4.1.

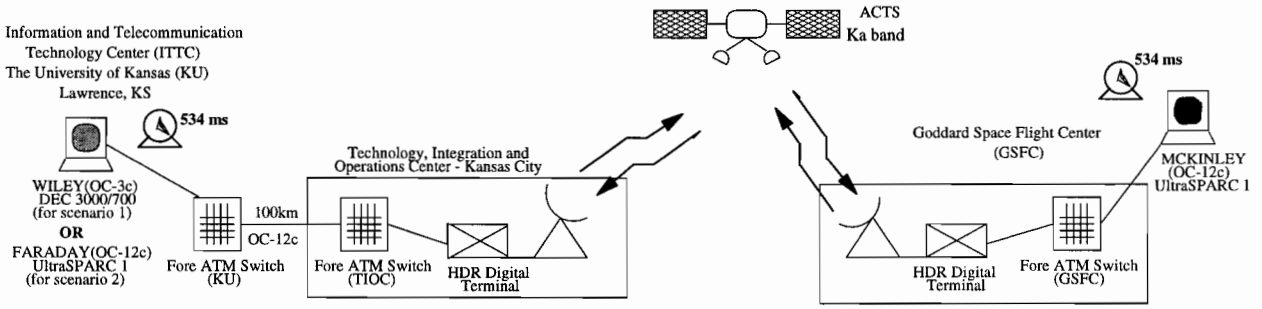


Figure 4.1 Maximum Throughput Network Configuration

In these figures the clocks represent round trip times (measured using ping). The results of this experiment are given in Table 4.2.

Trials	Throughput (Mbps)
1	100.973
2	105.270
3	105.986
4	106.124
5	106.550
6	111.600
7	112.923
8	114.068
9	114.679
10	119.000

Table 4.2 Results of Ten Maximum Throughout Experiments

Three network configurations were used to evaluate and compare the performance of LAN, WAN and terrestrial/satellite networks with congestion. The congestion free LAN and WAN configurations are shown in Figure 4.2. The stressed LAN and WAN configurations are shown in Figure 4.3. Figure 4.4 shows the terrestrial /satellite network configuration under stressed conditions. The results from these experiments are shown in Figure 4.5.

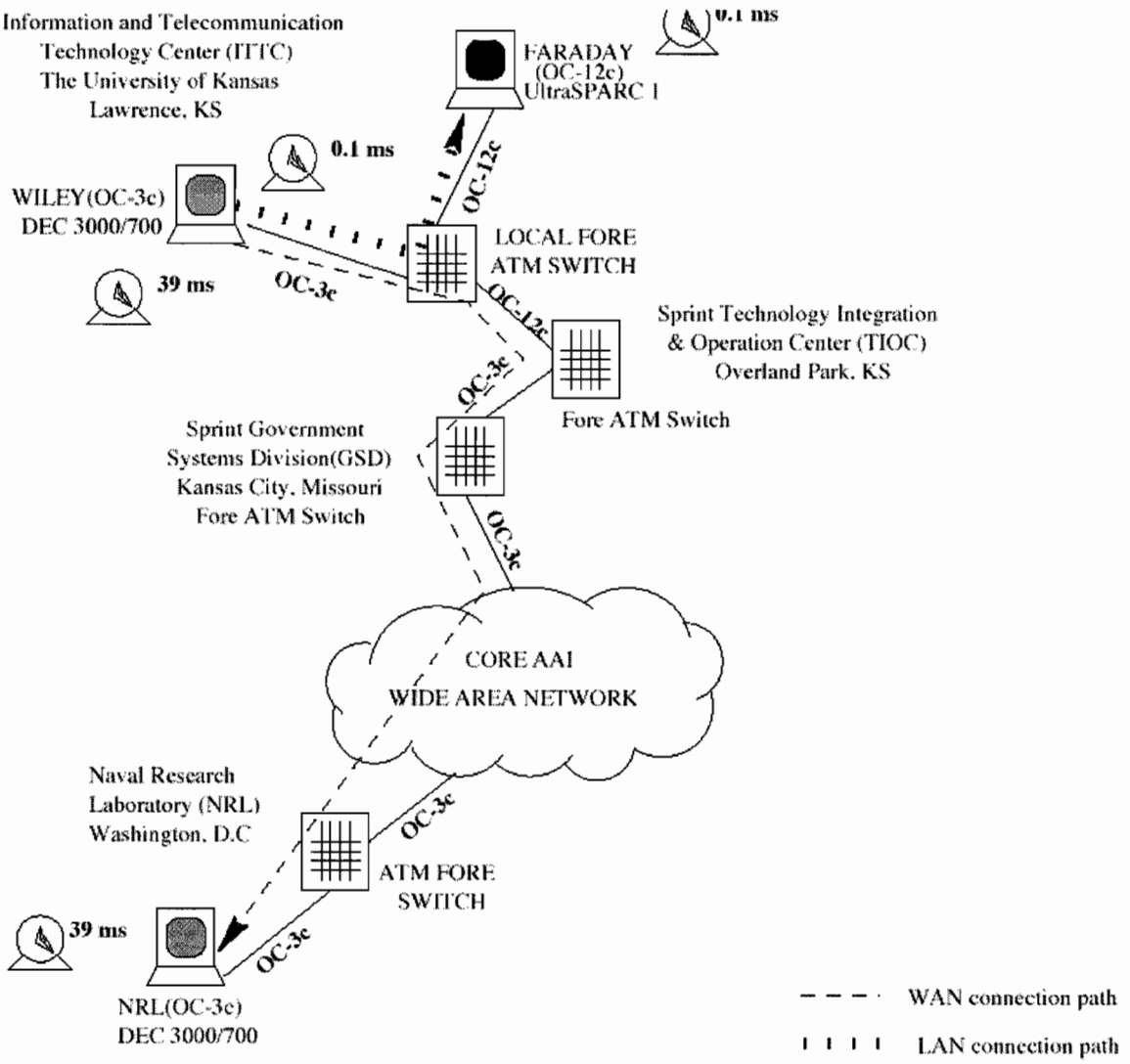


Figure 4.2 Congestion free LAN and WAN configurations

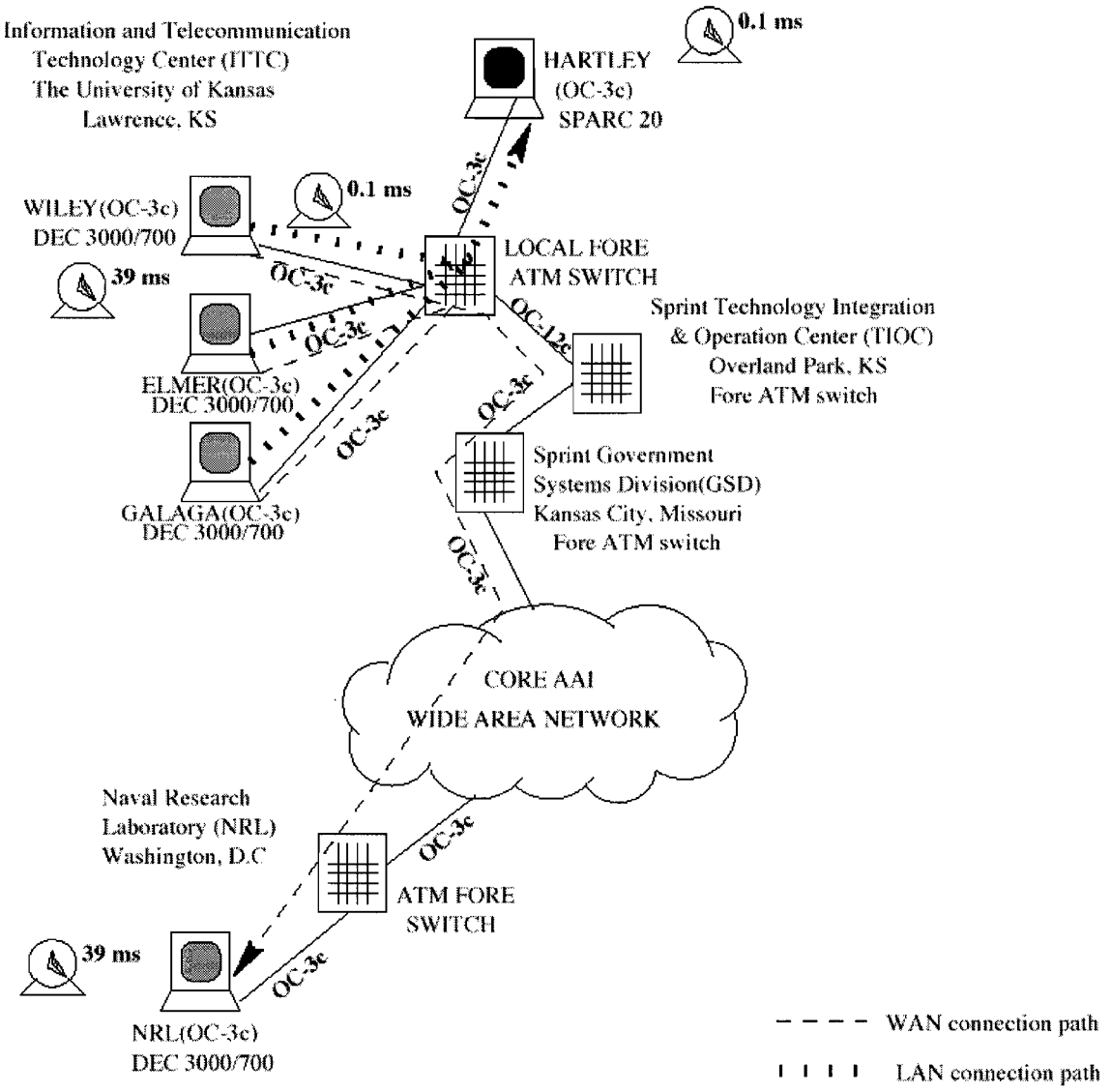


Figure 4.3 Stressed LAN and WAN configurations

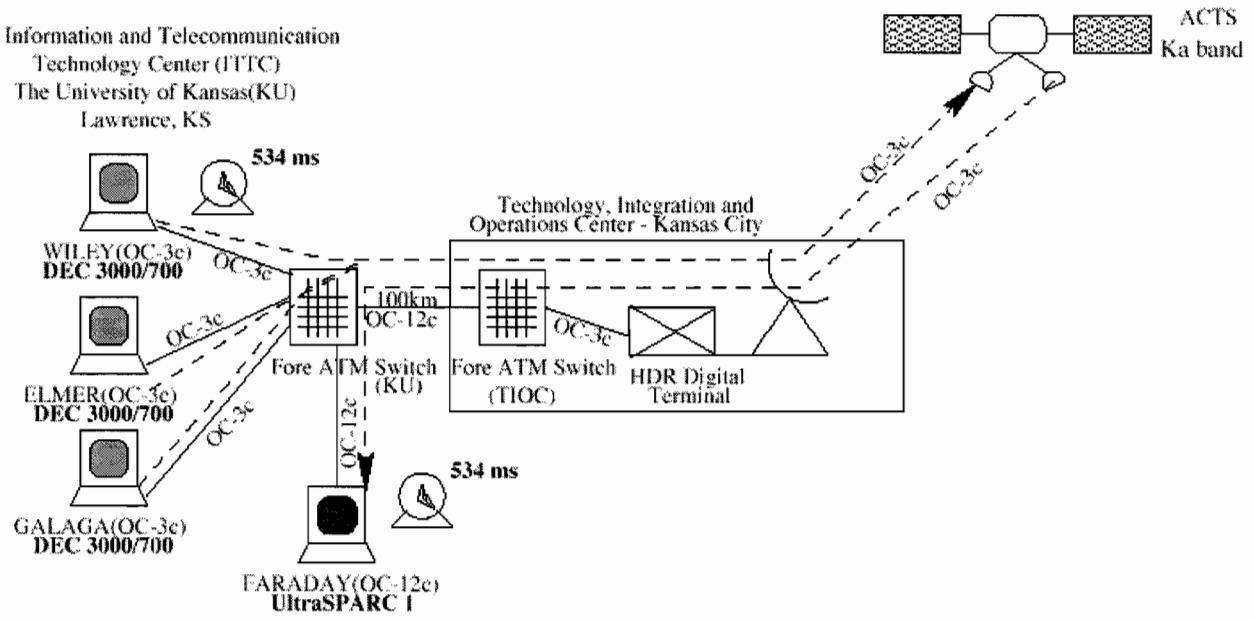


Figure 4.4 Stressed terrestrial /satellite configuration

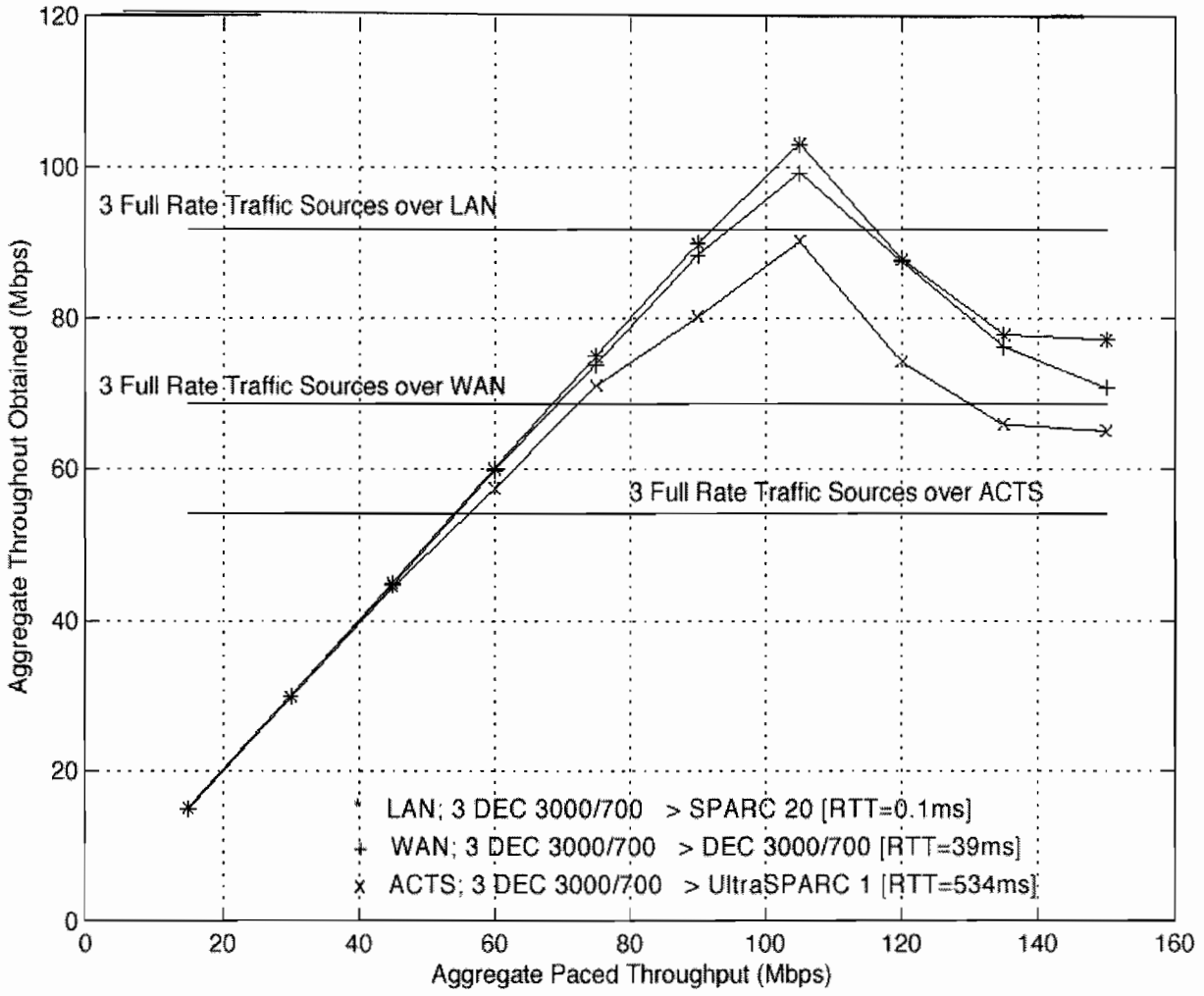


Figure 4.5. Aggregate Throughput Versus Offered Load in LAN, WAN and ACTS

4.2 Results of OC-12c Terrestrial /Satellite Throughput Experiments

In the OC-12c experimental scenario an OC-12 satellite link was established between the TIOC and GSFC. At the transmitting end an OC-12c UltraSPARC workstation (faraday), and at the receiving end another OC-12c UltraSPARC workstation (mckinley) were used. The RTT is dominated by the speed of light delay, which is 534 ms. As shown below, the minimum window size required for maximum throughput is about 34 MB.

Window Size = 534ms X 538.053Mbps = 287.320Mbits = 34.251MB

Unfortunately, due to technical problems in the OC-12 digital terminal at the GSFC ground station at the time of the experiment, we managed to run only two successful

tests with a 10 MB window size in the sender and receiver hosts. In this case the theoretical throughput that we should obtain with the 10 MB window size on an OC-12 link is 157.088 Mb/s. Due to a relatively high BER, (10^{-8} at the TIOC and 10^{-9} at GSFC) reported on the satellite links, we observed 121.835 Mb/s and 127.870 Mb/s instead.

4.3 Discussion of Network Throughput and TCP Performance Pitfalls

The results obtained from the experiments where no congestion conditions were present are slightly less than the theoretical predictions, for all three environments. This is due to the protocol implementations under different operating system kernels in the transmitting and receiving hosts, as well as to the congestion algorithms implemented in the TCP protocol. The slow-start and congestion avoidance algorithms in the TCP protocol have a negative effect on throughput, especially for long delay networks. The maximum throughput will be achieved if the send and receive maximum window sizes are set properly. This is, of course, true in cases where the increase in the window size takes effect instantaneously and not gradually, as in the case of TCP with slow start and congestion avoidance mechanisms. Under no congestion or segment losses, with a window size of 10 MB as used in our satellite experiments (thus, about 1148 MSSs of 9140 bytes in one window) and a RTT of 534 ms, it takes 9.813 seconds (18.377XRTT) to fill the transmission channel. In our experiments, we were transmitting an average of 1 GB (about 102 X Window Size) of data per trial. Therefore, one window of data was transferred in 9.813 seconds, and after that each remaining window of data was transferred in one RTT (534 ms); so in our case we transferred 102 windows of data in about 18.377 XRTT +101 X RTT = 119.377 XRTT seconds (about 63.747 seconds). This is equivalent to (102 X Window Size)/(119.377 XRTT) = 0.854 / (Window Size X RTT), and means that there is a 14..5% reduction in throughput, compared with an instantaneous transmission, caused by the TCP slow-start mechanism in our set of satellite experiments even when no congestion or segment losses were present.

When there is a segment loss, TCP assumes that is caused by congestion and therefore the transmitter has to reduce the rate of injecting data into the network. In the congestion avoidance phase the CWND halves and increases linearly (approximately one segment per RTT) until it reaches its original value. In the satellite experiments conducted here a 10 MB window size and default MTU sizes (9180 bytes) were used, and if we assume that the TCP sender managed to reach the receiver's window (10MB) without losses, it will then take 571.5 segments (5 MB) X RTT = 305.2 seconds to fill the pipe using this algorithm. In the fast recovery phase the CWND halves (plus 3 segments due to the three duplicated ACKS received) and congestion avoidance follows. Thus, for our satellite experiments it will take (571.5-3) segments XRTT = 303.5 seconds to fill the pipe. If the segment loss occurs early in slow start, then it will take hours to fully recover using this algorithm. TCP Reno with fast retransmit and fast recovery improves performance over the basic TCP implementation, but it exhibits another pitfall. Studies [4.1, 4.2] have shown that when more than one loss occurs

within one window, fast retransmit and fast recovery will be triggered several times in one RTT, resulting in reduction of the CWND several times and then linear growth. This leads to a reduction in throughput. Due to these TCP pitfalls, the satellite links must have a very low BER and no losses, otherwise the throughput will drop dramatically. If the satellite links are noisy with high BER and cell losses, then the probability of having more than one segment drop within one window is large, resulting in throughput degradation as discussed above. In the OC-3 experiments, the satellite links exhibited very low BER, comparable with those in fiber-optic terrestrial networks, therefore lost segments and retransmissions were limited, and throughput was comparable to that obtained over LANs and WANs (see Figure 4.5). In the OC-12 experiments, the BER in the satellite links was relatively high, resulting in a degradation in throughput which can be justified using the same logic as above. Even if we have only one segment loss per window which is detected by fast retransmit, it will take 303.5 seconds for the TCP end host to utilize the available bandwidth in the satellite environment. Using the same logic, in the AAI WAN this time is reduced to 22.18 seconds and for the local ATM network to 0.057 seconds. The experiments conducted under congestion conditions show that TCP congestion algorithms are not efficient for TCP over WAN and satellites, and result in throughput degradation in common cases. TCP end traffic sources competing for the same link will cause throughput to drop because of switch buffer overflow and cell losses. Traffic shaping is essential under these circumstances to prevent losses. Increasing the offered load above a certain level will cause the throughput to drop sharply. In the satellite environment, throughput drops faster than in the LAN or the WAN environments. This is because of the TCP mechanism, where bandwidth is wasted due to the long time needed to reach the receiver advertised window size after a segment loss occurs. For the same reason, when no traffic shaping is present and all sources inject data in the network simultaneously as fast as they can, the aggregated throughput obtained over ACTS is lower than that obtained over the WAN or LAN.

4.4 Remaining Experiments Over High-Bandwidth-Delay-Product Networks

During the second half of 1997 there were maintenance problems with the HDR terminal, both at the TIOC and GSFC. As the TIOC facility becomes available (at OC-3) we plan on conducting a comparison of four transport protocols. The four implementations that will be testing are TCP Reno, New TCP Reno, TCP SACK (Selective Acknowledgement) and an experimental new transport from NASA, SCPS-TP (Space Communications Protocol Standards - Transport Protocol).

The objective of this evaluation, is to determine how these four TCP implementations, with different congestion control algorithms and different philosophies perform over ATM over high data rate satellite channels, i.e. over high-bandwidth-delay-product networks.

The four implementations with their major characteristics are summarized very briefly below (note that all implementation support portion or all of RFC 1323, the extensions for high performance networks):

TCP Reno:
1) Jacobson Congestion Control: slow start, congestion avoidance, fast, retransmit, fast recovery.
2) Reactive algorithm
3) Poor performance when multiple drops within the same window of data sent.

TCP SACK:
1) SACK congestion control algorithm: Selective acknowledgement and cabability to transfer many lost segments in one RTT. Similar performance with Reno when no or one data drops.
2) Reactive algorithm
3) Other characteristics are under investigation.

New TCP Reno:
1) J. Hoe's congestion control algorithm: Capability of recovering from N data losses in N RTT.
2) Reactive algorithm
3) Other characteristics are under investigation.

SCPS-TP
1) Support for both Jacobson and Vegas congestion algorithms. For our test purposes, we will be using SCPS with TCP Vegas congestion algorithm.
2) Proactive algorithm (TCP Vegas philosophy)
3) Out of kernel (user space) implementation.
4) Ability to start a connection either with TCP Vegas and corruption default loss control or with TCP Vegas and congestion default loss control.

From our planned experiments we expect to learn the suitability of these protocols for IP/ATM Terrestrial /Satellite networks. These transport protocols have been implemented and tested in the ITTC local IP/ATM environment. The limiting factor in completing this study is the availability of an operational HDR terminal and suitable scheduled satellite time.

4. 5 References

[4.1] Furquan A. Ansari. Adapting TCP/IP over ATM, Master of Science Thesis, University of Kansas, 1996.

[4.2] Kevin Fall, Sally Floyd. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. Computer Communications Review, July 1996.

5.0 Overview of NetSpec

5.1 NetSpec Evolution

The first version of NetSpec was simply a modified version of TTCP [5.1]. It had a central point of control that enabled a user to control multiple ttcp connections simultaneously. TTCP is limited in its functionality to full stream, constant sized write calls. The first version of NetSpec was obsolete by the time it was completed. The AAI requirements were beyond what a 'remote ttcp' would offer. See Figure 5.1.

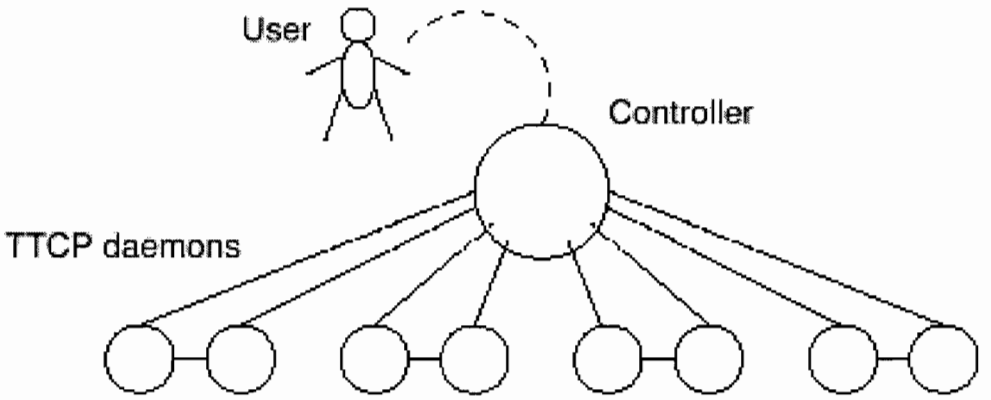


Figure 5.1 NetSpec Version 1 Architecture

The design of NetSpec 2 revolved around a central controller, and the controller also implemented the user interface. See Figure 5.2. The major differences with NetSpec version 1 were:

- Formalized script language
- Introduction of the reporter daemon, designed to off load the controller
- The traffic generators,
- The synchronization of the traffic generators
- A binary control protocol

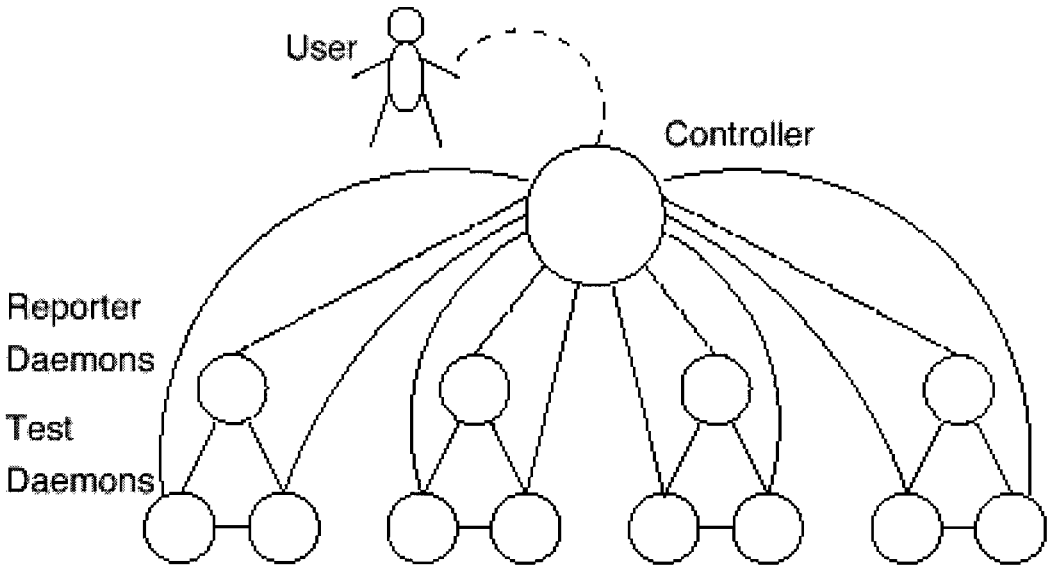


Figure 5.2 NetSpec Version 2 Architecture

Shortly after Version 2.0 was released it showed its weaknesses, and as such a thorough overhaul of the design was required. The major problems that were encountered during the use of Version 2.0 were:

- A single controller is cumbersome in networks that are not fully meshed (A can see B, but not C, B can see C though)
- The operating system limits impose limits on the central controller, which leads to scalability problems.
- The binary control protocol imposes tough limitations on extension of the protocol itself, and maintaining backward/forward compatibility at the same time. (Adding new parameters posses multiple problems for transparency.)
- Distribution of the controller, a centralized controller posses too much scalability problems.
- Semantical problems within the scripting language.
- Framework extension with new features too complicated.

To achieve the goals for the AAI NetSpec Version 3.0 was developed. The design of NetSpec Version 3.0 stressed portability, and modular, readable code, so that it will be easily extended and modified. The unique contribution and the key to the success of NetSpec Version 3.0 is its control framework, which will be summarized next. Complete details of the design and implementation of NetSpec can be found in Appendix D. Detailed examples of NetSpec experiments along with their corresponding scripts can be found in both Appendix A and D.

5.2 NetSpec Control Framework

The control framework is the part of NetSpec that by its definition is inherently complex; distributed control must be implemented over a wide area network. From the initial two versions of Netspec it was learned that a single central controller did not scale sufficiently for WAN performance evaluation. Aside from scaling it was also learned that the coupling between the leaves (performance hosts) and the controller is fragile. The two most striking aspects are that the control of the leaves needs to be fairly tight while the information being passed between the leaves and user needs to be fairly loosely coupled. A good example is that the controller should only have a syntactical understanding of the parameters being passed to the leaves; it should not in any way attribute semantic values to these parameters. On the other hand, in order to achieve synchronized execution the controller should be able to tell the leaves exactly what to do and when. This section focuses on the controller; the user interface will also be discussed.

5.2.1 Specifications

The controller needs to provide the glue between the user and the leaves. The user interface itself is assumed to be providing a simple pathway to the controller. The conceptual picture of the controller is depicted in Figure 5.3.

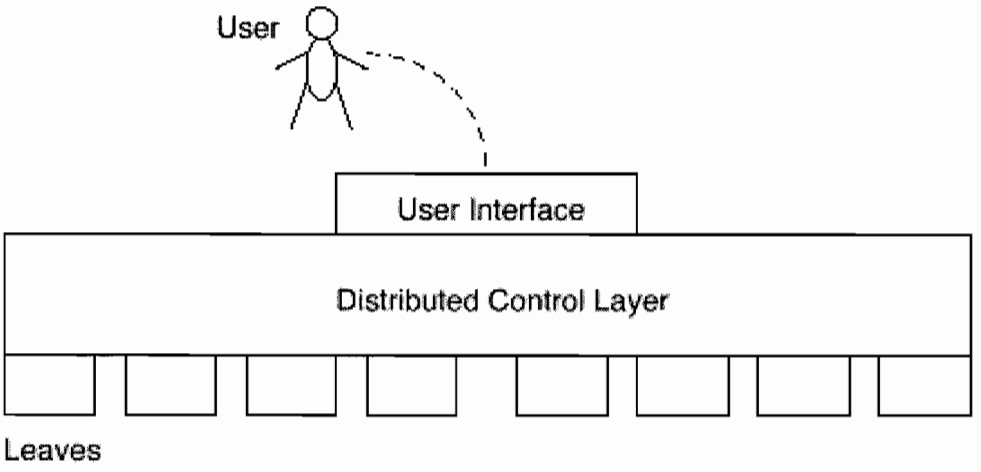


Figure 5.3 NetSpec 3.0 Controller Architecture

The controller has the following attributes:

Distributed. The controller is capable of being distributed across multiple machines and or processes. The reason for this is twofold; connectivity between nodes can not be expected to be full mesh, second, in order to circumvent host limitations, either multiple processes and or distribution over multiple hosts is in order. Also the distribution

aspect should if at all possible not impact the language if there is no need for distribution.

Control Language. The user is able to specify an experiment in a simple block structured language. The language is able to specify either parallel and serial execution constructs.

Transparency. The controller is oblivious to the parameters that are passed to the leaves. No semantic value can be attributed to the parameters that are passed to the leaves. This is done to guarantee controllers to be independent of the leaves

5.2.2 Controller Design

Two main design issues of the controller are an ASCII based protocol and its need to be distributable across multiple nodes. An ASCII based protocol will require parsing in order to be able to pass on parameters. In addition, the controller is required to interpret commands. Usually interpreters and compilers can be divided in various subfunctions. The two fundamental functions that form the compiler are a lexical analyzer and a parser. The lexical analyzer takes the input stream and "tokenizes" it; that is, it chops a stream of letters and punctuation marks into words. The parser that controls the lexer then interprets the words and tags a meaning to the sentence that comes out. Parser based protocols are common, e.g., FTP and Telnet are ASCII based control protocols.

The basic implementation and design of the language and the protocol is fairly straight forward. The difficulty is not the language or the control protocol; the far most significant problem is being able to distribute measurement tasks across multiple nodes in an arbitrary and user friendly fashion. One of the techniques that is used in parallel parsers is a technique called recursive descent. Recursive descent is a technique in which every semantic block is parsed by an iteration of the parser. Most block structured languages have properties that allow a recursive descent parser to function easily. Considering the requirements on the NetSpec language which is block structured, a recursive descent principle is applicable. Instead of a parser iterating itself in order to parse the blocks, it can also be done by multiple threads of execution, and this is exactly why it is being applied in parallel parsing techniques. In Figure 5.4 a block structure is given with potential different threads of execution.

This structure is exactly what the distribution required of the NetSpec controller; different threads of execution. The only major disadvantage of using a recursive descent technique is the resulting structure is inherently a hierarchic tree. The other problem with recursive descent is that the interaction between the blocks determines the level of parallelism one can achieve. If the interaction is fairly minimal, i.e., block "a" does not affect its enclosed block "b," then using different threads is possible. In programming languages this is not always true, and as such recursive descent applied

in parallel compilers is not common. However, for the structure and level of interaction between nodes that NetSpec requires an arbitrary tree type structure is well suited. The user interface represents the root of the tree. The idea here is to use a TCP/IP connection for each recursion step which simplifies implementation considerably. Here each iteration is transparent to the previous one. Each iteration can be on a different host. Another major advantage of this approach is that it deals with the problem of multiple nested constructs for parallel execution. If other techniques were to be applied, the controller needs to parallelize itself in order to control more than 1 nested block. In Figure 5.5 the resulting tree structure of the parsing of the blocks in Figure 5.4 is shown.

The application of a recursive descent principle without collapsing the results of the parse back to the root of the tree is somewhat unusual. In our case the result of the parse sets up the entire control architecture, after which the actual connections between the instantiations of each of the blocks provide the required communication structure.

```
A {
    B {
        D {
            ...
          }
        E {
            G {
                ...
    }
            H {
                ...
              }
          }
        F {
            ...
          }
    }
  C {
      ...
}
```

Figure 5.4 NetSpec Block Language

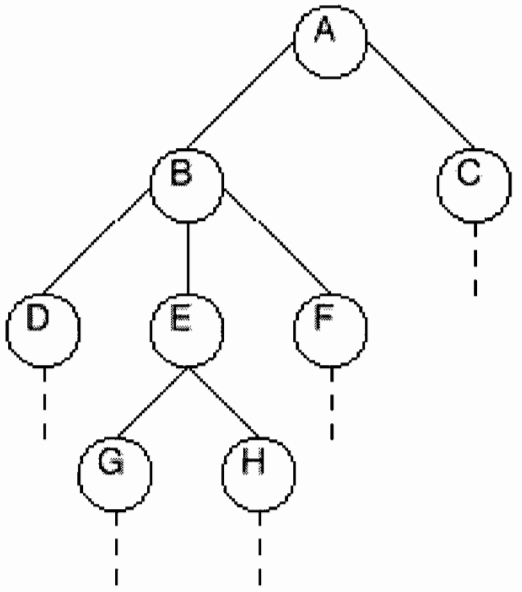


Figure 5.5. Tree Structure of the Parsing of the Blocks in Figure 5.4

The language should have familiar syntactical properties; C like features. Since the prospective audience of these tools is the academic and research communities this should form a minimal obstacle to learning NetSpec. The next requirement is that it needs to be block structured. Though the controller can not attribute any semantic value to most of the script it still needs a way to derive connection addresses and execution behavior. Principally there are two basic forms of execution;

- Parallel
- Serial

Intuitively the idea is to use a syntax like outlined in Figure 5.6. Using this approach there exists a clear distinction between the parts that the controller needs to understand, the keyword that determines the execution type and the keywords that represent the address at which to contact the next node. This construct also nests easily, and is suitable for the recursive descent distribution approach. There is only one subtle problem with this syntax, the top level block defines the current node. Obviously, the current node can not contact itself, in other words, the address lost its meaning, but the execution keyword did not. The simple way around this is to just ignore the address within the current block. That way the current node can derive its intended execution behavior.

parallel <address> {
...
}

Figure 5.6. Basic Block Syntax

A special case of execution constructs is needed in order to distinguish between various leaves. It is stated clearly that the controller can not attribute any semantic value to the parameters passed within the blocks. Essentially the keywords parallel and serial mean that the controller needs to be addressed. Similarly, for each of the leaves we could create unique keywords that could be used to address the specific leaves. Like the controller it would be possible to use multiple keywords to address one leave.

Since the primary objective of the control framework is to control traffic sources and sinks the control protocol is tailored to execute network connections with as precise control as possible. The way any network connection can be charachterized as shown in Figure 5.6.

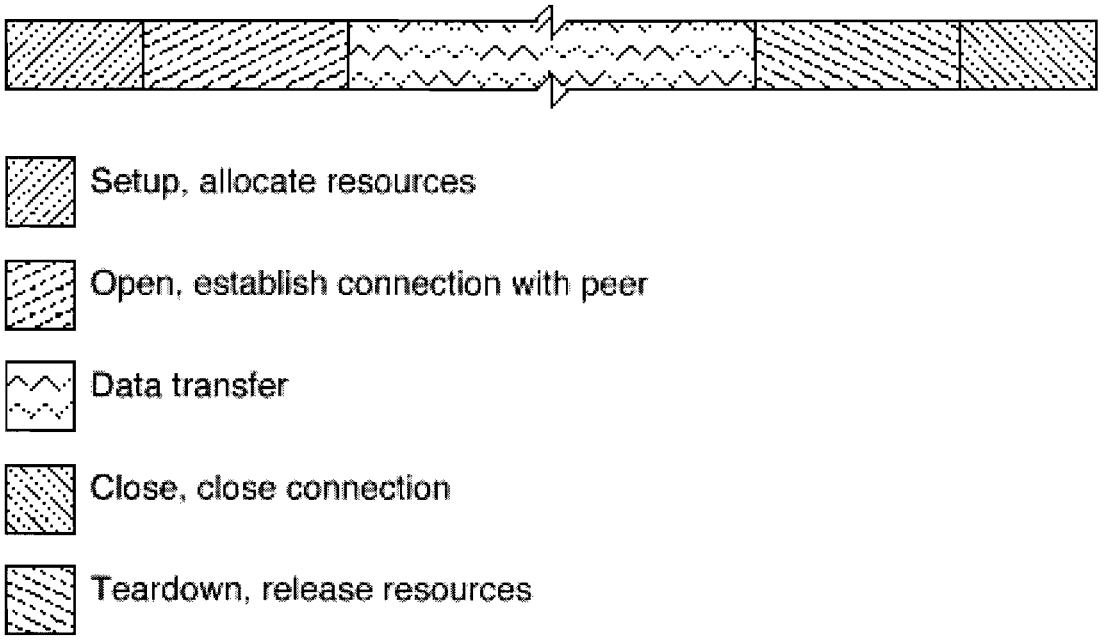


Figure 5.6. Phases of a Network Connection

The phases of the connection as such dominate the protocol that is used to control the leaves. Other leaves that are associated with doing measurements related to the connection will fit this protocol reasonably well. The only difference is that some measurements lack the open and close phases. Within the NetSpec implementation this simply results in a null action. There is also a fairly distinct difference between

unreliable and reliable protocols. Within the open phase of a connection, the reliable protocols will establish a connection, and as such will exchange data. Unreliable protocols in principle do not establish a connection. To control the phases of the connection the following commands are implemented in NetSpec:

setup Allocate memory, fork processes, create service access points.
open Establish connection, i.e., TCP initiates the 3 way handshake.
run Start the data transfer.
finish Finish the data transfer.
close Close the connection.
teardown Free up all resources allocated, do not however discard test results.

Among these five core control commands there is also a need for some administrative commands:

report Tells the leave to return its results.
reset Reset the state of the leave to it's initial state if possible.
kill Over and out.
parameters Transfer parameters, do nothing else.
config Request configuration information.

Principally every command has the syntax shown in Figure 5.7.

```
keywordCommand {
        ...
}
```

Figure 5.7. Command Syntax

All of the commands can be accompanied by parameters which follow the exact same format as lined out for the control language.

As mentioned before, the controller nodes should not have a semantic understanding of the parameters, but should have at least a syntactic understanding of the parameters. The reason for this is, first the controller can become derailed simply by problems within parameters, next there is a need for checking at the highest level possible, so simple syntactical errors do not propagate through the entire control tree, before being detected. The syntactical checking has one drawback though. It slows the process of setting up the control structure. But setting up connections for each recursion takes considerably longer. Seen in that light the overhead of syntactic checking is not significant.

The general specification of a parameter is <keyword> = <value> in which the keyword is an identifier and the value can be any of the following:

IP address Either the numerical or dns representation of an IP address, i.e., peer = stephens.tisl.ukans.edu, own = 129.237.125.220. There is one variation of this that is non standard, and that is whenever there is a need to identify specific ports, the ip address needs to be extended with the port number which is an unsigned short integer. i.e. peer = stephens:36644, in this the colon is used as delimiter.

Identifier A typical (C) definition of an identifier. i.e.,

> ***String*** A string constant,
> ***Integer*** A plain integer, there is also a slightly more involved version of this, for example an integer with a unit multiplier, ie window = 16K, which would be 16 Kilo byte.
> ***Real*** A foating point double precision number, a unit multiplier also can also be applied.
> ***Function*** A function is a concatenation of parameters, ie. protocol = tcp (window = 131072, tcpmtu=536) .

All the parameters can be internally dealt with as plain strings. There is no need for conversions until the parameters are handed off to the leaves. Refer Appendix D for the specifics of the implementation of the parameter and slave control protocol parsers.

In order to respond to any of the commands issued there needs to be a set of defined acknowledgements, here:

acknowledge Command is executed correctly.

error Command could not be correctly executed.

report Has the same meaning as acknowledge, the way the text is handled is different, purely for performance purposes.

All of the replies can be accompanied by text which is passed on to the user interface in a verbatim manner. Due to the volume of information that reports can represent, the report command is dealt with differently than the acknowledge and error replies. Acknowledge and error replies are accumulated on a per node basis and are as such propagated to the parent levels. An error that is received by the controller by one of its leaves results in an acknowledge to the parent. There is no warning reply. The reason for this is that a warning is basically a non-fatal reply, so it serves an information purpose only. This can be represented by an acknowledge accompanied by a message.

The only major problem associated with using a recursive descent technique is that the communication between various nodes through the use of the control channels is somewhat cumbersome. In principle if leaves have a need to communicate they could potentially set up a direct connection to each other. This would at a minimum require

the user to supply more information in the script in order to enable the leaves to set up a peering connection. The obvious way around this problem would be to use the existing control channel. The problem with this is that in principle the controller does not have any knowledge of relationships between leaves. The solution was to enlighten the controller somewhat as far as relationships between leaves goes. In addition to the earlier discussed parallel { } and serial { } constructs, a third one is introduced here, the cluster { }. The cluster { } is in behavior very similar to the parallel construct. The difference is that with cluster the controller can conclude that the leaves that are enclosed within the construct are associated with each other. This opens up the potential of using the control channels as communication channels for the peering leaves also. The initial setup phase is still somewhat cumbersome. The peer that needs information is required to emit a request to the controller and the controller rebroadcasts the information in a sequential manner to the other leaves within the cluster. If the peering leave gets the request it turns around and responds with the requested information. From there on the controller "knows" which peers are associated with each other. This feature has not yet been implemented in NetSpec 3.0, though the structure provides mechanisms. The mechanism is called directed broadcast; it is a broadcast, but directed at a certain part of the tree, and aside from that it is sequential in nature, which is not the real definition of broadcast.

Replies generated by one of the leaves within a cluster are dealt with differently than in the case of a parallel or a serial constructs. The reason for this is that within a cluster leaves interact with each other, so if one fails, the entire cluster can be considered failed. For a cluster this means that if one of its leaves generates an error reply, then the controller itself also needs to generate an error reply. The reason for this is that when an error is considered fatal for either of the three constructs a single error somewhere in the tree becomes fatal for the top level controller and as such will essentially cause the experiment to fail. While this most certainly will not be true, only a part of the experiment will have failed. This subtle drawback can be contributed to the choice for a tree structure; in any given tree structure with a single root this would have proven to be a problem. Again this is also justifies the use of a separate construct for the conglomeration of closely related leaves.

5. 3 Summary of NetSpec

Extensive examples of NetSpec experiments can be found in Appendix A. NetSpec has proven to be a successful tool for the evaluation of wide area high performance networks. One of the major accomplishments achieved with NetSpec was the 772 successful experiments, 150 hours total, about 1 gigabyte of data that took 24 hours to process, as discussed previously. Other research that has benefited from NetSpec include Firewall testing [5.2] and TCP Pacing [5.3]. In both these efforts NetSpec was an instrumental tool that simplified the problem of extensive testing. Over 140 academic and research organizations have obtained a copy of NetSpec. Their feedback has been both positive and useful. NetSpec has been designed to be robust and

extendable. We anticipate that the framework developed here will continue to evolve as the tool is used for new experiments on other networks.

5.4 References

[5.1] Roelof J.T. Jonkman. The design and implementation of NetSpec. University of Kansas, Information and Telecommunication Sciences Laboratory, May 1995.

[5.2] Steven G. Pennington. Gauntlet rewall performance benchmarks. http://www.ittc.ukans.edu/

[5.3] Brian Buchanan. Header compression on atm network. Master's thesis, University of Kansas, January 1998.

6.0 Summary of 1997 AAI Traffic Flows

6.1 Introduction

This section describes the traffic monitoring process conducted on the AAI network for a twelve month period between January 1997 and December 1997. In addition to monitoring the sites on the AAI network, certain sites on the MAGIC network were also monitored. In Section 6.2, the topology of the network and the switches that are being sampled is discussed.

The sites involved in the measurement process are listed and significant changes in the topology of the network are shown. Section 6.3 describes the methods and tools used for collecting the measurements and obtaining the plots. In Section 6.4, traffic profiles from data transfers of specific demonstrations over the AAI network are presented. Initial observations regarding the traffic flows are discussed in Section 6.5. Following Section 6.5, the plots are shown on a per-site basis site for each site that was monitored. More detailed information about each site is presented before the plots for that site. This information includes the details about the switch at each site. The information also shows the different active connections on the ports that are being sampled. The traffic patterns are correlated with network usage when information about the particular data transfers is available. Knowledge of application profiles is important for 1). Efficient design of congestion control algorithms and 2). For constructing empirical, work load models for driving WAN simulations.

6.2 Sites and Topology

During the sample year specific AAI edge switches and MAGIC switches were monitored at 60 second intervals at the following sites.

1. Naval Research Lab (NRL).
   Location: Washington, D.C.

2. Army Research Lab (ARL).
   Location: APG, Maryland.

3. Naval Command Control and Ocean Surveillance Center (NCCOSC).
   Location: San Diego, California.

4. Corps of Engineers Waterways Experiments Station (CEWES).
   Location: Vicksburg, Mississippi.

5. Naval Research Lab, Stennis Space Center (NRLSSC).
   Location: Stennis Space Center, Mississippi.

6. Government Systems Division (GSD).
   Location: Kansas City, Missouri.

7. EROS data center (EDC).
   Location: Sioux Falls, South Dakota.

8. Sprint Technology Integration and Operations Center (TIOC).
   Location: Overland Park, Kansas.

9. University of Kansas (KU).
   Location: Lawrence, Kansas.

At each of the above-mentioned sites, at least one port on the stated switch was being sampled. All the switches currently being monitored are FORE switches. Switches inside the AAI cloud are not monitored as a part of this study. The SNMP queries for each switch are done through the switch Ethernet connection through the Internet, through the AAI network using ATM, and in some cases, it is done both ways. The use of the Ethernet connection through the Internet or ATM for monitoring refers to the network used for sending the periodic SNMP queries and getting the responses. For example, the FORE switch at NRL is monitored over the AAI network using ATM while the FORE switch at KU is being monitored through the Ethernet connection. The FORE switch at EDC is being monitored over both Internet and AAI using ATM. The names of the switches, which are monitored at each site, are shown in the Table 6.1 and the overall topology of the network is shown in Figure 6.1. The site names are as listed at the beginning of Section 6.2.

Table 6.1: Switches at the Various Sites that are Monitored

SITE	SWITCH NAME	IP NUMBER
NRL	aai-pop.nrl.aai.net	204.235.68.1
ARL	aai-pop-ether.arl.aai.net	128.63.58.58
NCCOSC	aai-pop-ether.nccosc.aai.net	134.164.124.5
CEWES	aai-pop-ether.gsd.aai.net	192.157.66.194
EDC	merlin.edc.magic.net	198.207.141.252
TIOC	hertz.tioc.magic.net	198.207.141.241
NRLSSC	aai-pop-ether.nrlssc.aai.net	128.160.10.111
KU	spork.tisl.ukans.edu	129.237.125.231

As can be seen from Figure 6.1, the sites EDC and KU are not directly connected to the AAI network. Both of the sites, which are on the MAGIC network, are attached to the AAI network at the TIOC site. In particular both the sites are connected to the AAI

network through FORE switches at the TIOC site and SPRINT GSD. Figure 6.1 also shows the link capacities of the different links connecting the switches as of December 31, 1997.

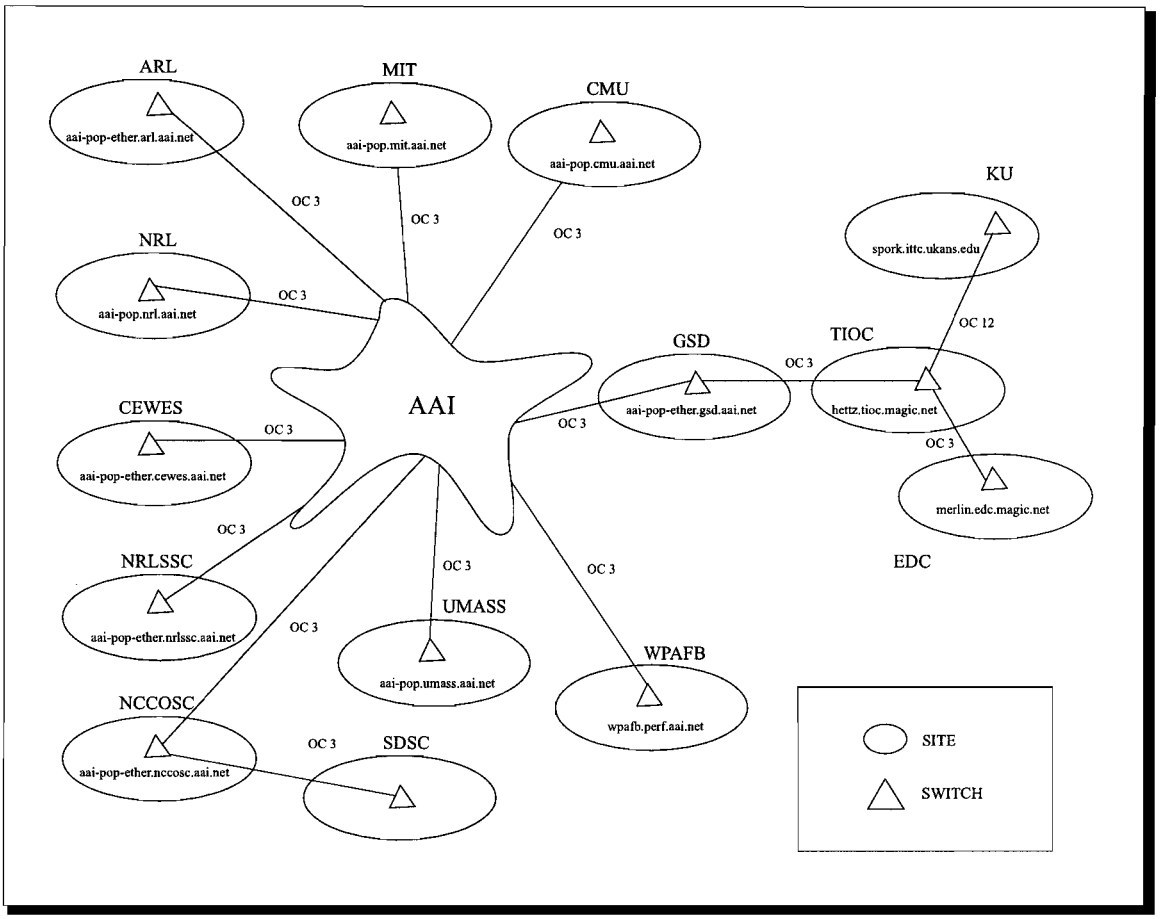


Figure 6.1: Connections of the Sites and Switches During 1997

6.3 Throughput Measurement

In this section, the methods and tools that have been used for obtaining the measurements are discussed. Data was collected in the form of switch buffer cell counts, using SNMP. The method adopted relies on querying the SNMP agent that is maintained on the FORE switch. The SNMP agent is maintained by the SNMP daemon running on the switch. The agent supports a Management Information Base (MIB) containing information about the operation of the particular switch. Among other things, the MIB supports counters for the number of cells transferring over a port, a particular virtual path, or a particular virtual channel. Traffic flowing out of a port cannot be monitored on a virtual channel basis, as the FORE MIB does not support the MIB element. The information can be obtained by querying the appropriate MIB variable. SNMP requests are sent to each AAI switch every 60 seconds from a SPARCcenter 2000 machine at KU. This is shown in Figure 6.2 where

armstrong.ittc.ukans.edu is the name of the SPARCcenter. The requested cell count is returned as response to the request. The cell count is time-stamped when it is successfully received.

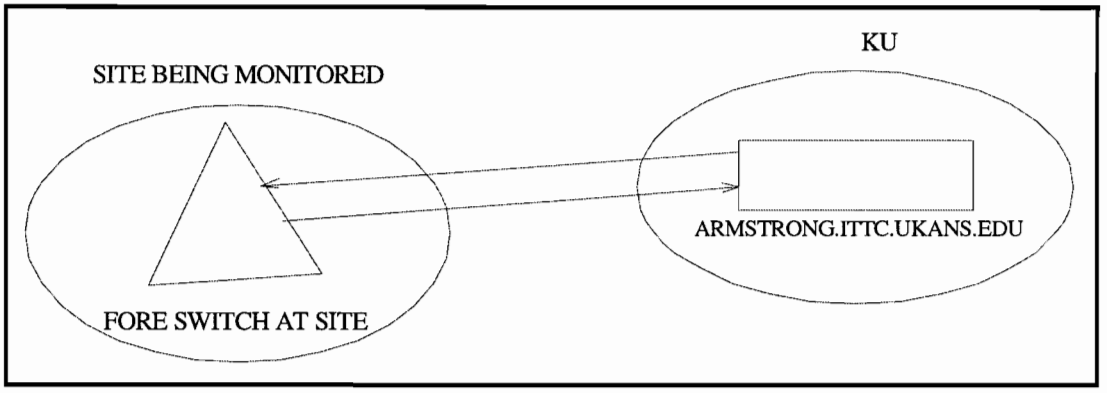


Figure 6.2: SNMP Requests and Responses.

The Tricklet library has been used for making the SNMP requests. The Tricklet library offers a convenient interface for obtaining network management information using SNMP. In particular the SNMP-GET, SNMP-GET NEXT, SNMP-TBL and SNMP-SET requests are implemented in the Tricklet library. The SNMP-GET requests the value of the specified MIB variable. SNMP-GET NEXT retrieves the next MIB variable in the hierarchy to the specified variable. SNMP-TBL is implemented as a series of GET and GET NEXT requests. It returns a group of MIB variables, which form a subtree with the requested variable as the root. The cell counts obtained from the responses to the SNMP requests and their corresponding time-stamps are continuously archived. Successive time-stamps in the archived data differ by approximately 60 seconds, as the SNMP requests are sent out at approximately 60 second intervals with a variation of a few seconds. From this data, the throughput for each sampling period is calculated as the time average of the number of cells transferred during this period. The length of the sampling period is equal to approximately 60 seconds.

$$Y(t) = \frac{[x(t+\Delta t) - x(t)] * 424}{\Delta t} b/s$$

Where,
x(t) is the cell count at time t.
Y (t) is the throughput at time t.

In the above equation, Δt is approximately 60 seconds as the actual values of Δt are obtained from the observed time-stamps. The GNU utility GNUPLOT has been used to obtain the throughput vs. time plots in a format that can be printed. Since SNMP uses the underlying User Datagram Protocol (UDP) for transporting SNMP queries and

getting responses, connectivity to the sites is crucial for the data collection process. FORE 's ATM network interface management tool ami was used to check if the SNMP daemon on the switch at the site was reachable. The ami tool provides a convenient way for obtaining information about the current operation of the switch by reading the FORE MIB variables supported on the SNMP agent. This tool was also used to check the state of the ports being monitored and changes in the remote connections to these ports. In addition to monitoring the traffic flows, throughput experiments to study ATM WAN performance were conducted between different sites on the AAI network. The throughput experiments were possible as KU had performance machines connected to the FORE switches at the following sites: NRL, ARL, EDC, GSD, and NCCOSC. Information on the KU performance machines at various sites is summarized in Table 6.2.

Table 6.2: List of KU Performance Machines

SITE	KU PERFORMANCE MACHINE	TYPE	IP-NUMBER
NRL	nrl.perf.aai.net	Dec Alpha 3000/700	204.235.68.3
ARL	arl.perf.aai.net	Dec Alpha 3000/700	204.235.65.6
NCCOSC	nccosc.perf.aai.net	Dec Alpha 3000/700	204.235.67.223
EDC	edc.perf.aai.net	Sun Sparc 10	204.235.71.132
GSD	gsd.perf.aai.net	Sun Sparc 10	204.235.71.3

The KU AAI Data Plotter is a web-based tool created to display throughput vs. time plots of the monitored port for any specified period of time during which the port was monitored. The plots are obtained from the data collected and archived at KU. Any of the plots that are presented in the later sections in can be viewed with the data plotter. The data plotter is a convenient tool for viewing and analyzing the traffic profiles generated by experiments being done on the network. The data plotter was designed to accentuate the peak transfer rates. That is, when a graph is viewed of several weeks or longer, peak transfer rates may make it look like there is more average traffic then actually exists. This is done so that the user may determine what section of interest on the graph they would like to zoom-in. In order to observe events at various parts of the network, a tool was developed to summarize each day of data collected in the form of thumb-nail gifs plotted on a per site basis, for every port being monitored. The gif images can be accessed from the data plotter page given earlier. In addition, the gif images act as links to cgi-bin scripts so that the current traffic profiles at the port of interest can be viewed when clicked on. This way, the user does not have to wait for automatic updating of plots, which occur every 3 hours. When plotting Megabits per

second, the AAI Data Plotter will use 53 bytes per ATM cell, which is the sum of the cell header and payload.

6.4. Significant Experiments

Netspec, a tool for network experimentation and measurement, was used to do the majority of testing from KU. Most experiments performed by KU students used Netspec. Throughout 1997, numerous experiments were conducted through the Advance Communications Technology Satellite (ACTS) (see Section 4). Most of these tests were bent pipe OC-3 satellite experiments to test different TCP implementations on ATM/SONET Unix machines over OC-3 ACTS links. A few OC-12 tests were conducted, when equipment permitted, from a KU Sun UltraSparc 1 to a Goddard Space Flight Center (GSFC) Sun UltraSparc 1. Figure 6.3 shows a peak of 174 Mbps during one set of OC-12 tests. This ATM data was sent from KU to the TIOC where it was transmitted up to the ACTS satellite and then directed back down to GSFC. The tests shown in Figure 6.3 were performed by using Netspec in full blast mode with a 10MB window size.

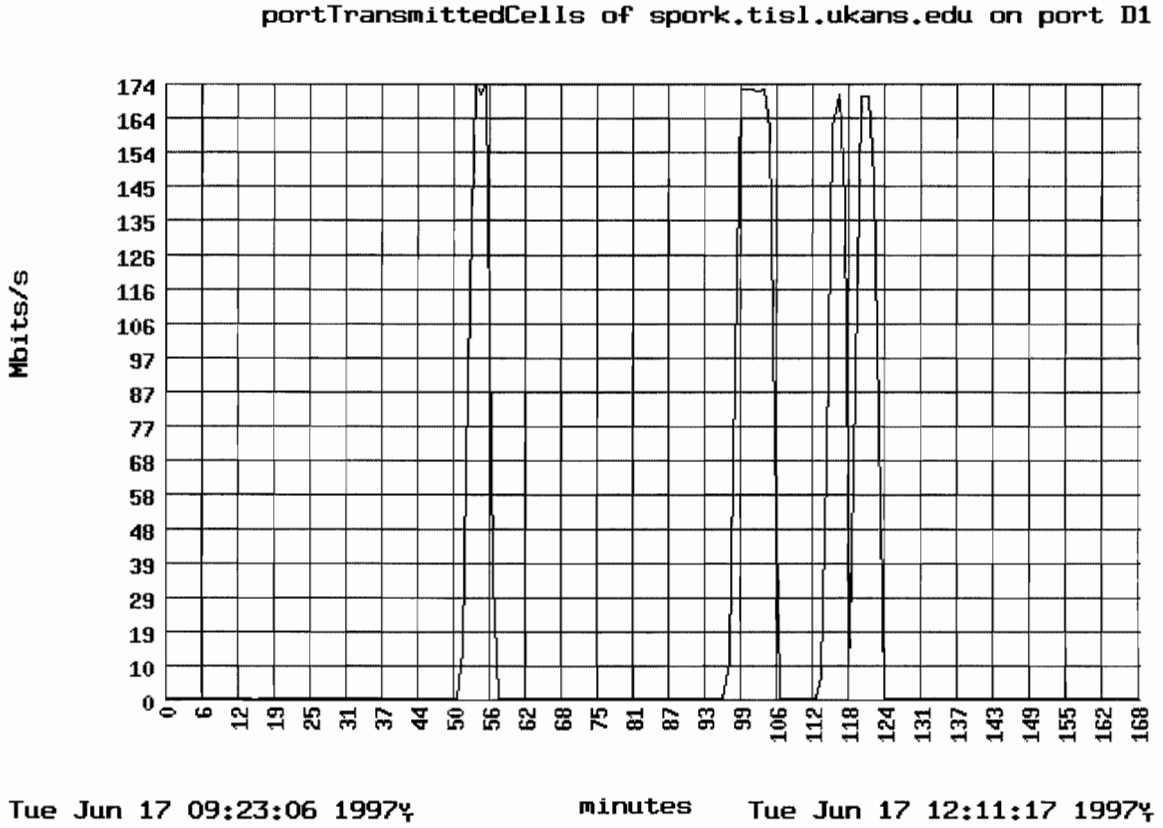


Figure 6.3. AAI Network Data Transmitted by KU to TIOC Over an OC-12 Link for ACTS Tests

During the months from May 1997 to September 1997, congestion experiments were performed (see Section 2). These tests were preformed in order to examine the impact

of background traffic on a TCP and UDP target flow in terms of throughput, segment interarrival time jitter and segment loss (for UDP). Cell Level Pacing was also experimented with on target flows and background traffic. Figure 6.4 shows a set of background traffic experiments. During the first 2.7 hours a set of experiments created data ranging from 30 Mbps to 55 Mbps were repeated a total of 6 times. The next six sets of experiments created data ranging from 65 Mbps to 90 Mbps. The data consisted of Netspec simulated streams of FTP, Videoconference, WWW, MPEG, and CBR traffic.

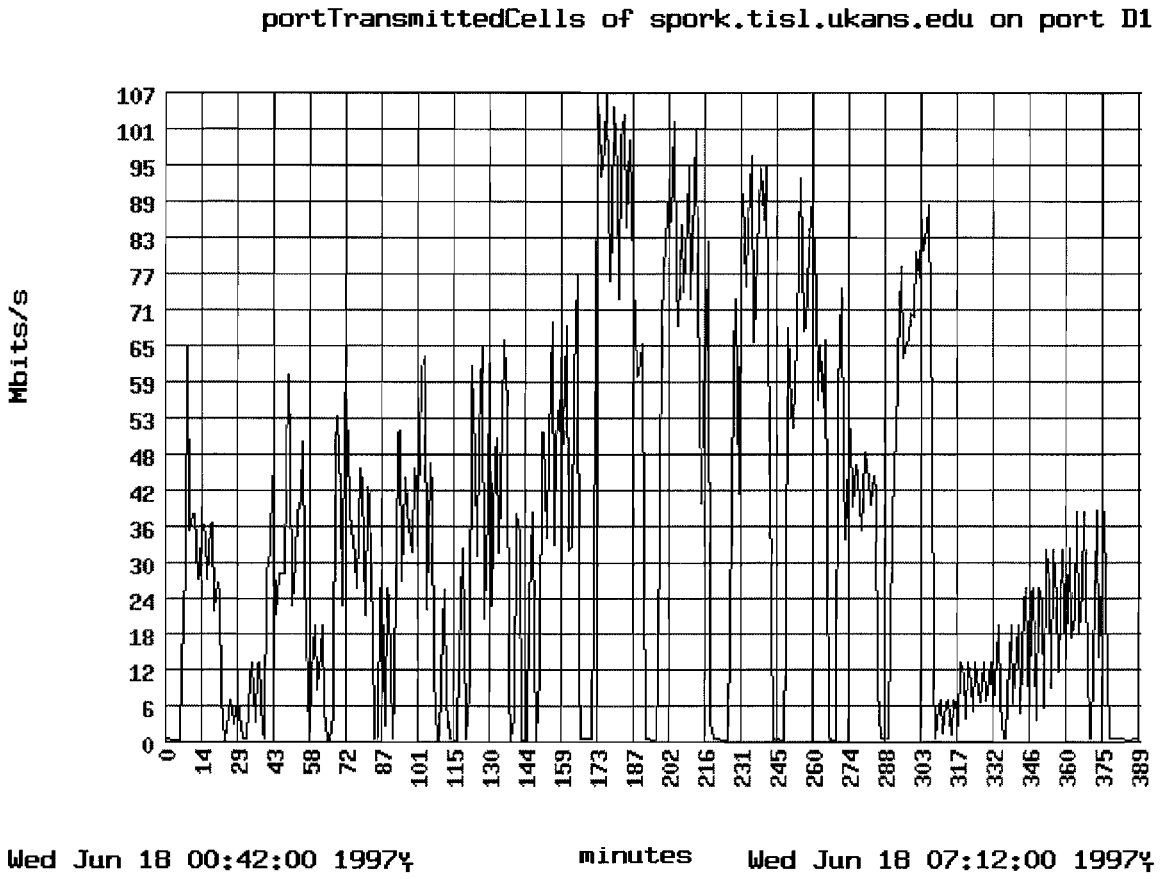


Figure 6.4. AAI Network Data TX by KU to TIOC Over an OC-12 Link

Several of the congestion experiments were repeated during October 1997 to determine any changes in the results after changes to the AAI network were completed. Some of the changes include the additions of edge switches, UBR traffic setting, and early packet discard (EPD) enabled. Figure 6.5 shows nine sets of background traffic experiments, which range in data from 65 Mbps to 90 Mbps. There was a considerable difference in results from the two different test times. The addition of edge switches, UBR, and EPD made a dramatic improvement.

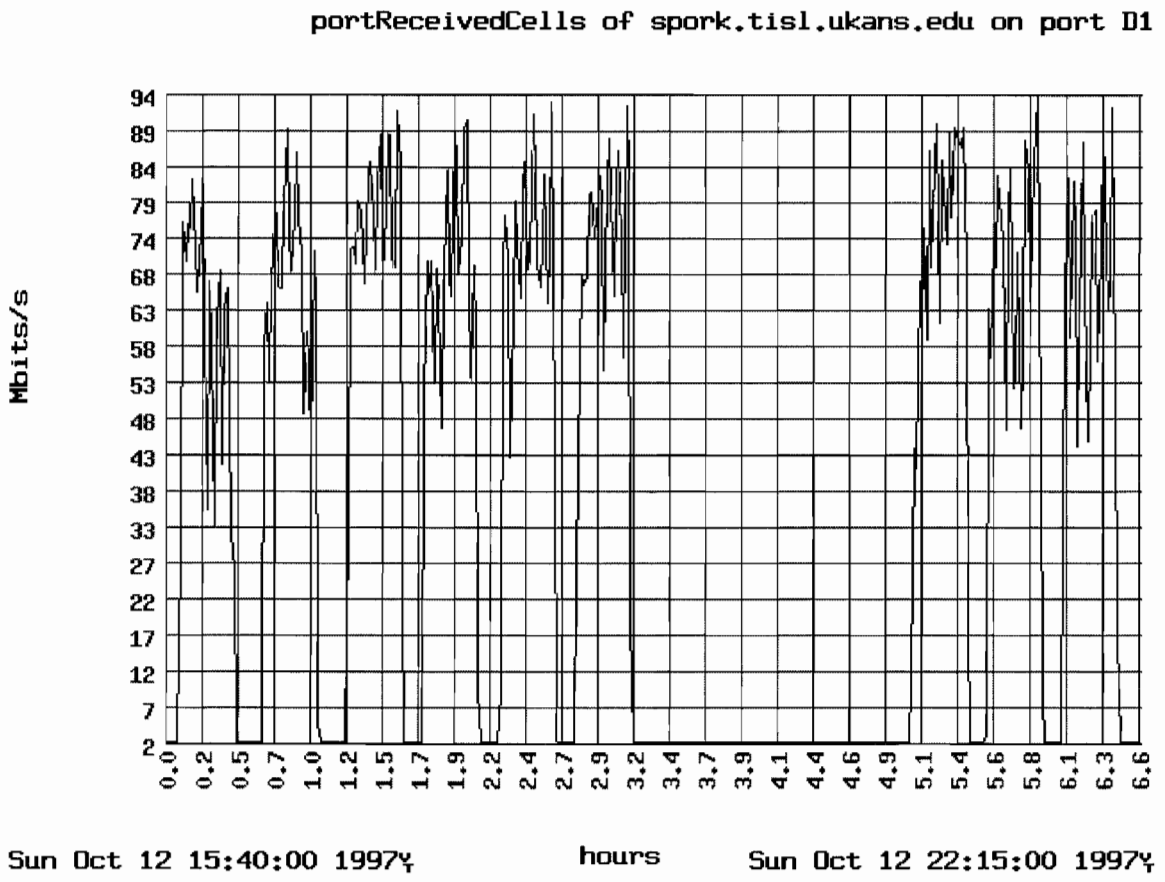


Figure 6.5. AAI Network Data Received from TIOC to KU Over an OC-12 Link

6.5 Initial Observations

Traffic flows at various locations in the AAI network were extensively monitored. Data, totaling more than 2.1 Gigabytes have been collected to date. In this section, some initial observations about the flow patterns observed in the AAI network are presented.

As expected of traffic in a broadband network, the AAI traffic flow patterns show significant burstiness. Bursts of varying lengths and magnitudes separated by periods of comparatively low throughput form a broad picture of the traffic flow in the AAI network. The extreme amount of burstiness is partly due to the traffic generated from the network experiments being done at different AAI hosts.

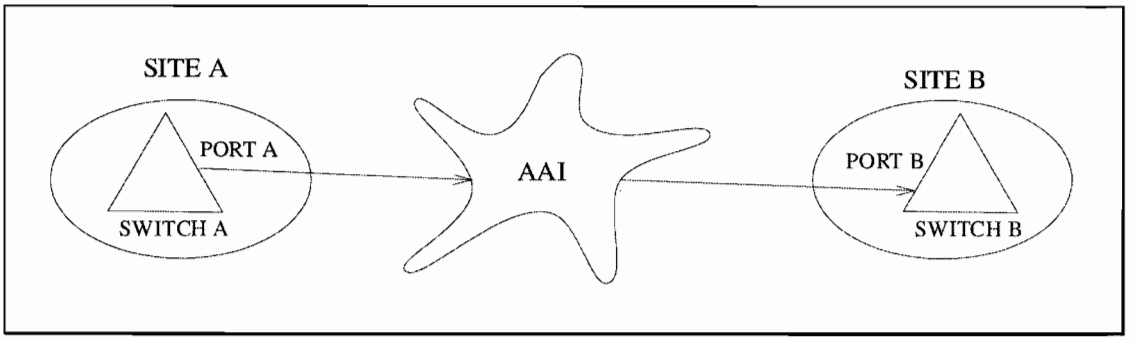


Figure 6.6. Flow Patterns at Ports A and B Show Similar Characteristics

In the plots of traffic flows, strong correlation are seen in the flow patterns on switches located at different sites, i.e. sources and destinations, in the AAI network. As shown in Figure 6.6 the traffic profiles at port A and port B on the switches at sites A and B, show similar characteristics. Sites A and B are typically the source and destination sites respectively for the data transfer.

6.6 NRL

Switch: aai-pop.nrl.aai.net (204.235.68.1)
Type: FORE asx200bx.
Hardware Version: 1.0
Software Version: S Fore thought 4.1.1(1.7)
Ports sampled: A2, A3, and B2.

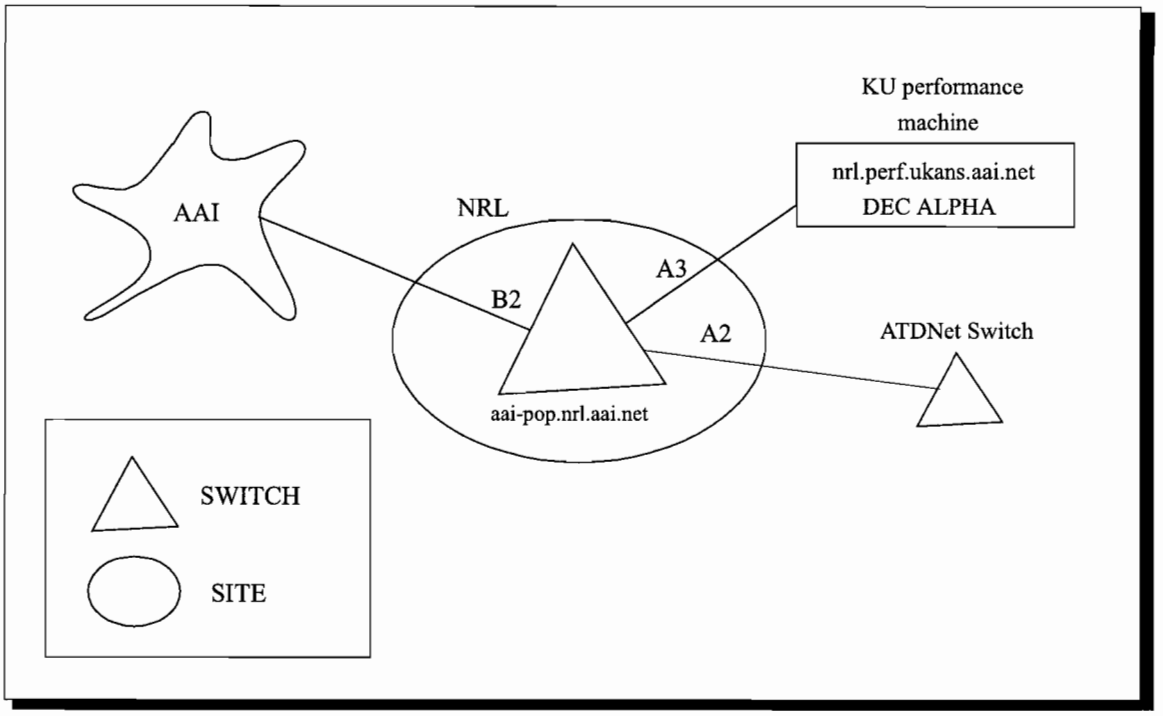


Figure 6.7. Connections to the FORE Switch aai-pop.nrl.aai.net at NRL

Figure 6.7 shows the topology for the NRL AAI network. In this section information about the ports sampled on the switch at NRL is given. KU has a performance machine connected to the switch aai-pop.nrl.aai.net at port A3. The machine is a DEC Alpha 3000/700 with IP number 204.235.68.3 and named aai-perf.nrl.aai.net. It is used for performing network experiments over the AAI network. Port B2 is an OC-3 link connecting NRL to the AAI network. Port A2 on the FORE switch aai-pop.nrl.aai.net is connected to the ATD net. Traffic flows associated with port A2 were the flows between hosts on the ATD net and the AAI network. Figure 6.8 shows the connections to the FORE switch at NRL. Figure 6.2 shows ATM traffic for the 1997 year received at NRL port B2 from the AAI network.

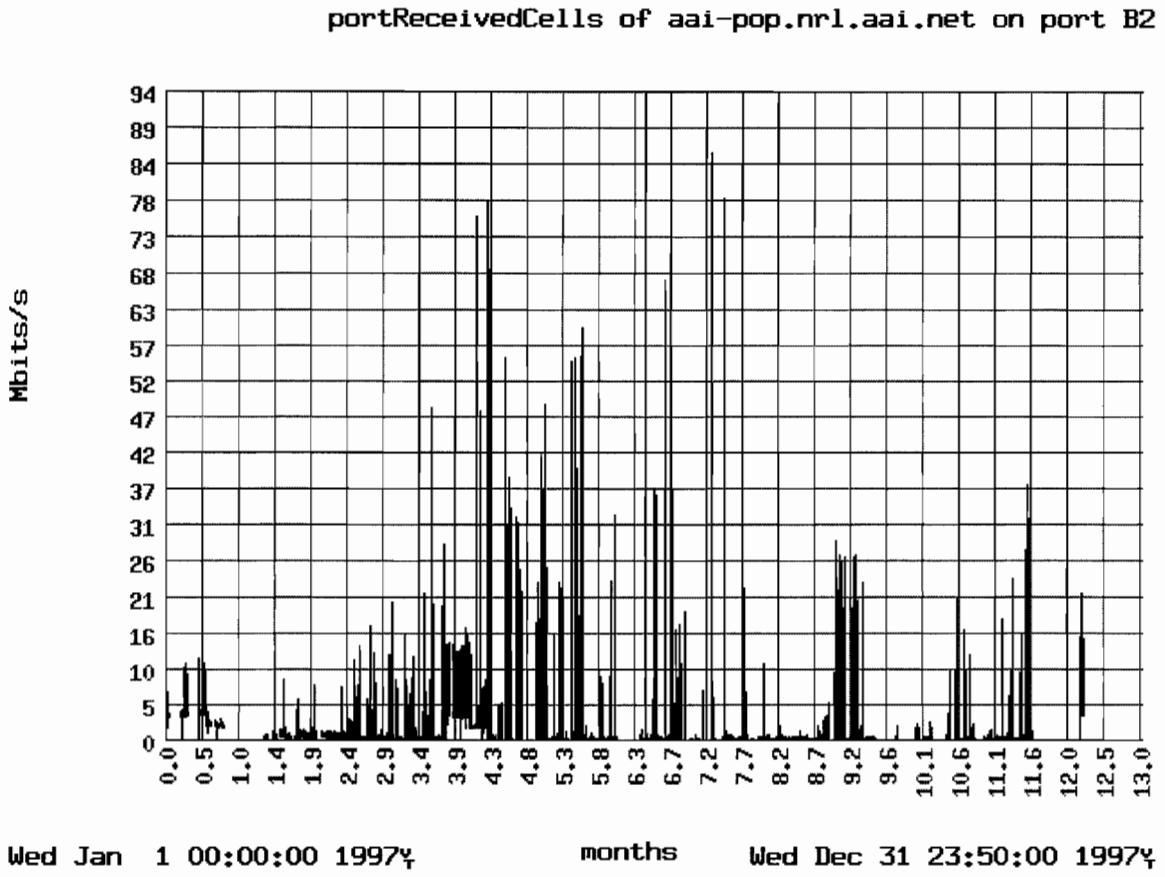


Figure 6.8. AAI Network Data Received by NRL During 1997

6.7. ARL

Switch: aai-pop-ether.arl.aai.net (128.63.58.58), aai-pop.arl.aai.net (204.235.65.1)
Type: FORE asx200.
Hardware Version: 1.0
Software Version: Fore thought 4.1.1(1.7)
Port Sampled: C4

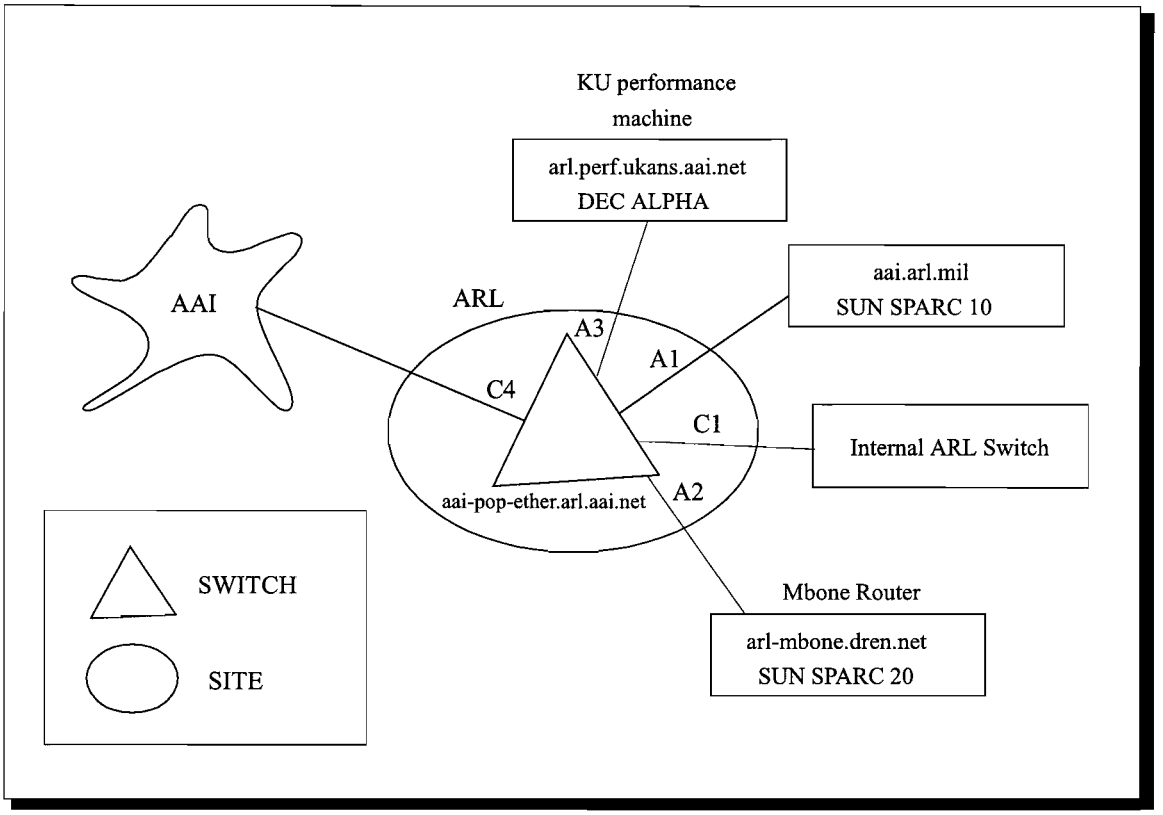


Figure 6.9. Connection to the FORE Switch aai-pop-ether.arl.aai.net at ARL

Figure 6.9 shows the topology for the ARL AAI network. In this section information about the ports sampled on the switch at ARL is given. KU has a performance machine connected to the switch aai-pop.arl.aai.net at port A3. The machine is a DEC Alpha 3000/700 with IP number 204.235.65.6 and named aai-perf.arl.aai.net. It is used for performing network experiments over the AAI network. Port C4 is an OC-3 link connecting ARL to the AAI network. The port currently sampled is port C4 that connects to the AAI network. Figure 6.10 shows the traffic received on port C4 from the AAI cloud. During September the ARL switch was upgraded to Fore thought 5.0 beta and the mib parameters did not match Fore thought 4.x. Since then the switch was returned to Fore thought 4.1.1(1.7) but our connectivity to the switch SNMP information has been unreliable.

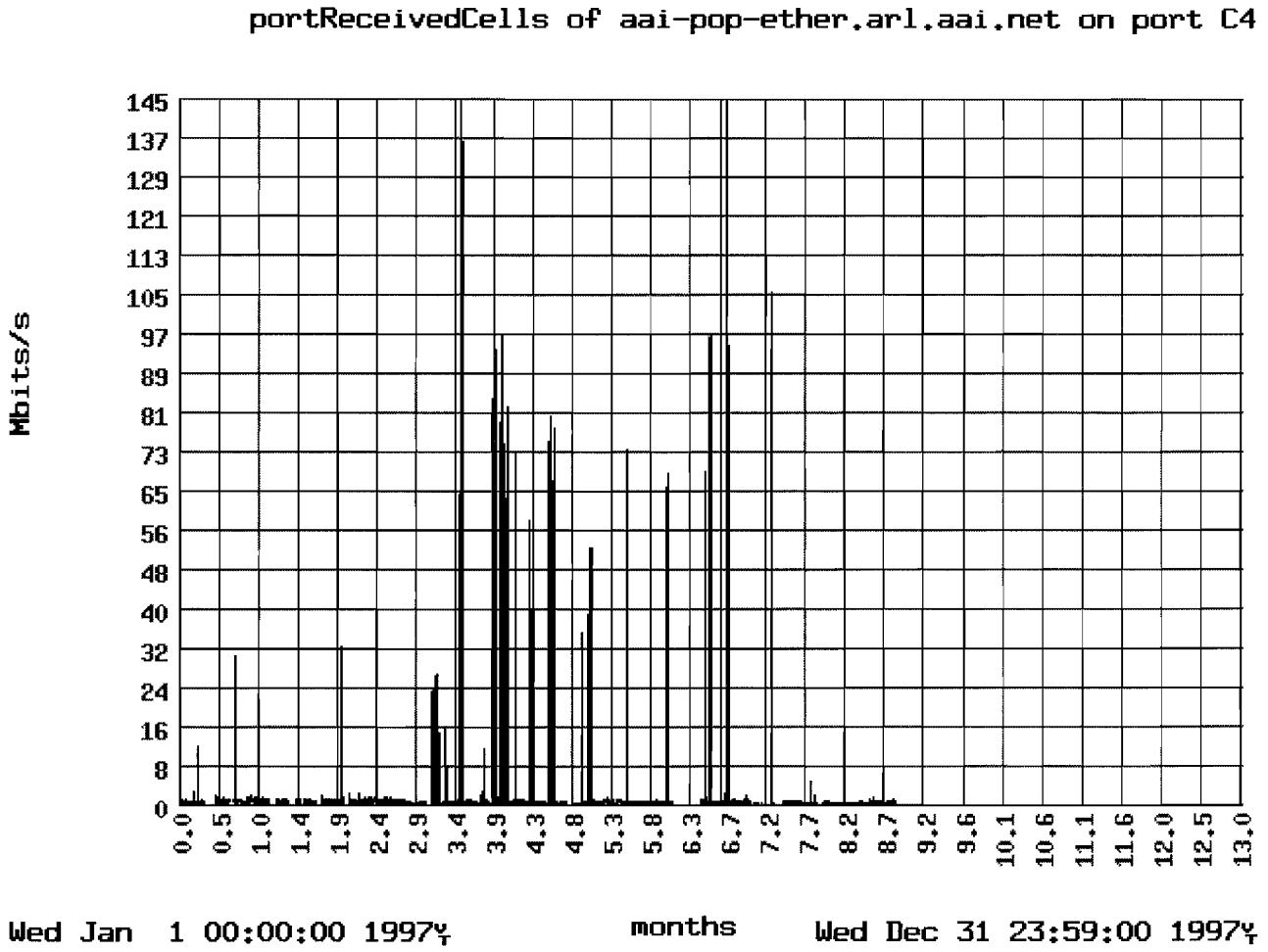


Figure 6.10. AAI network data received by ARL during 1997

6.8 NCCOSC

Switch: aai-pop.nccosc.aai.net (204.235.67.1), nrad-dren-tb.nosc.mil (128.49.4.184).
Type: FORE asx200bx.
Hardware Version: 1.0
Software Version: S Forethought 4.1.1(1.7)
Ports sampled: A1, A3, and C4

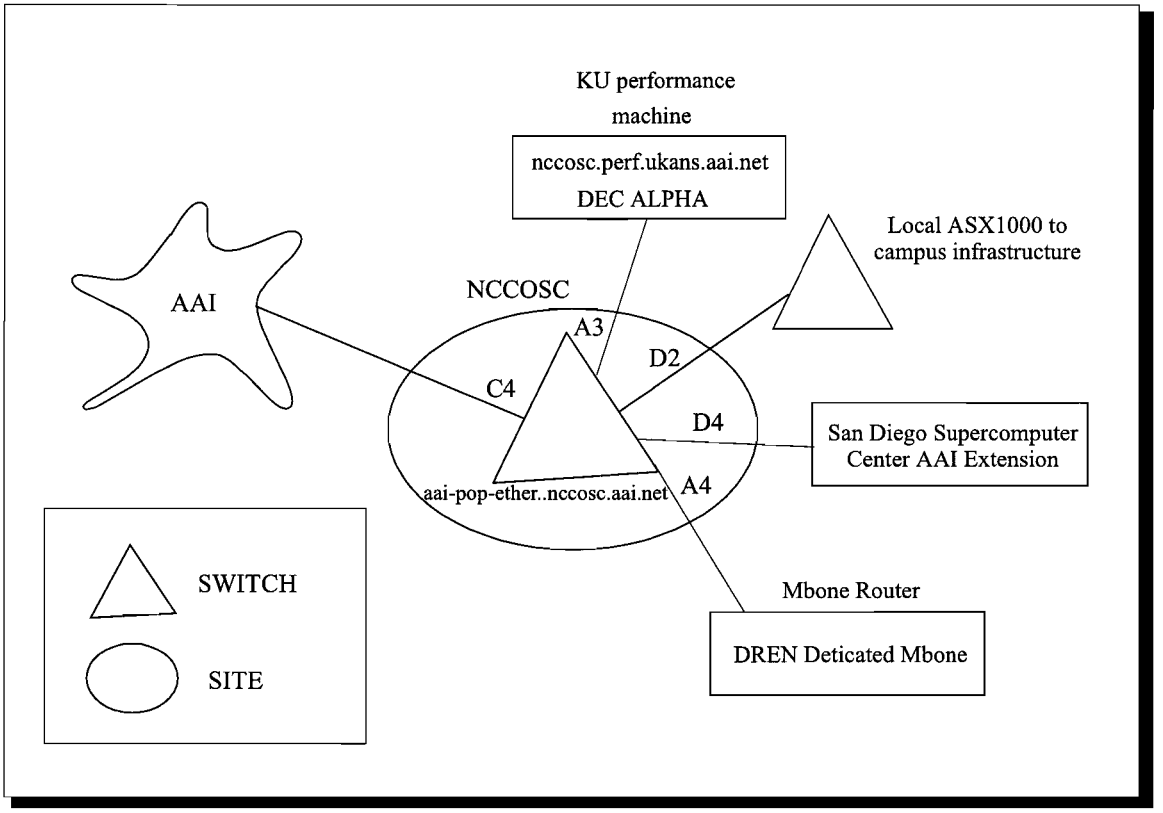


Figure 6.11. Connection to the FORE switch aai-pop-ether.nccosc.aai.net at NCCOSC

Figure 6.11 shows the topology for the NCCOSC AAI network. In this section information about the ports sampled on the switch at NCCOSC is given. KU has a performance machine connected to the switch aai-pop.nccosc.aai.net at port A3. The machine is a DEC Alpha 3000/700 with IP number 204.235.67.223 and named aai-perf.nccosc.aai.net. It is used for performing network experiments over the AAI network. Port C4 is an OC-3 link connecting NCCOSC to the AAI network. Port D2 is an OC-3 link connecting NCCOSC to their local Fore ASX1000 campus switch. Port D4 is connected to San Diego Supercomputer Center, an AAI extension. Port A4 is connected to a DREN dedicated Mbone multicast router. The ports currently sampled are A1, A3, and C4, which connect to the AAI network. Figure 6.12 shows the traffic received on port A3 from the KU performance machine.

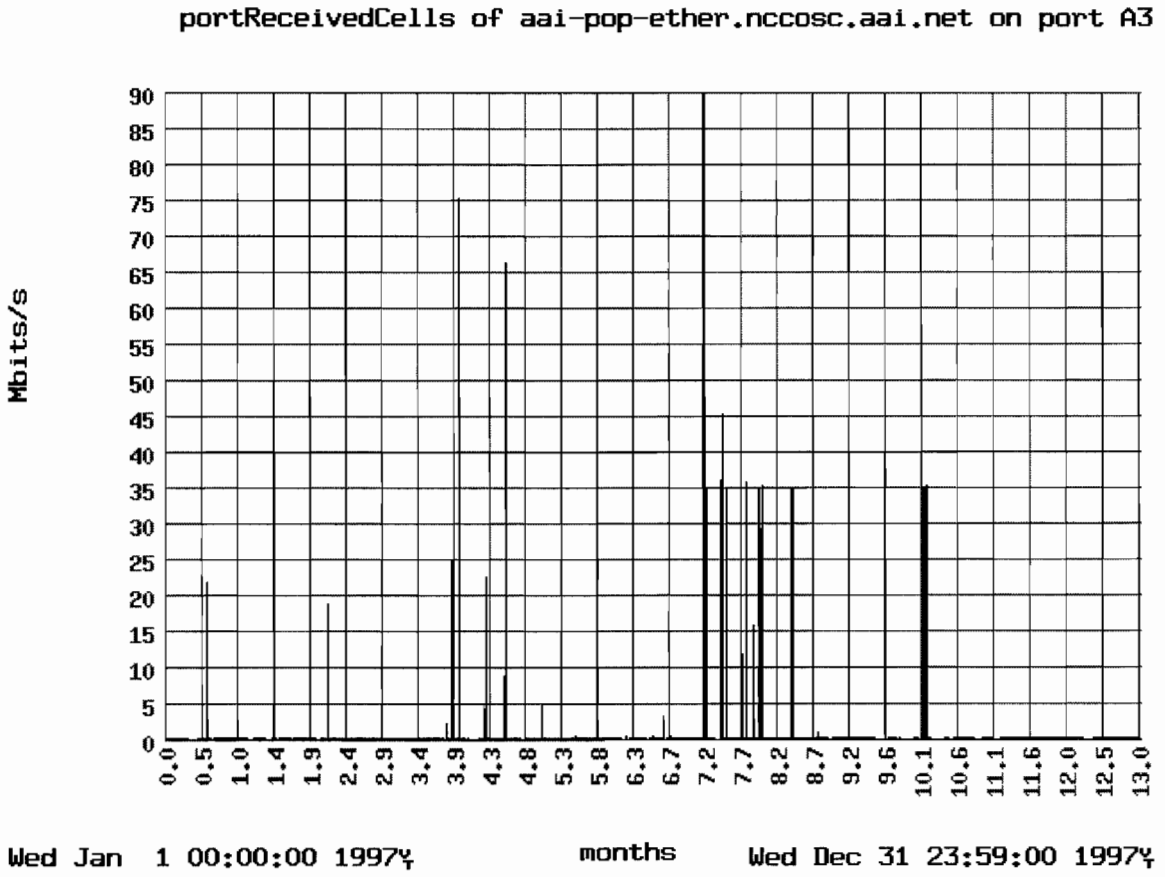


Figure 6.12. AAI Network Data Received from nccosc.perf.ukans.aai.net During 1997

6.9 CEWES

Switch: aai-pop-ether.cewes.aai.net (134.164.124.5)
Type: FORE asx200.
Hardware Version: 1.0
Software Version: S Fore thought 4.0.3(1.6)
Port sampled: B1

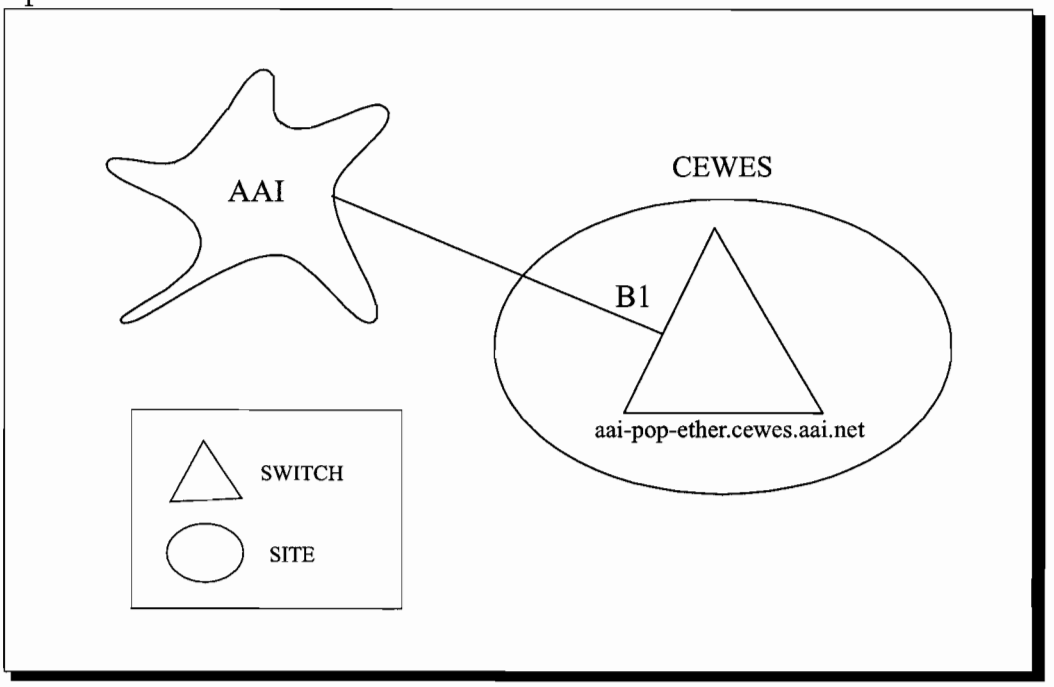


Figure 6.13. Connection to the FORE Switch aai-pop-ether.cewes.aai.net at CEWES

The monitored port B1 is the port connecting to the AAI network, which is shown in Figure 6.13. In Figure 6.14 the traffic received from the AAI network to the CEWES switch is shown. Sometime during July 1997, we lost SNMP access to the CEWES switch and therefore could not monitor the network traffic at this switch.

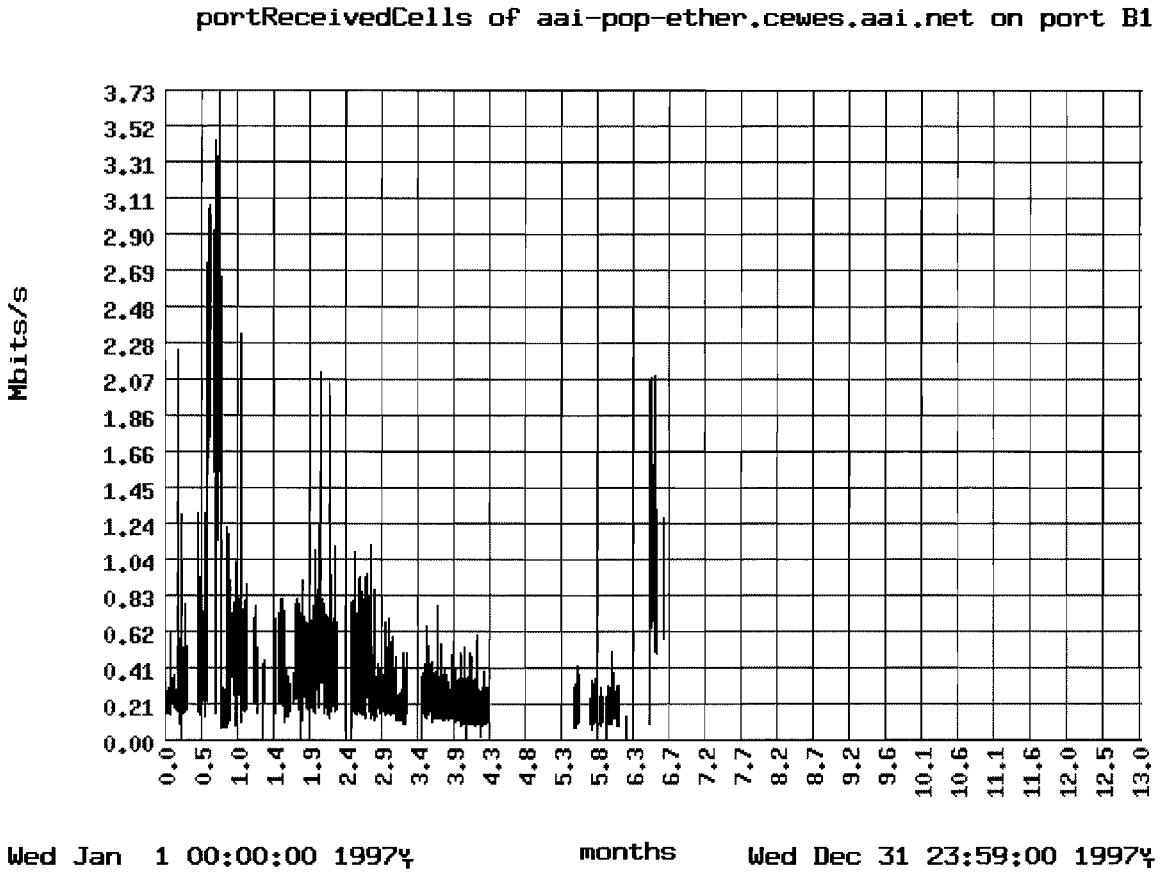


Figure 6.14. AAI Network Data Received by CEWES During 1997

6.10 NRLSSC

Switch: aai-pop-ether.nrlssc.aai.net (128.160.10.111)
Type: asx200.
Hardware Version: 1.0
Software Version: S Fore thought 4.0.0(1.35)
Port sampled: B1

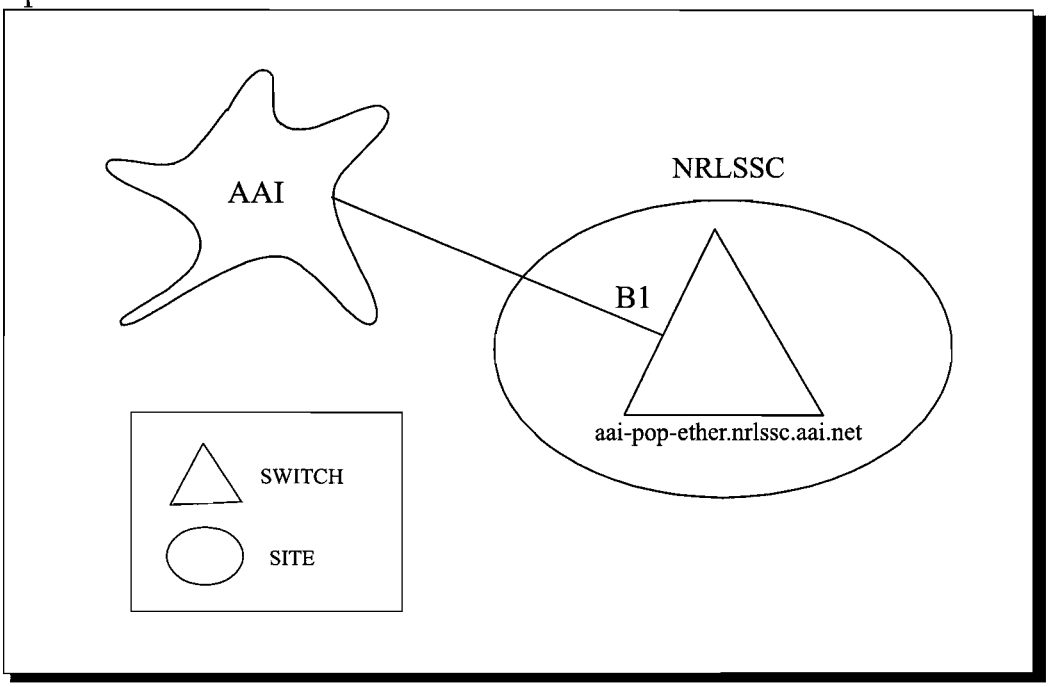


Figure 6.15. Connection to the FORE Switch aai-pop-ether.nrlssc.aai.net at NRLSSC

The monitored port B1 is the port connecting to the AAI network, which is shown in Figure 6.15. Very little data was sampled from the NRLSSC switch due to unknown problems.

6.11 EDC

Switch: merlin.edc.magic.net (128.49.4.184).
Type: asx200bx
Hardware Version: 1.0
Software Version: S Forethought 4.1.1 (1.7)
Ports currently sampled: A1, A2, and B2

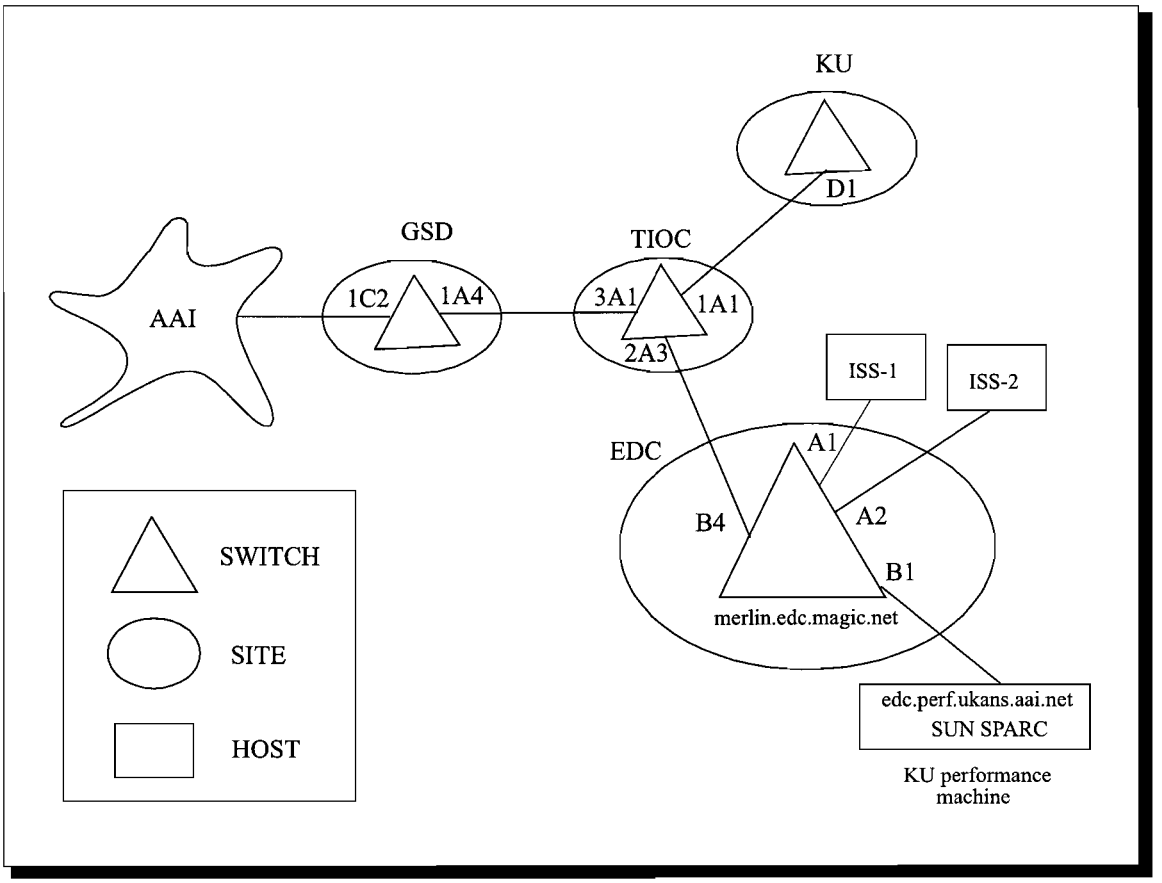


Figure 6.16. Connection to the FORE Switch merlin.edc.magic.net at EDC

The monitored ports A1 and A2 are connected to ISS machines, ISS-1 and ISS-2 respectively. The other monitored port B4 is connected to TIOC.

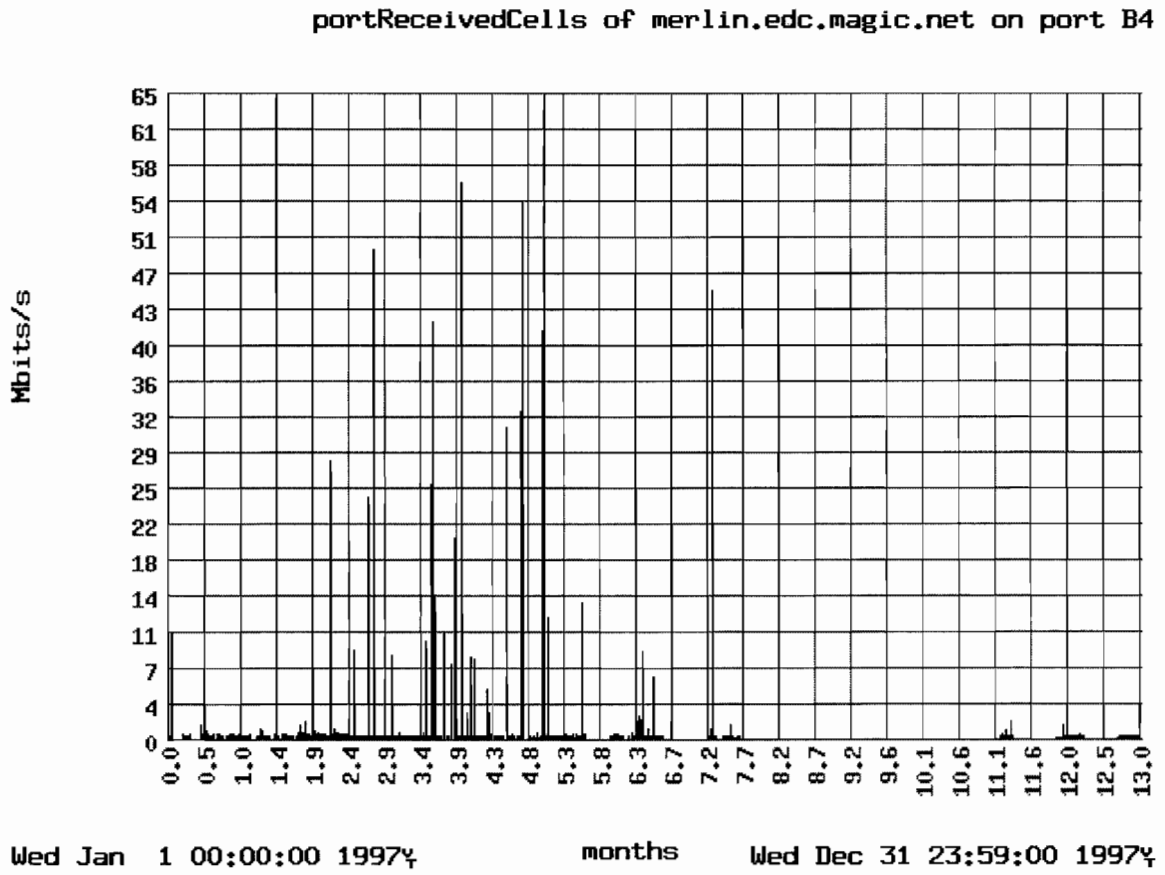


Figure 6.17. AAI Network Data Received by EDC During 1997

The traffic profiles generated from these experiments are also noted in the flow patterns into port A1 on the FORE switch at TIOC. This is expected from the connections shown in Figure 6.16. A SPARC station owned by KU is connected to the FORE switch MERLIN at this site. The performance evaluation machine at this site is edc.perf.ukans.aai.net and its IP number is 204.235.71.132.

6.12 GSD

Switch: aai-pop-ether.gsd.aai.net (192.157.66.194)
Type: asx200bx
Hardware Version: 1.0
Software Version: S Forethought 4.1.1(1.10)
Port sampled: B1, C2.

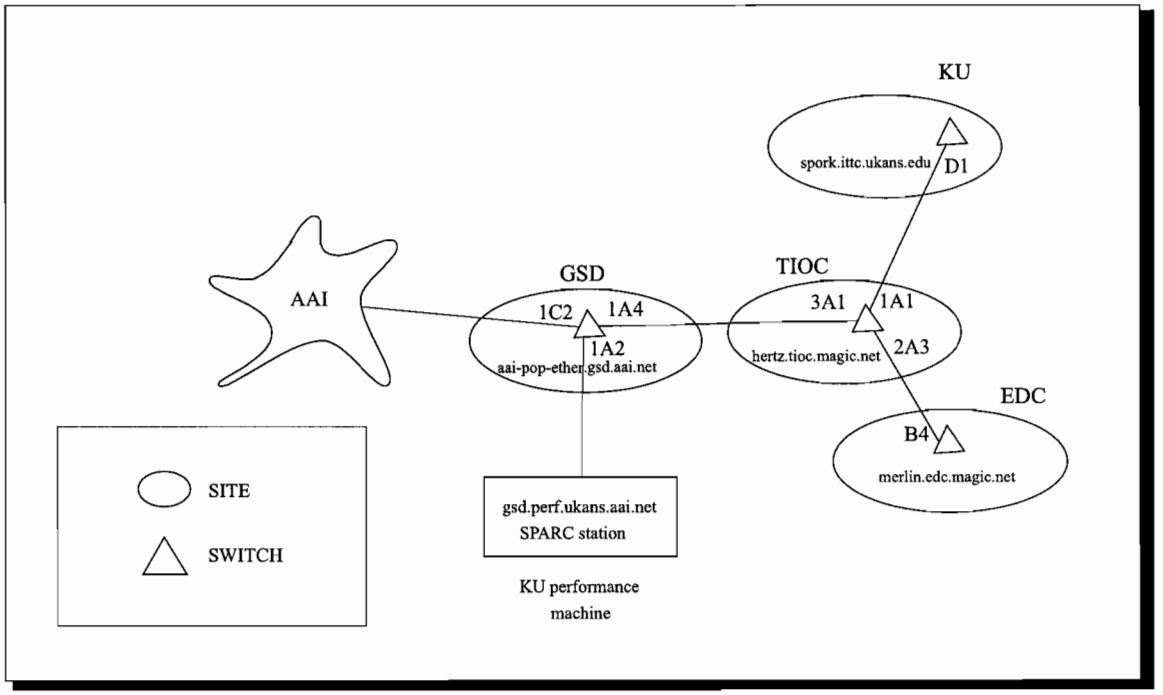


Figure 6.18. Connection to the FORE Switch aai-pop-ether.gsd.aai.net at GSD

Connections to the monitored ports on the FORE switch at SPRINT GSD are shown in Figure 12.1. The monitored port 1C2 was connected to the AAI network. On port 1A2 KU has a Sun SPARCstation, gsd.perf.ukans.aai.net with an IP number 204.235.71.3 that is located at this site for doing performance evaluation experiments.

6.13 TIOC

Switch: hertz1-uni.tioc.magic.net (198.207.143.241)
Interface: hertz1.tioc.magic.net (198.207.141.241)
Type: asx1000
Hardware Version: 1.0
Software Version: S ForeThought 4.1.1 (1.7)
Port sampled: A1.
Interface: hertz2.tioc.magic.net
Type: asx1000

Hardware Version: 1.0
Software Version: S ForeThought 4.1.1 (1.7)
Ports sampled: A3, C1, and C2

As can be seen from Figure 6.19 traffic patterns on port 3A1 represent the flows to and from the AAI network. The EDC site is connected to port 2A3 and the KU site at port 1A1. The rest of the AAI was connected through the port 3A1. Various connections on the monitored ports are shown in Figure 6.19.

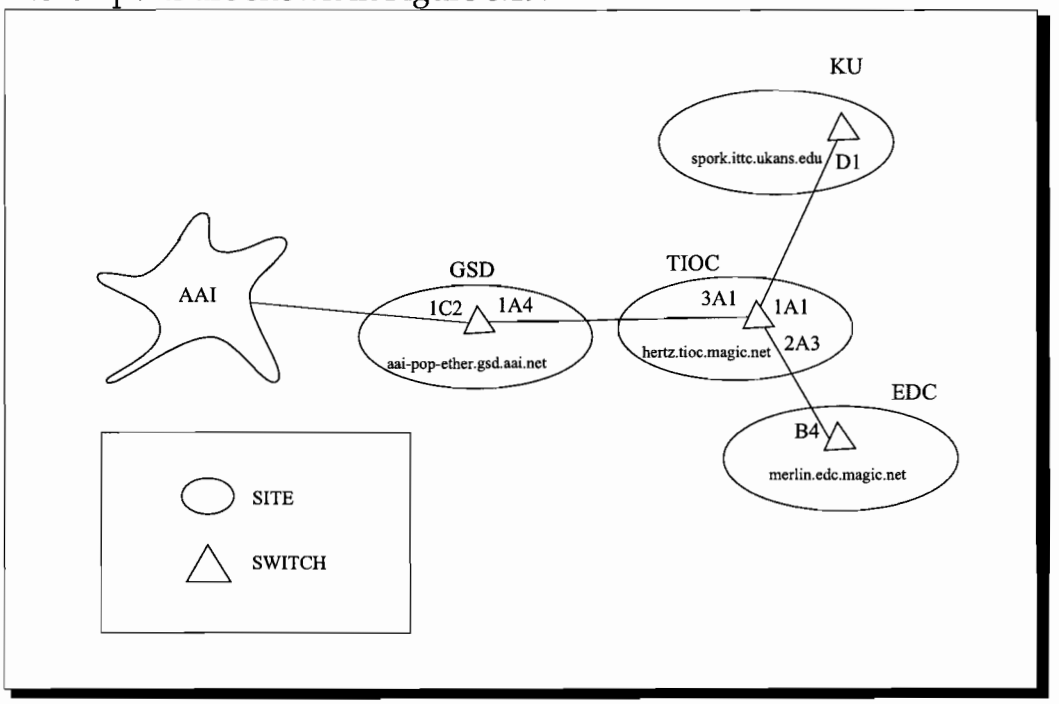


Figure 6.19. Connection to the FORE Switch hertz.tioc.magic.net at TIOC

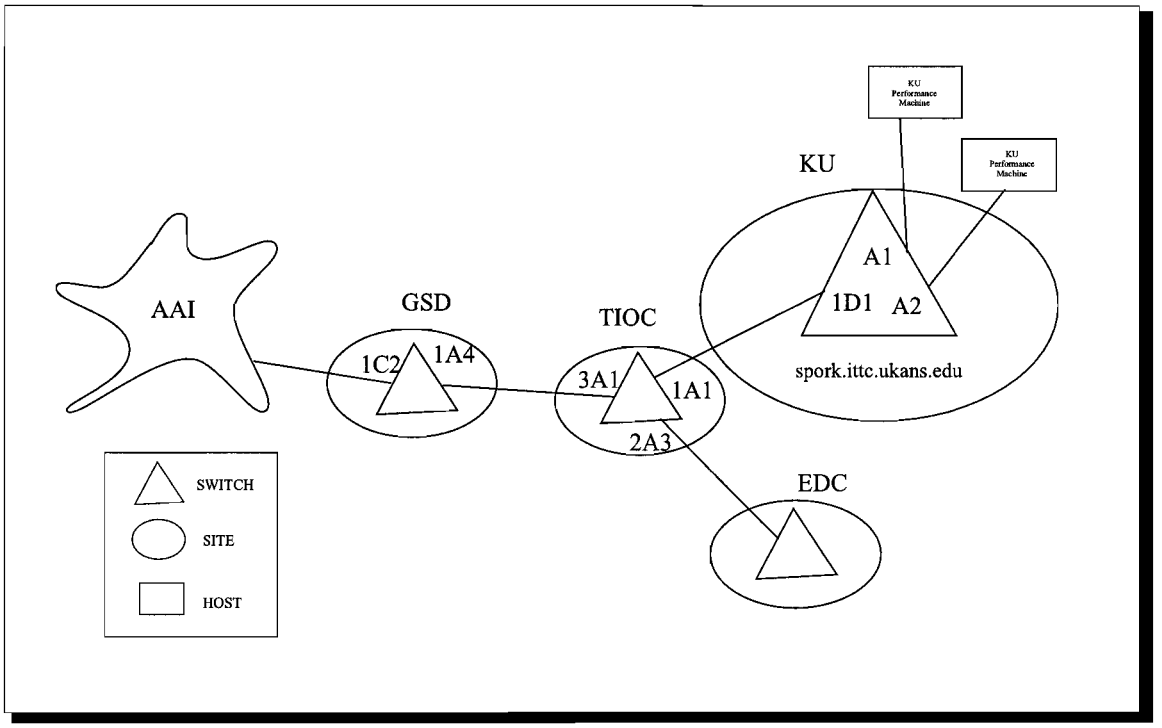


Figure 6.21. Connection to the FORE Switch spork.tisl.ukans.edu at KU

Throughout the year, a number of experiments to study ATMWAN performance were conducted by KU using Netspec. The traffic from the experiments is seen as bursts of short duration.

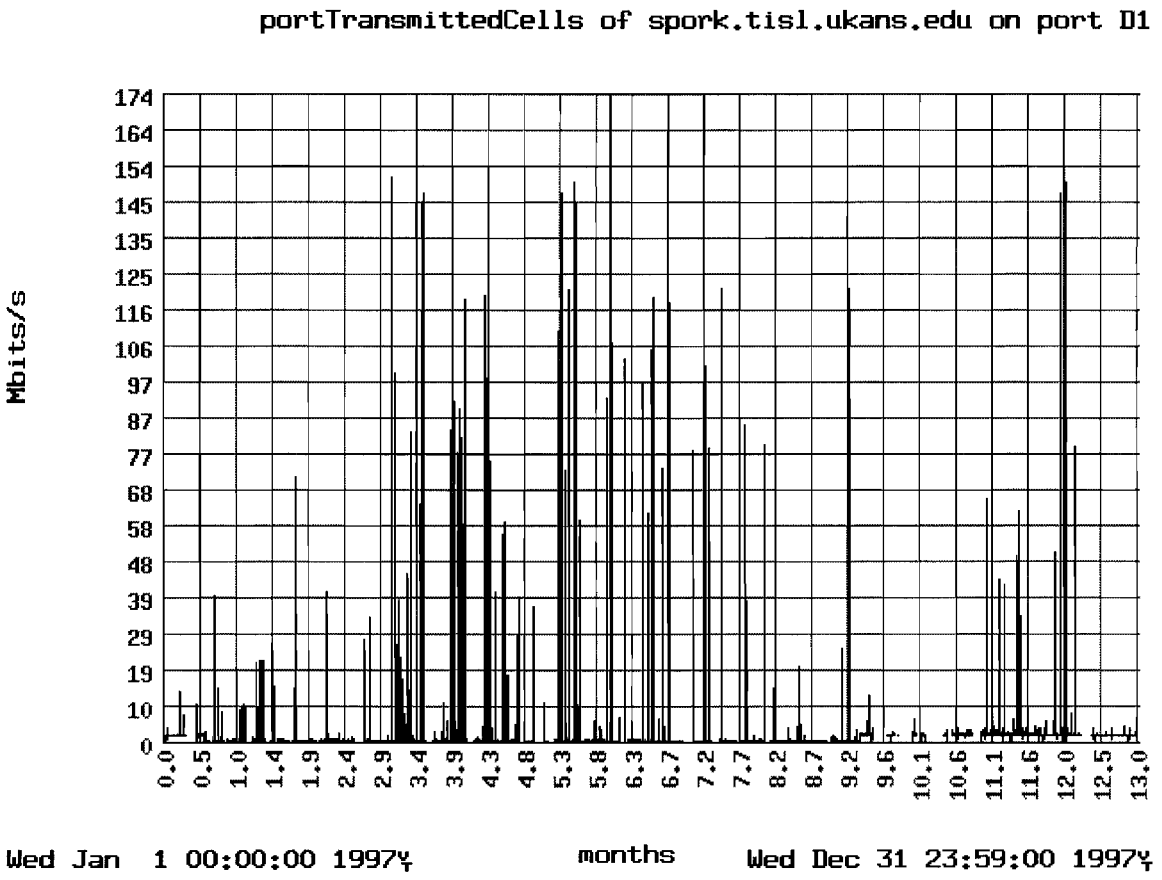


Figure 6.22. AAI Network Data Transmitted by KU During 1997

6.15 Conclusions

During the 1997-year of the ACTS ATM Internetwork (AAI) project, traffic flows at various locations in the AAI network were extensively monitored. Data, totaling to more than 2.1 Gigabytes, was collected to date. Bursty traffic profiles with a wide scale of burst lengths are observed in the AAI network. Bursts of duration varying from a minute to over an hour are seen in the traffic flow plots. Smaller bursts riding on larger bursts are commonly seen in many of the traffic plots. The expected correlation in the traffic profiles observed at different sites is characteristic in the AAI network. A significant amount of management traffic was observed in collected data flows. The plots presented in this document can be used as references of traffic profiles between various sites on the AAI network.

6.16 References

[1] S. Muppidi, V. S. Frost, "Traffic Flow patterns in the AAI Network: January 1996 to December 1997", ITTC Technical Report, ITTC-TR-10980-18.

[2] Dirk Wisse, Jan Voorschot, Tricklet V6.0, ftp://dnpap.et.tudelft.nl/pub/btng

[3] Roelof Jonkman, NetSpec, http://www.ittc.ukans.edu/Projects/AAI/products/netspec/.

[4] B. O. Lee, V. S. Frost, "Wide Area ATM Network Experiments using Emulated Traffic Sources", ITTC Technical Report, ITTC-FY98-TR-10980-24.

[5] M. D. Linhart, V. S. Frost, "ATM Background Traffic Impact on UDP Packets", ITTC Technical Report, ITTC-FY98-TR-10980-26.

[6] T. Blackman, KU AAI Data Plotter, http://www.ittc.ukans.edu/Projects/AAI/

[7] Fore Systems, Fore Switch ASX200BX/ASX200BXe User's Manual. From Mike's report, delete the abstract.

7.0 AAI Related Publications

The following are published works related to our AAI research.

IEEE Communications Magazine, Feature Topic on "Performance Experiences with Wide-Area High Speed Networks," August 1997 (with Donald L. Endicott Jr)

H. Zhu, L.A. DaSilva, J.B. Evans, and V.S. Frost, "Performance Evaluation of Congestion Control Mechanisms in ATM Networks," 20th International Conference of the Computer Measurement Group, Nashville, TN, December 3-8, 1995.

G.Y. Lazarou, V.S. Frost, J.B. Evans, and D. Niehaus, "Using Measurements to Validate Simulation Models of TCP/IP over High Speed ATM Wide Area Networks," IEEE International Conference on Communications, Dallas, TX, June 1996.

K. Liu, H. Zhu, D.W. Petr, V.S. Frost, C. Braun, and W. Edwards, "Design and Analysis of a Bandwidth Management Framework for ATM-Based Broadband ISDN," IEEE International Conference on Communications, Dallas, TX, June 1996.

L. DaSilva, Rick Lett, and V.S. Frost, "Performance Considerations in File Transfes Using FTP Over Wide-Area ATM Networks", 22nd International Conference for the Resource Management and Performance Evaluation of Enterprise Computing Systems, San Diego Cal, Dec. 8-13 1996.

Luiz DaSilva, J. Evans, D. Niehaus, V. Frost, R. Jonkman, B. Lee, G. Lazarou, "Performance Experiences in a Wide Area ATM Network" IEEE International Conference on Communications, Montreal, Canada, pp 1694-1698, June 1997

Charalambous P. Charalambos, Georgios Y. Lazarou, Victor S. Frost, Joseph Evans,and Roelof Jonkman, "Experimental and Simulation Performance Results of TCP/IP over High-Speed ATM over ACTS", accepted for publications at " IEEE International Conference on Communications 98.

L. Dasilva, J. B. Evans, D. Niehaus, V. S. Frost, R. Jonkman, B. Lee, G. Lazarou, "ATM WAN Performance Tools, Experiments, and Results," IEEE Communications Magazine, Vol 35, No. 8, vol 35 August 1997, pp 118-125.

K. Liu, H. Zhu, D.W. Petr, V.S. Frost, C. Braun, and W. Edwards, "Design and Analysis of a Bandwidth Management Framework for ATM-Based Broadband ISDN," IEEE Communications Magazine, Vol 35, No. 5, May 1997, pp 138- 145.

Georgios Y. Lazarou, Victor S. Frost, Joseph B. Evans, Douglas Niehaus, "Simulation & Measurement of TCP/IP over ATM Wide Area Networks" Accepted for publication in IEICE Transactions on Communications special issue on ATM Switching Systems for Future B-ISDN.

S. Muppidi and V. Frost, "A Simple Model and Performance Evaluation Methodology for ATM Queues", Submitted to *IEEE/ACM Transactions on Networking.*

APPENDIX A

Wide Area ATM Network Experiments Using Emulated Traffic Sources

Beng-Ong Lee and Victor S. Frost

Technical Report ITTC-FY98-TR-10980-24

APPENDIX B

Modeling and Analysis of Traffic in High Speed Networks

Soma S. Muppidi and Victor S. Frost

Technical Report ITTC-FY98-TR-10980-22

APPENDIX C

Experiments and Simulations of TCP/IP Over ATM Over a High Data Rate Satellite Channel

Charalambous P. Charalambos, Georgios Y. Lazarou, Victor S. Frost, Joseph Evans and Roelof Jonkman

Technical Report ITTC-FY98-TR-10980-25

APPENDIX D

NetSpec: Philosophy, Design and Implementation

Roelof J.T. Jonkman and Joseph B. Evans

Technical Report ITTC-FY98-TR-10980-28