FINAL REPORT

Colorado Advanced Software Institute

ATMROS: A Network Design System for ATM Network Optimization and Traffic Management

Principal Investigator:	C. Edward Chow, Ph.D. Associate Professor University of Colorado, Colorado Springs
Graduate Student:	Albert M. Schaffer Department of Computer Science University of Colorado, Colorado Springs
Collaborating Company:	US WEST Advanced Technologies Steve Y. Chiu, Ph.D. Member of Technical Staff

PROJECT TITLE:	*ATMROS:* A Network Design System for ATM Network Optimization and Traffic Management
PRINCIPAL INVESTIGATOR:	C. Edward Chow, Ph.D.
UNIVERSITY:	University of Colorado, Colorado Springs
COLLABORATING COMPANY:	US WEST Advanced Technologies
REPRESENTATIVE OF COLLABORATING COMPANY:	Steve Y. Chiu, Member of Technical Staff

As authorized representative of the collaborating company, I have reviewed this report and approve it for release to the Colorado Advanced Software Institute.

SIGNATURE:

DATE:

Table of Contents

Abstract

This project deals with the development of algorithms and tools to facilitate efficient use of Asynchronous Transfer Mode (ATM) network resources and satisfy the customer's requirements for quality of service (QoS). A software design system called ATMROS was built to characterize the network traces, identify the hot regions, feed the hot regions to a simulator that simulates the ATM network, and analyze the traffic at the destination for QoS. ATMROS consists of a set of PERL scripts that plot a network trace at various time scales, scan a trace for hot regions based on a selected burstiness metric, including the Hurst parameter, and extract selected regions of a trace file. The extracted trace region can be read by a modified NIST ATM simulator, which can be configured to simulate the process of a VP connection on a ATM network. The ATM simulator generates the log file which captures the timing and pattern of ATM cells arrived at the destination of the VP connection. A QoS script was written to analyze the log file and report the QoS statistics of the selected trace over the ATM connections.

A set of burstiness metrics was proposed and their effectiveness on a LAN trace, OctExt.TL from Bellcore, was analyzed. It was found that the Hurst constant, H, is not an effective metric for identifying local bursty regions in a network trace. Local bursty regions can be identified effectively using either metric called "Peak/Average with floor" or metric called "variance with floor".

By improving the modified NIST ATM simulator for longer range simulation and integrating efficient search routine, ATMROS can be used to suggest cost-effective bandwidth for a network customer subscribing an ATM network service. The scripts developed can be used to analyze LAN and other media type traces. They augment the library of tools that can foster the research, education, and development efforts in the area of network resource optimization and traffic management.

ATMROS: A Network Design System for ATM Network Optimization and Traffic Management

1. Introduction

The context of the problem which this project addresses is how a network service provider such as US West or MCI can assign network bandwidth to a customer prospect, based on an understanding of typical network traffic originating with that customer. If the traffic under consideration was “telephone traffic”, the assignment of bandwidth is straightforward. In fact, many of the existing network formats were designed specifically with telephone traffic in mind. For example, a T1 (or DS1) network connection can support a single telephone call on each of its 24 channels. For various reasons, which will be described in the following sections, “computer generated network traffic” represents a formidable challenge to both network service provider and the network customer. In particular, the “bursty” nature of computer generated traffic complicates the assignment of network bandwidth. Furthermore, the customers expectations on such things as “information loss” or “delay” of that information as it traverses the network can be difficult to satisfy. This project focuses specifically on ATM network traffic, which is described in the next section. Important terms are defined later in this section.

1.1. Overview of Asynchronous Transfer Mode

Asynchronous Transfer Mode (ATM) could become the dominant means of transporting Wide Area Network (WAN) traffic within the next several years. It offers the flexibility to carry any type of traffic, whether constant bit rate (CBR), variable bit rate (VBR), or best efforts, i.e. available bit rate (ABR). ATM features cells having a fixed size of 53 bytes, including a 5 byte header and a 48 byte payload. Traffic is first chopped or "segmented" into cells at the sending end, and after being transported across the network, are "reassembled" at the receiving end.

Customer traffic, whether from a single customer, or the multiplexed aggregate from a number of customers, enters an ATM network via one of several input ports on an ***n x m*** ATM *switch*. In order to avoid swamping the network with traffic, it is reasonable to assume some sort of queuing structure as part of the ATM switch, router or similar equipment where traffic enters the network. ATM switches generally are constructed with identically sized queues on each output (and/or input) port. Conceptually, Figure 1 shows the traffic from multiple customers feeding into a common ATM switch.

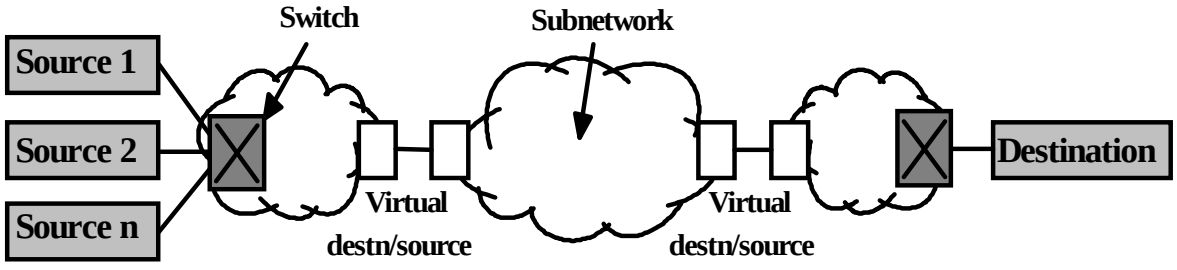


Figure 1. End-to-end ATM network with multiplexed sources to a common switch.

1.2. The Project Problem Further Defined

As an example, suppose we are given an 8-port ATM switch with traffic of 6 Mb/sec on each those ports. One would expect that the ATM switch should have a capacity of 48 Mb/sec. This is probably in the right ball park, but things are much more complicated due to the inherently bursty nature of network traffic. The primary function of an ATM switch is to combine or multiplex the traffic from all input ports to one or more output trunks. If traffic bursts from several input ports occur at the same instant, the switch may be overwhelmed with resultant loss of traffic (ATM cells). From the customer's standpoint, such a loss would result in a poor Quality of Service.

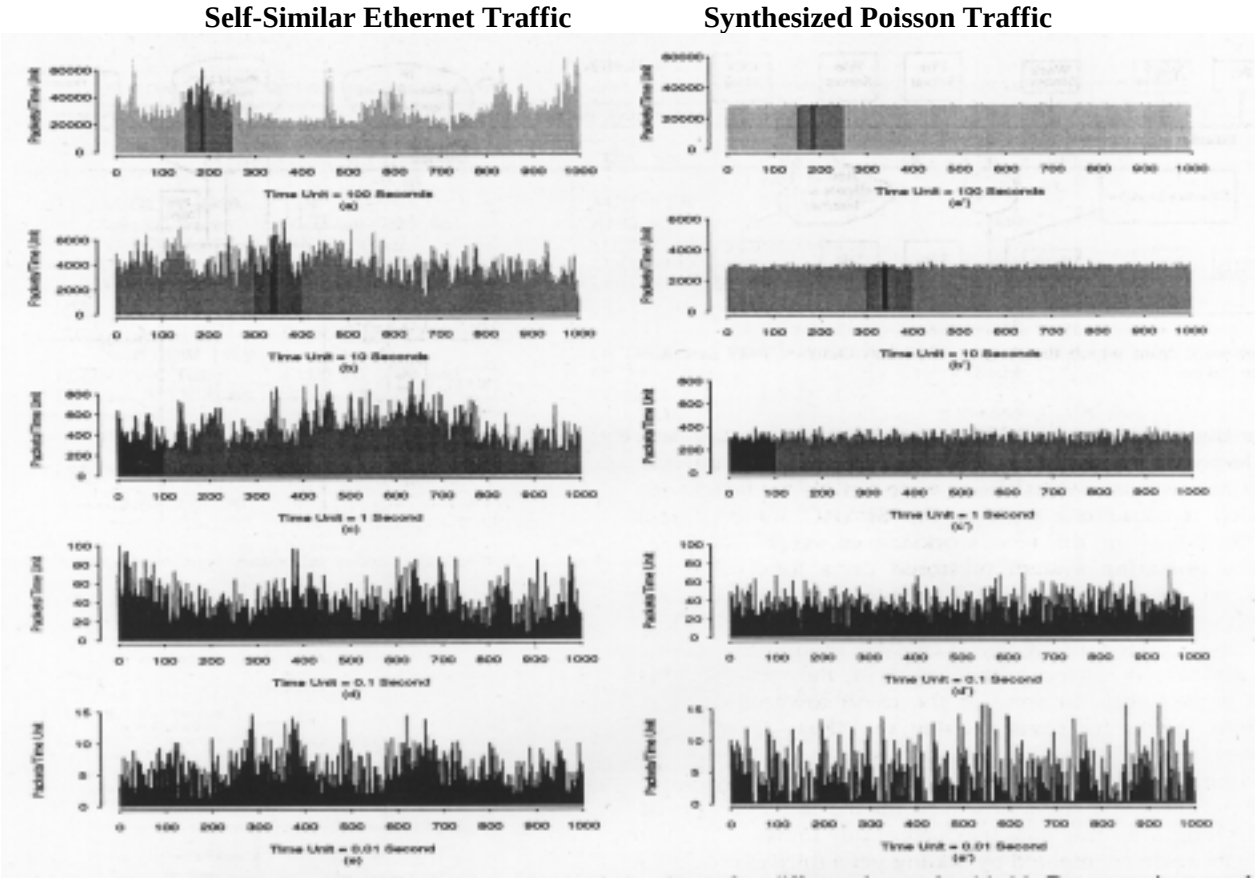


Figure 2. Actual self-similar traffic vs. fictitious (synthesized) Poisson network traffic.

Characterizing the "burstiness" of ATM traffic is a particularly sticky challenge. Historically, network traffic had been assumed to obey a Poisson distribution. That is, traffic bursts are assumed to start at random times, and message traffic is assumed to be of arbitrary length. However, actual network burstiness is worse than Poisson, and is characterized variously as being *self-similar*, *heavy-tailed*, or *long-term*. This discovery was the result of recent (1993) research at Bellcore [Lela94], [Lau95]. Even the most casual observer can notice the marked difference between self-similar traffic and Poisson traffic, as shown in Figure 2, which is a reproduced here from the 1994 Bellcore paper. A good description of the problem can be found in [Doul95].

1.3. Important Definitions

1.3.1. Burstiness

Burstiness relates to the non-uniform flow of network traffic. A channel which supports synchronous traffic at a constant bit rate is not bursty. One common metric of network traffic "burstiness" is the ratio of the peak-to-average bit rate for *message traffic* (as opposed to frame separators and idle frames). Since there seems to be no industry consensus for a single burstiness metric, several candidate metrics had to be evaluated as part of this project. For a discussion of burstiness see p.12 of Leland et. al. [Lela 94].

1.3.2. Quality of Service (QoS)

For ATM networks, the two levels of performance to consider are: 1) the cell-level performance (cell loss and delay), and 2) call-level performance (call blocking as relates to Connection Admission Control (CAC)). No universally accepted metric exists for *QoS* on ATM networks, although work is being actively pursued in this area by the ATM Forum. In order to guarantee *QoS* to the user, it is necessary to introduce a set of *QoS* parameters whose properties indicate the nature and requirements in the layered protocol stack. For this project, the *QoS* focus will be on cell-level performance, and key parameters are: 1) the ratio of cells lost to total cells offered for transmission, and 2) the delay distribution as cells traverse the network, end-to-end.

2. Objectives

The general objective of this project is the development of tools and algorithms which facilitate efficient use of ATM network resources and satisfy the customer's requirements for quality of service (QoS). The software modules developed in this project should augment the library of tools that can foster the research, education, and development efforts in the areas of network resource optimization and traffic management.

A list of specific objectives includes:

- Extension of our existing literature survey for related work on ATM network optimization and traffic management.
- Definition of network parameters suitable to the problem context for a typical elemental ATM network, which also must be designed.
- Definition of a set of metrics for evaluating the ATM network design. Of particular importance is the determination of a suitable metric for traffic "burstiness", since there does not appear to be a commonly accepted metric in use today. Metrics relating to quality of service (QoS) should include cell loss ratio, bandwidth utilization, and end-to-end delay jitter.
- Development of a "toolbox" of script programs to process and manipulate network trace files suitable for input to the NIST ATM simulator, i.e. simulator pre-processing tools.
- Development of post-processing scripts to analyze simulation log files and extract QoS data and statistics. Script files for "pre" and "post" processing are a major deliverables of this work. The role which these scripts play in the simulation process is depicted in Figure 3.
- Development of an algorithm capable of predicting the ATM network bandwidth needed to carry traffic characterized by varying degrees of burstiness. The goal here is to be able to provide network administrators with efficient tools to plan or utilize more efficiently the available bandwidth in ATM networks and to provide reliable network services to the users.

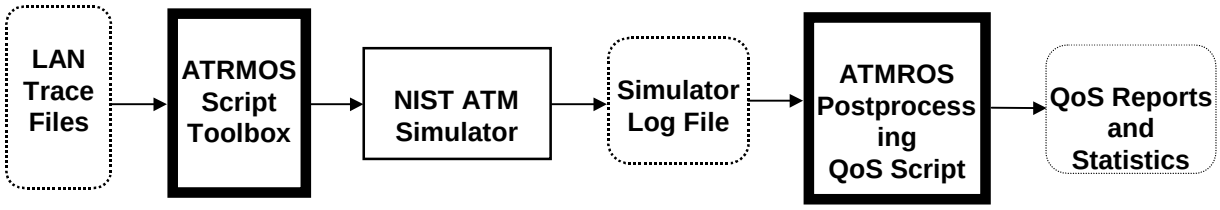


Figure 3. Simulation flow, highlighting the role of ATMROS simulation scripts.

3. Approach

3.1. Definition Phase

3.1.1. Literature Search

Initial efforts on this project focused on becoming familiar with the current state of ATM research and industry practice. To this end, a literature search was conducted which included network performance modeling, quality of service metrics, switch architecture, traffic analysis, traffic management, and recent papers on self-similar traffic. The bibliography for this project is included as Appendix-A.

3.1.2. Simulation Approach, Simulator Selection and Installation

Besides the literature search, a discrete event simulator was needed on which to carry out the network modeling. ATM cells must be tracked as they arrive at the network, are processed by network elements, and as they leave the network. Several candidate simulators were investigated which would be affordable in a university environment on the one hand, were well suited to network simulation, and which would be acceptable to the project sponsor. It was finally decided to adopt the ATM simulator developed by NIST (National Institute of Standards and Technology), which became available in August of 1995. The NIST simulator was specifically designed for ATM and there was no charge for it. It provides a GUI for displaying simulation results and permits specifying network topology and routs of VC connections. In retrospect, the downside of the NIST selection was that no support was available, documentation was lacking, and the simulator was prone to crash with the least provocation. There were also surprises such as hard coded limits for some parameters, premature run termination, and a myriad of parameters which were not described in the documentation. As received, the NIST simulator did not support input from a trace file. Modifications to the simulator have been made which accepts a LAN trace and generates the corresponding ATM cell arrival events. Also added was code to generate the VCI and cell ID to facilitate the correlation of cells traversing the network.

Existing computer resources available at the University of Colorado at Colorado Springs (UCCS) were used. Initially, the NIST simulator was installed on a Digital Equipment Alpha workstation running UNIX. The NIST simulator was then ported to a networked Pentium PC running LINUX, since the screen image was improved, and there were far fewer crashes.

3.1.3. Network Traffic Traces-- Simulation Issues and Decisions

Recent papers underscored the importance of using self-similar network traffic in order that results are meaningful. The NIST simulator can only generate Poisson traffic, which is not considered to be adequate. As a project task, it would be necessary to modify the simulator to accept a table driven format. Choices then, were either to attempt to generate self-similar LAN traffic, or to use network traces which had been captured on real LAN networks. It quickly became clear that generating self-similar traffic with known characteristics was mathematically daunting, and would also consume human resources which would not be available for the main work. Therefore it was decided to use archived LAN traces for the current study. Such traces are available from Bellcore[1], the ITA[2], and elsewhere[3] via the Internet.

[1] URLs are: flash.bellcore.com dir = pub/lan_traffic, thumper.bellcore.com dir = pub/vbr.video.trace.
[2] ITA = Internet Traffic Archive. URL is: http://town.hall.org/Archives/pub/ITA

3.1.4. Establishing a Burstiness Metric

In order to provide a consistent basis for comparing the "burstiness" of network trace files or segments from those traces, a burstiness metric is needed. At present there is no clear consensus within the network industry for any single burstiness metric. A common metric for burstiness is the peak-to-average value of the bit rate. However this is not an intuitively satisfying metric, especially for self-similar traffic (see Section 4.2). The Hurst constant, H, is most frequently used in the context of self-similar network traffic. However, little correlation was evident between either the visual (plotted) or peak-to-average burstiness and the Hurst constant. Therefore, a considerable part of the project activity was spent addressing the issue of establishing a burstiness metric which was both intuitive as well as practical. Section 4 covers this work in detail.

An additional task implicit in doing a visual "sanity check" on trace burstiness is the need to be able to plot network traces. In this respect, there are two separate needs. First, a plot routine must be developed to verify the self-similar nature of network traces over multiple decades of sampling intervals, similar to the plot sequence shown in Section 1. Second, there is a need to examine network traces both in their entirety, and on an expanded time scale to compare the relative burstiness of two specific intervals. Plot routines for both must be developed.

Besides doing visual checks on burstiness, there is a need to be able to process an entire network trace file to identify the ***n*** burstiest spots, against whatever burstiness metric is being evaluated. Bursty regions so identified should be ranked ordered by burstiness value and written to file for comparison of one burstiness metric with another. Thus, there is a need to develop a program which can identify trace "hot spots", and the user should be able to specify the time duration for bursty regions as a parameter.

3.2. Network Modeling Phase

3.2.1. Install Simulator and Validate Operation

A number of tasks are implied here. Installation of the NIST discrete event simulator for ATM is one task. The NIST simulator is capable of simulating constant bit rate (CBR) traffic and traffic which varies according to a Poisson distribution. Basic operation can be verified using CBR traffic and a simple single source / single destination network featuring two ATM switches and a common trunk between them. Examination of the simulator log file can verify whether simulator behavior is reasonable. Ultimately, however, the NIST simulator must be modified to accept actual network traces, which display self-similar burstiness, since Poisson traffic represents a lesser stress on the network. Thus, the NIST simulator must be modified to accept tabular input.

Once modified to accept tabular input, the simulator should again be tested for proper operation. Here again, it would be convenient to have the capability to generate files representing CBR traffic which could be input to the simulator. Thus, a utility to generate CBR traffic must be developed using as parameters, bytes per interval, interval time, and total trace duration.

3.2.2. Modeling Network Bandwidth Needs vs. Quality of Service

It has already been emphasized that there is no industry consensus on a common metric for traffic burstiness. To further compound the problem, there is also no common set of "quality of service" (QoS) metrics. The state of ATM network services is such that network service

[3] URL: http://www.nlanr.net/Flowresearch

providers often negotiate QoS parameters on a customer-by-customer basis, depending on the "perceived" nature of the customer's traffic. In fact, this curious circumstance is the prime reason for undertaking this project. Our goal is to better understand how to characterize the nature of customer traffic so that both the customer and ATM service provider can reach an agreement which admits efficient use of the ATM network on the one hand, and so that the customer is pleased with the quality of service on the other. A separate task, therefore, is the identification of a consistent set of parameters which might be agreed upon between network provider and customer. For example, what is a reasonable cell loss ratio for specific traffic types (CBR, ABR, UBR, etc.). Cell delay and delay jitter across the network are also important parameters. These parameter sets can then be used both to set parameters for the NIST simulator, and also for evaluating QoS results.

Another task in modeling ATM traffic is the need to specify the network topology. Owing to the large number of parameters surrounding the project, it was decided to restrict the set of network topologies to primitive cases. One topology to be simulated is simply the single source / single destination network described above. The other configuration builds on the first by adding several other traffic sources to multiplex into the originating switch. The size of this project does not permit expanding the network scope beyond these possibilities.

While the NIST simulator is capable of generating a log file of user selected metrics, the information is not complete enough to extract the necessary QoS metrics. For example, there is no cell ID information for correlating the cell delay, delay jitter, and cell loss. It also does not provide module for reading a network trace file and generate cell arrival events. Therefore, the simulator must be modified to enhance the log file with that additional information. Then, once the simulator has been modified to produce a complete log file, an additional program must be written to process the file and extract QoS parameters for specific simulation runs, and to create summary reports of those parameters and statistics for subsequent analysis and review.

3.2.3. Data Analysis Phase

Only after the considerable work of generating all the utility script programs and modifying the NIST simulator to accept table driven traces, can the exciting work begin. Then another series of tasks must be carried out. At a minimum, these include:

- Extraction of bursty segments from actual LAN traces to use as input to the simulator.
- Running initial simulations to understand what constitutes a "balanced" set of parameters.
- Design and execution of a set of experiments (simulations) using trace burstiness as a parameter, while holding queue size and other network parameters constant.
- Design and execution of a set of experiments using bandwidth as a parameter, while holding trace burstiness constant.
- Process the log files to determine the level of QoS achieved with each simulation.

Data analysis of the log files should permit the creation of a families of curves which show degradation of QoS vs. Burstiness at fixed network bandwidth. A second view of the problem domain would show how QoS improves at fixed burstiness as network bandwidth is increased. Being able to visualize this tradeoff is an important goal of this project.

4. Results

4.1. Analysis of Bellcore Trace File: OctExt.TL

4.1.1. Introduction

The discussion which follows highlights initial efforts at scanning a LAN trace for high activity periods. Specifically, a method is sought which can identify intervals from the trace which could stress a LAN/WAN switch queue to the extent that either cells may be lost, or delay in delivering the cell could become excessive. A major part of this effort was directed at identifying a meaningful metric for these high activity or "bursty" periods. There does not seem to be a consensus within the literature for any specific "burstiness" metric. The next several sections examine several candidate burstiness metrics, and their relative effectiveness in identifying high activity areas within LAN traces.

4.1.2. The Inadequacy of Peak/Average Ratio as a Burstiness Metric

Early in the analysis effort, it became obvious that a simple "peak-to-average" (P/A) technique would not be satisfactory. A little reflection will give the reason why. Suppose that we are looking for high levels of activity in 60 consecutive one second intervals. Suppose further that 59 of the 60 one second intervals have zero (0) cell count (arrivals), and that during the remaining one second interval X cells arrived. Then the peak-to-average ratio for this period is:

$$P/A = X(\text{i.e. peak})/(X/60)(\text{i.e. average}) = 60$$

First notice that P/A has its highest possible value, and second, that it doesn't even matter how many arrivals there were. In this instance, P/A is independent of X, the number of cell arrivals. A quiet window having a single interval burst, while it represents a very high "peak-to-average" ratio, would not typically overstress the queue of a network switch, since the average value is low. Not only does P/A fail to identify most trouble spots in a LAN trace, but it is not an intuitively satisfying measure.

4.1.3. Peak/Average with a Floor as a Burstiness Metric

In order to overcome this limitation, it was decided to retain the P/A concept, but to further require that each qualifying window of 60 seconds duration have a number of cell arrivals greater or equal to the average for the entire trace file. Suppose that it is desired to capture the *K* burstiest regions in the trace. If these are ordered such that we know the P/A value of the least bursty interval, *P/A_min*, then we will accept any new 60 second interval if it has *P/A_new* which satisfies the conditions:

if ((*P/A_new* > *P/A_min*) && (*Cell Arrivals* >= *Avg Cell Arrivals*))
then replace old minimum interval with new interval P/A and its corresponding interval start time.

In terms of the above description, we have created an arbitrary "floor" which must be greater than or equal to the average cell arrivals for the LAN trace as a whole. In this instance, the floor is said to have a value of 1x. If it is required that twice the average of cells must arrive within

each 60 second interval, then the floor is 2x, etc. For simplicity, this modified peak/average metric is hereafter called "*P/A with floor*".

4.1.4. Hurst Constant (*H*) as a Burstiness Metric

The principal parameter characterizing "self-similar" processes reported in the literature is the Hurst constant, *H*, which is frequently used as a burstiness metric. Due to the computational complexity of *H*, a mean likelihood estimate (MLE) is generally used. The most popular MLE is "Whittle's Estimator", which was developed in 1953 in the context of short-memory time series. Code which implements Whittle's Estimator was acquired via the Internet[4], and was used in the analysis below to approximate *H*. *H* can take on values in the range 0.5 - 1.0. A value of 0.5 over some region does not imply a constant value, but rather that the distribution is normal, and that there is no cross-correlation between elements. The higher the value of *H*, the burstier the region. *H* is nonlinear, and values greater than 0.9 are exceedingly bursty. Strictly speaking, Whittle's Estimator only has relevance for phenomena which obey a Gaussian distribution, but it is often used in the context of burstiness of LAN, WAN and video network traffic [Lela 94], [Garr95], [Bera95].

4.1.5. Trace File and Assumptions Used in Burstiness Metric Analysis

The trace file used in this initial characterization was the Bellcore file OctExt.TL, which includes 49,366 one second intervals of LAN traffic. This represents 13.7 hours of traffic. A complete view of the LAN trace is given in the Appendix-B as a series of 9 plots, each containing 1000 intervals of 6 seconds each. In order to get some idea how effective intuition might be in identifying bursty periods in the LAN trace file, eleven "regions" were selected visually, and labeled #1 through #11. These regions nominally "looked like" the worst 11 regions in the trace, exhibiting high peak values of cell arrival. Initial efforts at identifying bursty intervals of 60 seconds used both the *P/A with floor*, and *H* metrics.

4.1.6. Relative Effectiveness of Burstiness Metrics "*P/A with Floor*" and "*H*"

- It has already been pointed out that raw *P/A* as a metric is neither effective nor intuitive. Results of scanning trace OctExt.TL with script scan_trace.pl using 60 second windows comprised of 60 one second 'bins' found only one (1) of the 11 'visually bursty' regions. Stated another way, there were a total of 29 bursty regions identified either visually or by the several script files developed to scan trace files. Raw *P/A* found only 7 of those 29.

- Five different floor values {1x, 2x, 4x, 8x, and 12x} were used to better understand the effectiveness of *P/A with Floor*. In the best case, *P/A with Floor* found 23 of 29 bursty regions, but only 6 of the 11 visually identified regions. Tables 1 - 4 summarize results.

  Intervals visually identified as #11, #8, #9, and #3 were never found by the *P/A with Floor* metric. There were other areas of bursty behavior which were also missed. More detailed plots of the "hot spots" identified by *P/A with Floor* showed that although the peak values may have been high, they were relatively uniform. That is, the average was also high, so that the P/A values might not have been significantly greater than 1.0. Results of a subsequent effort which used statistical "variance" as a burstiness metric are given in a later section.

[4] G.J. Miller, S. Dastangoo, The MITRE Corp. URL: ftp://foghorn.ie.org/pub/selfsim/whittle.tar.Z

1X floor													
Region	12	2	10	13	14	15	16	17C	20	24	17A		
Peak/Avg	39.28	38.41	32.76	28.25	25.98	25.78	25.37	24.37	22.96	22.89	22.57		
Start	24429	30119	8393	10760	7736	22865	17867	28691	30753	25178	28260		
Stop	24489	30179	8453	10820	7796	22925	17927	28751	30813	25238	28320		
Whittle H	0.596	0.500	0.500	0.767	0.500	0.618	0.575	0.607	0.934	0.511	0.713		
2X floor													
Region	2	14	1	6A	17A	19	17C	23	16	17B	20	6A	21
Peak/Avg	23.46	18.42	18.33	16.22	13.34	13.24	12.80	12.52	12.17	11.89	11.42	11.07	11.06
Start	30120	7740	40046	29368	28262	36395	28695	31763	17890	28490	30750	29192	4278
Stop	30180	7800	40106	29428	28322	36455	28755	31823	17950	28550	30810	29252	4338
Whittle H	0.500	0.608	0.698	0.895	0.665	0.521	0.740	0.500	0.750	0.926	0.956	0.949	0.813
4X floor													
Region	2	17B	1	6A	17A	22							
Peak/Avg	14.46	11.89	11.64	9.37	7.62	7.43							
Start	30136	28490	40049	29461	28270	37679							
Stop	30196	28550	40109	29521	28330	37739							
Whittle H	0.558	0.926	0.728	0.954	0.613	0.615							
8X floor													
Region	17B	1	6A	4	6A-	25	23A	5	7				
Peak/Avg	6.77	6.32	5.20	3.00	2.98	2.97	2.78	2.75	2.35				
Start	28530	40090	29427	45978	29169	45484	31929	49243	15095				
Stop	28590	40150	29487	46038	29229	45544	31989	49303	15155				
Whittle H	0.940	0.968	0.938	0.914	0.948	0.743	0.935	0.745	0.953				
12X floor													
Region	17B	25	6A-	23A									
Peak/Avg	5.48	2.49	2.36	2.19									
Start	28539	45501	29162	31972									
Stop	28599	45561	29222	32032									
Whittle H	0.941	0.811	0.926	0.943									

Table 1: Comparison of Burstiest Regions (P/A with Floor vs. H) for Trace: OctExt.TL

- Use of the Hurst constant, *H*, was not effective in identifying bursty periods. There is very little correlation between *H* and *P/A with Floor*, as can be seen in Table 1. For example, values of *P/A with Floor* have their greatest value on the left, and decrease progressively to the right. Inspection of detailed plots of these regions confirms this general trend. By contrast, the value of *H* shows no apparent trend. The Hurst constant, *H*, may have merit for characterizing LAN traffic on a long time scale, but it does not appear to be useful for estimating burstiness of short periods such as the one minute intervals analyzed. A later section discusses a series of tests comparing *H* with a number of metrics for both short and long traces. First, however, we complete the comparison between *P/A with Floor* and *H*. In addition, "variance" is also introduced as a burstiness metric. Then all burstiness metrics are compared over common intervals ranging from 100 seconds to 12,800 seconds.

- As a point of interest, plots of "cell arrivals per second" and similar plots measuring "packet arrivals per second" looked very much alike. This is no surprise, since bytes in each

packet are segmented into ATM cells each carrying a 48 byte payload. Script programs were developed to permit either representation.

- Although five floor values were used, within each floor level, *P/A with Floor* did not identify many "visual" hot spots (i.e. those numbered 1-11). A total of 29 bursty areas were identified by these five separate scans of the traffic. Each pass over the LAN trace captured the 200 burstiest 60 second intervals. Taking the burstiest region first, and proceeding to the least bursty of the 200, regions of high burstiness were identified. If the same region was identified multiple times, it still was counted only once. Given this procedure, the distribution of regions identified for OctExt.TL is shown in Table 2 with the most bursty regions at the left.

Floor Value	Regions Identified (visually and/or with scripts)
1x	12, 2, 10, 13, 14, 15, 16, 17C, 20, 24, 17A
2x	2, 14, 1, 6A, 17A, 19, 17C, 23, 16, 17B, 20, 6A-, 21
4x	2, 17B, 1, 6A, 17A, 22
8x	17B, 1, 6A, 4, 6A-, 25, 5, 7
12x	17B, 25, 6A-, 23A

Table 2: Regions identified vs. Floor Value for *P/A with Floor*

- Table-2 shows that of the 29 regions identified (either visually or with program scripts), the five different floors captured widely differing region sets, with some small amount of overlap. This underscores the need for more efficient filters for bursty periods. A second "phase" was launched to improve the bursty region identification algorithm.

4.1.7. An Improved Filter Algorithm for Local Bursty Regions

As outlined above, initial efforts at identifying bursty regions within the trace resulted in many duplicate entries in the output array which represented the same region. When a floor of 4x was used, for example, only six (6) unique regions were represented out of the 200 burstiest minutes. A simple and efficient alternative "passes over" bursty regions already identified. A parameter called "factor" was defined, which is a "distance" multiplier on the window size (seconds). Suppose that a window size of 60 seconds is used in scanning the LAN trace, and that "factor" has a value of 0.4. Then any new candidate region having a starting time differing by less than 24 seconds (time distance) from the starting time of a previously captured region will be discarded. This is true unless the candidate region is burstier than the previously captured region, in which case the candidate region will replace the previously captured region.

The algorithm was further enhanced to guarantee that the desired number, ***n***, of bursty regions were identified. This was achieved by filling the array of bursty regions with the first ***n*** non-competing regions. Otherwise, if the burstiest portion of the trace occurred early, subsequent bursty regions would be discarded before ***n*** regions could be identified. It was in this way that script *scan_trace.pl* evolved into *scan_fltr.pl*.

4.1.8. Variance, a Discriminating Local Burstiness Metric

The first trace scanning script, scan_trace.pl used "variance" as a metric for finding quiet periods in the trace. On the chance that variance might also prove effective as a burstiness metric, a

third script called "scan_var.pl" was created. This script replaced the P/A metric in scan_fltr.pl with statistical variance. Variance performed very well indeed, as described in the next section.

4.1.9. Discriminating Power of Burstiness Metrics as a Function of Parameters

Recall that a major goal of this work is to find a technique which can be efficient at locating bursty regions within a network trace file. To this end, a series of runs were made on trace OctExt.TL using each of the burstiness metrics discussed above. Program scripts representing *P/A with Floor*, "raw bytes", and "variance", were used to examine 60 second windows in an attempt to identify a good compromise in identifying bursty regions. The values of both *factor* and *floor* were varied, in the case of *P/A with Floor* and *variance*. As a measure of success, the technique which identifies the greatest number of bursty regions (of the 29 originally identified in Phase-1) is judged the winner. Results of these tests are summarized in Table 3.

	Raw Bytes	P/A with Floor	P/A with Floor	P/A with Floor	P/A with Floor	Variance	Variance
Factor: Floor:	0.5 N/A	0.8 1x	0.8 2x	0.8 4x	1.0 2x	1.0 N/A	1.0 1x
Regions found	15	23	21	17	23	26	27
Regions not found	14	6	8	12	6	3	2
New found Regions	7	4	7	5	10	10	11
Range of P/A ratio	N/A	39.3-14.5	23.5-8.0	14.5-3.0	23.5-7.6	N/A	N/A

Table 3: Performance of various burstiness metrics.

In the case of "raw bytes", the script simply reported which regions had the greatest number of byte arrivals. Note that raw bytes earned the poorest score, since only 15 of 29 bursty regions were identified. *P/A with floor* demonstrated an improvement over *raw bytes*. Here, a filter width equal to the window width (60 sec) and a qualifying floor of twice the trace average byte arrivals correctly identified 23 of 29 bursty regions. In addition, *P/A with Floor* also identified 10 new bursty regions beyond the initial 29. However, results with *variance* proved to be best of the three. Using a filter equal to the window width (60 seconds), variance correctly identified 26 of 29 visually or script identified bursty regions. Moreover, this was variance with no floor. When a 1x floor was added, the score improved to 27 out of 29 regions, and it also identified 11 new areas beyond the initial 29.

4.1.10. Comparison of Burstiness Metrics on Variable Trace Intervals

Earlier it was pointed out that the Hurst constant, *H*, did not seem to work well when applied to short trace intervals. In order to better understand how *H* performed, a series of trace intervals from OctExt.TL were used ranging from 100 seconds to 12,800 seconds in binary multiples. All intervals shared the same start time of 15000 seconds. Start time was chosen because there are two very strong spikes which occur in the 15100-15160 region, followed by a fairly long quiet period. Thus, the 100 second interval just misses seeing the bursty region. However, all subsequent intervals contain this bursty region (#7 in Appendix-B). The longest interval extends to 27,800 seconds, and contains a region of moderate activity having an almost constant level near its termination. Note that the time scale in Appendix-B is compressed by a factor of 6. Scripts representing all burstiness metrics were run on each interval. Results are summarized in Table 4.

Time Interval (seconds)	**Hurst Constant (H)**	**P/A (raw)**	**P/A** (60 sec. window) **scan_trace.pl**	**P/A, 2x floor** (60 sec. window) **scan_fltr.pl**	**Variance** (60 sec. window) **scan_var.pl**
100	0.5	9.92	14.84	---	344,826
200	0.987184	7.33	27.16	3.63	81,098,608
400	0.995287	14.01	27.16	6.69	81,098,608
800	0.997559	23.20	27.16	10.94	81,098,608
1600	0.998178	35.38	27.16	15.72	81,098,608
3200	0.998932	58.72	47.03	25.37	81,098,608
6400	0.921004	91.94	47.03	28.29	81,098,608
12800	0.872788	87.65	50.03	40.74	81,098,608

Table 4 Burstiness metrics compared over varying time intervals.

4.1.10.1. Hurst Constant *(H)*

Whittle's estimate of the Hurst constant in the first interval is given as 0.5 indicating that this region contains activity which exhibits a "normal" distribution. It is, in fact quiet and relatively uniform. All other intervals contain one or more of the 29 bursty regions. Estimates for H increase until the last two intervals. This is reasonable, since the activity exclusive of region #7 is fairly quiet. That is, the longer the quiet interval persists, the more unpredictable (and bursty) is the interval, given that activity such as region #7 has occurred. The last two intervals again start to pick up some substantive traffic, but which is not particularly bursty. It is gratifying to see that Whittle estimates of *H* begin to decrease accordingly.

4.1.10.2. Raw Peak/Average Ratio

The raw P/A ratio is calculated by taking as peak, the highest one second "bin" within the interval, and dividing by the average bin activity over the entire interval. Although bursty region #7 occurs during the second interval, raw P/A shows a greater burstiness in the first interval. This is not a surprise, since the cumulative activity in region #7 increases the "average" to the point that the ratio decreases. Then, in subsequent intervals having very low levels of activity, the average falls off again, but the peak value remains constant, leading to gradually increasing values of raw P/A ratio. As discussed before, raw P/A is a counterintuitive metric for burstiness.

4.1.10.3. P/A using script scan_trace.pl

This metric is still raw P/A, but instead of using the entire interval to compute an average, it slides a 60 second window over the trace. The greatest value found for any 60 second window is what is reported. Again, not a very intuitive or gratifying metric.

4.1.10.4. P/A with floor using script scan_fltr.pl

As discussed earlier, adding a minimum "floor" of window activity to qualify that window improves the ability to identify local bursty regions. However, if the maximum value for a given 60 second window is retained for a trace over a longer interval as it is here, it looks less satisfactory than *H*.

4.1.10.5. *Variance* using script scan_var.pl

Similar comments apply to variance as for *P/A with floor*. *Variance* can be a very effective screen for local bursty regions. However, using the maximum *variance* found for any 60 second window does not weight new portions of the trace as the trace continues. If the entire trace interval were segmented, and if each segment's contribution were weighted, then *variance* and *P/A with floor* would increase and decrease in a similar manor to *H* for long traces intervals.

4.2. Description of Script Programs and Simulator Modifications

4.2.1. Overview of Script Programs

The focus of this project is network traffic, and specifically the impact which traffic burstiness has on the bandwidth needed to provide ATM service at some QoS level. In order that meaningful research can proceed, it is necessary that tools are available to manipulate the traffic traces. A number of tools, in the form of Perl scripts, were developed on this project to fill this basic need. They fall into two major categories. The first category includes script programs whose function relates to "manipulating" or "comparing" traces. Example functions include plotting traces, packet-to-cell conversion, or calculating the Hurst (Whittle) "burstiness" of network traces. Script programs in the second category are those which facilitate "simulation" of network traces. Functions of programs in this category include finding "hot spots" in the trace, extracting those hot spots as separate input files, generating CBR traffic independently of the simulator, and processing the simulation log file to generate summary QoS reports. Appendix-C is a flow diagram which describes how programs in the first category interact, including program function, input arguments, and output. Appendix-D is a flow diagram which gives a comparable overview of programs in the second or "simulation" category. Script programs were written in the Perl language, since it has powerful string manipulation capability. Perl is also portable to PC and UNIX platforms under Windows 95, NT, and Linux. A brief description of each script is provided in the sections which follow.

4.2.2. Script trc2plt.pl

trc2plt.pl accepts a trace file in the Bellcore format (packet arrival time, bytes/packet) and creates Postscript plot files showing burstiness vs. time as in Figure 4 of [Lela 94].

Name

trc2plt.pl - a Perl script

Syntax

trc2plt.pl plot_size time_mult 1st_time_interval start_t <trace_file>

Program Description

Perl script, trc2plt.pl, operates on a LAN trace file and outputs an intermediate file, ss_gnu.dat, which is suitable for plotting trace activity as a function of time. The script creates Postscript plot files, by calling script ss_plots.pl, and finally deletes the intermediate file, ss_gnu.dat. The entire trace file is processed, and one or more plots is generated, depending on the trace file and user supplied parameters. The input trace file is assumed to be in Bellcore format (packet arrival time, bytes/packet).

Argument Descriptions

plot_size: The number of time intervals included in each plot.

time_mult: If the time interval for the first plot, i.e. "1st_time_interval" is 1 second, and the user specifies 10 for the value of "time_mult", then the second plot will have time intervals of 10 seconds, the third plot will have time intervals of 100 seconds. etc. NOTE: If only a single plot is desired, set time_mult = -1.

1st_time_interval: Time period for each interval in the first plot.

start_t: The LAN trace time at which the first plot is to begin.

<trace_file>: The name of the LAN trace file whose plot is desired.

Example Input

trc2plt.pl 1000 10 1 5.0 OctExt.TL

Here, trc2plt.pl will operate on trace file OctExt.TL starting at time = 5.0 seconds. Earlier trace activity is discarded. Plots will be formatted having 1000 time intervals per plot, where the time interval for the first plot is 1 sec. The time interval for each subsequent plot will be 10 times greater than that in the preceding plot. See also cell2plt.pl.

Example Output

When printed, the plot files from the above command appear as shown. Compare with Figure 1 from the Bellcore paper.

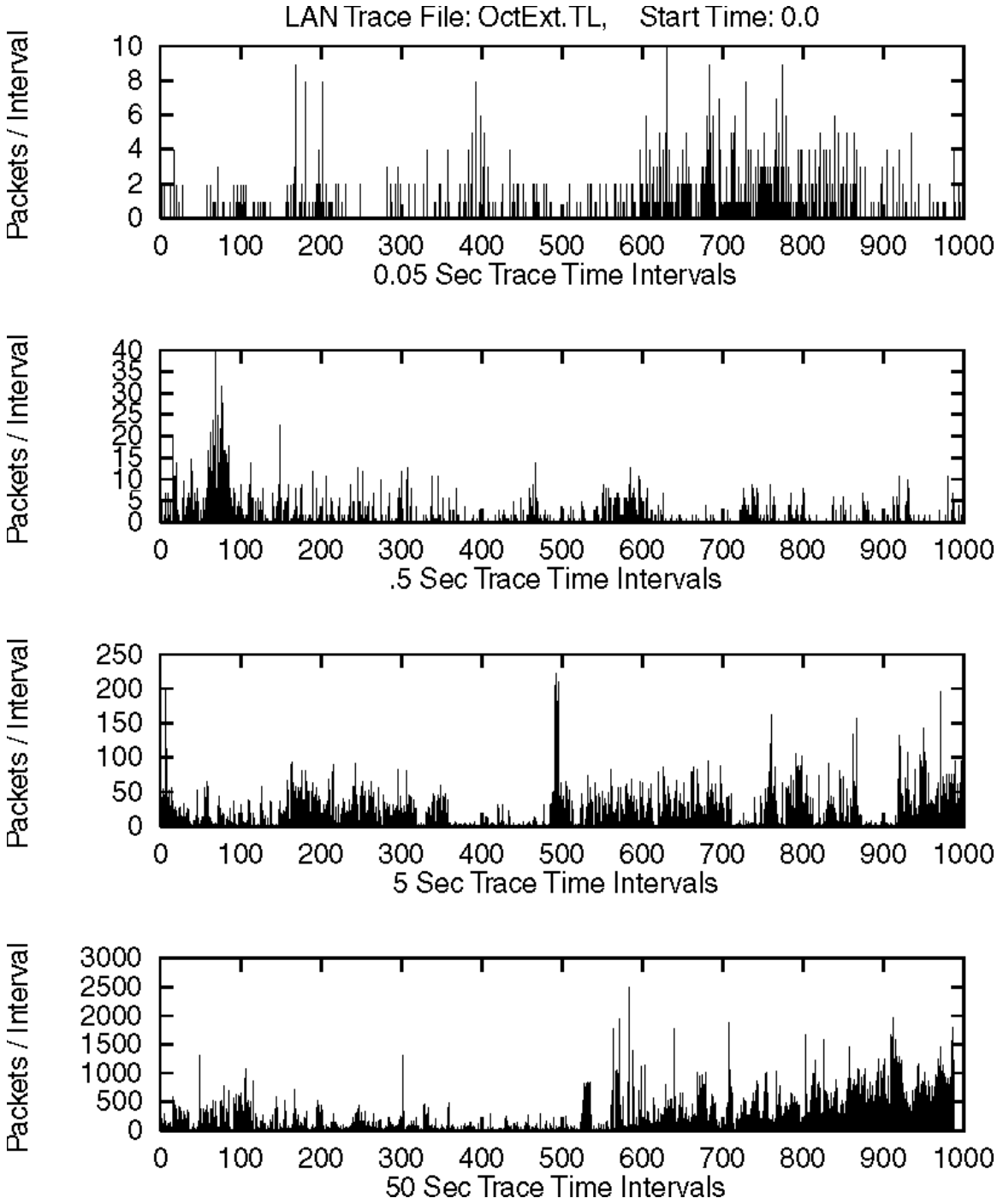


Figure 4. Trace OctExt.TL as plotted by script trc2plt.pl (4 plot files).

4.2.3. Script cell2plt.pl

cell2plt.pl is a first cousin to trc2plt.pl. The single difference is that instead of counting packet arrivals per interval, the script counts cell arrivals per interval, and expects <trace_file> to have the correct cell-based format. Another script, trc2cell.pl, can produce this input format.

4.2.4. Script trc2cell.pl

trc2cell.pl simply converts a packet-based trace to an ATM cell-based trace format. Each input record is converted to a corresponding output record.

Name

trc2cell.pl - a Perl script

Syntax

trc2cell.pl <trace_file>

Program Description

Perl script, trc2cell.pl, operates on a LAN trace file, which is assumed to be in Bellcore format (time, byte_count). The principal function of these script program is to convert each packet byte count to an equivalent number of ATM cells. This format is then written to file: <trace_file>.cel. No "binning" occurs in this script, only byte-to-cell conversion.

Argument Descriptions

<trace_file>: The name of the LAN trace file whose plot is desired.

Example Input

trc2cell.pl BC-pAug89.TL

Here, trc2cell.pl converts trace BC-pAug89.TL from a byte_count packet oriented file to 'out_file', which is ATM cell oriented.

Example Output

A truncated section of the input file, BC-pAug89.TL is included, followed by the comparable output.

```
0.001340  1090
0.001508   174
0.004176   162
0.008140   174
0.011036   162
```

```
blanca.uccs.edu% trc2cell.pl BC-pAug89.TL

BC-pAug89.TL tracefile converted to BC-pAug89.cel

blanca.uccs.edu% head -50 BC-pAug89.cel
  0.001340       23
  0.001508        4
  0.004176        4
```

0.008140 4
0.011036 4

4.2.5. Script trc2atm.pl

trc2atm.pl processes a trace in the Bellcore (packet time arrival, bytes/packet) format, and produces auxiliary files for plotting and/or computation of the Hurst (H) constant. A major difference between this script and ***trc2cell.pl*** is that ***trc2atm.pl*** accumulates cell counts in "bins" of fixed time intervals. That is, multiple records from the trace file may be consolidated into a single record of the output file(s).

Name

trc2atm.pl - a Perl script

Syntax

trc2atm.pl <trace_file> interval_time [W|P|B]

Program Description

Perl script, trc2atm.pl, is intended to accumulate the records from a trace file into "bins", based on record arrival time. Furthermore, the cumulative byte count from all packets arriving within a given bin are converted to ATM cells on a packet-by-packet basis. Thus multiple records from the trace file having a (time, byte_count) format are converted into a single equivalent "bin record" having the format (time, cell- count). Each "bin" has the same user specified length.

Three options exist for output, which the user can specify by entering W (Whittle), P (Plot), or B (both). If W is specified, then a file named <trace_file>.w is generated having a single cell-count field and no time field, since records represent cumulative cell arrivals at in constant "bin" times. This is the format required by program whittle.c, which is used to calculate the Hurst constant (*H*).

If output option P is selected, then binned or cell accumulation records will be output to file <trace_file>.p having the form (time, accumulated_cell_count) for subsequent plotting by Perl script cell2plt.pl. Finally, if output option B is selected, both file types are created.

Argument Descriptions

trace_file: The name of the LAN trace file which is to be processed by trc2atm.pl

interval_time: This it the time period over which packets are to be converted into cells and accumulated into a composite cell count record. Same as 'bin time'.

W, P, or B: Parameter which selects which output files to create.

Example Input

trc2atm.pl OctExt.TL 1 B

Here, trc2atm.pl will scan trace file OctExt.TL and accumulate cell equivalents in bins of equal time, 1 second each. Since output option B was selected, the cell counts will be written out in two files named OctExt.w and OctExt.p for subsequent processing.

Example Output

The first few records of OctExt.TL are provided below, followed by selected output corresponding to those records for both the .w and the .p files.

```
blanca.uccs.edu% head -19 OctExt.TL
   0.119076    82
   0.125692    82
   0.265360    82
   0.269592    82
   0.689840   112
   0.699500    64
   0.890468   108
   0.892932    64
   0.895044    64
   0.896060    64
   0.911968    75
   1.003232   123
   1.025148    64
   1.164304    64
   1.432864    64
   1.435644    64
   2.915004   132
   2.920388    83
   3.137452   121
blanca.uccs.edu%   trc2atm.pl    OctExt.TL    1    B

    Please Wait .. Processing 'OctExt.TL' .. creating output file

    Trace file 'OctExt.TL' has been converted to binned cell counts.

    49366 records (bins) were written to file.

blanca.uccs.edu% head -6 OctExt.p
*** Input trace file:    OctExt.TL
*** Time interval width: 1 seconds per bin
*** Fields: Bin Start Time,  Cells per Bin
     0.00        24
     1.00        11
     2.00         5
blanca.uccs.edu% head -6 OctExt.w
*** Input trace file:    OctExt.TL
*** Time interval width: 1 seconds per bin
*** Fields: Cells per Bin
     24
     11
      5
```

4.2.6. Program whittle

whittle is a compiled 'C' program obtained via the Internet which calculates the burstiness metric, *H* (Hurst constant) for a given trace file or a segment from a trace file.

Name

whittle - a 'C' executable program.

Syntax

whittle (Arguments provided interactively: <trace_file>.w records start_rec stop_rec num_series)

Program Description

This is a top-level routine for computing the Hurst parameter, (H), for a set of data using Whittle's estimator. The algorithm was translated by Gregory J. Miller and Siamak Dastangoo, The MITRE Corp, from an S-Plus implementation published in "Statistics for Long-Memory Processes" by Jan Beran, Chapman and Hall Publishers, NY, 1994.

Argument Descriptions

<trace_file>.w: The name of the LAN trace file having a single field per record (cells, bytes, etc.), with records assumed to be at constant time intervals.

records: The total number of data records in the trace file.

start_rec: The index of the first data record at which whittle.c is to begin computing the Hurst parameter, H.

stop_rec: The index of the last data record at which whittle.c is to terminate computing the Hurst parameter, H.

number of series: Statistics parameter .. always set = 1.

Example Input / Output

The **whittle** program asks for each parameter interactively. In the example below, trace file OctExt.w contains a total of 49366 records, each containing only byte counts, packet arrivals, or cell arrivals as the single field in each record. Calculation of the Hurst constant will begin with record 36299 and proceed through all records up to 33499. Results are written to the active output as shown below.

```
bilbo% whittle

In which file are the data? OctExt.p
Total data records in file: 49366
Start processing with i-th record: 36250
Stop processing with j-th record: 36450
Into how many sub-series should the data be divided? 1
Will use 128 points per sub-series (largest power of 2 <= 201).
Reading data file "OctExt.w" . . . done.

H = 0.514900
```

Computing confidence interval . . . done.
theta = 0.162050
95%-C.I. for H: [0.406254 0.623546]

4.2.7. Script scan_trace.pl

scan_trace.pl scans a trace file in the Bellcore format for both bursty and quiet "hot_spots". Bursty spots use peak-to-average as a metric, whereas quiet spots use variance. No "floor" is used, and the same region may be found many times, but regions are ranked in order of start time.

Name

scan_trace.pl - a Perl script

Syntax

scan_trace.pl window_size bins/window num_periods <trace_file>

Program Description

Perl script, scan_trace.pl, scans a LAN <trace_file> for both bursty and quiet regions. Bursty regions are identified using peak-to-average ratio as burstiness metric. Quiet regions are identified using statistical variance as a quietness metric. The program permits the user to identify the starting times for the "n" burstiest regions of the specified trace. The user must specify the region size (window_size) and an integral number of "bins" per window. Regions found are written to file 'hot_spots'.

Argument Descriptions

window_size: Defines the time span of target bursty/quiet regions.

bins/window: Refers to the granularity of the window (region). There must be in integral number of bins per window. "bins" all have the same duration, defined by the quotient of window_size divided by bins/window.

num_periods: The desired number of burstiest regions for the program to identify. Two lists are created; one for bursty regions, and one for quiet regions.

<trace_file>: The name of the LAN trace file which is to be scanned for bursty regions.

Example Input

scan_trace.pl 60 60 50 OctExt.TL

Here, scan_trace.pl will scan trace file OctExt.TL using windows of 60 seconds each (one minute), and having 60 one second bins. The program will identify the 10 burstiest regions and the 50 quietest regions found, and these regions will be written to file 'hot_spots'. See also: scan_fltr.pl, scan_var.pl.

Example Output

owl.uccs.edu% scan_trace.pl 60 60 50 OctExt.TL
Max individual bin count in trace is 67096
Average arrivals in a single bin is 1133.43
Average sum of all bins in a window is 68005.95

Please be patient while windows are being processed.
This could take minutes or hours. Parameter dependent!
2500 bins, 100 sec windows, 100 bins/win .. took 4.25 min.

Bursty and Quiet periods stored to file 'hot_spots'.

owl.uccs.edu% more hot_spots

****
****

Output from 'scan_trace.pl' is file: hot_spots

****
****

Window size (sec): 60
Bins per window: 60
Num. of hot spots: 50
LAN Trace file: OctExt.TL

Max individual bin count in trace is 67096
Average arrivals in a single bin is 1133.43
Average sum of all bins in a window is 68005.95

The 50 most bursty windows in OctExt.TL are:

Window Start Time	Peak/Average Burstiness
8392.00	51.37
22862.00	50.03
25152.00	48.17
29792.00	48.06
24420.00	47.17
:	:
21982.00	45.46
21983.00	45.37
22002.00	45.37
21984.00	45.37
21985.00	45.37

The 50 quietest windows in OctExt.TL are:

Window Start Time	Quietness (variance)
16447.00	41569.73
16450.00	41364.93
16449.00	41296.67
16448.00	41296.67
12479.00	41236.37
:	:
35833.00	32344.68
34761.00	31607.92
34760.00	30573.18
34762.00	29179.32
34763.00	28916.92

4.2.8. Script scan_fltr.pl

scan_fltr.pl scans a trace file in the Bellcore format for both bursty “hot_spots”. Bursty spots use peak-to-average as a metric, and also use a “floor” to guarantee that minimum level of activity within each qualifying region. Additionally, a filter is used to prevent recording duplicates of the same region. Once hot spots are identified, they can be extracted and used for simulation input.

Name

scan_fltr.pl - a Perl script

Syntax

scan_fltr.pl window_size bins/window num_periods <trace_file>

Program Description

Perl script, scan_fltr.pl, scans a LAN <trace_file> for bursty regions using a peak-to-average ratio as burstiness metric. The program permits the user to identify the starting times for the "n" burstiest regions of the specified trace. The user must specify the region size (window_size) and an integral number of "bins" per window. Regions are written to the file 'hot_spots'.

There is also a parameter which is hard coded in the program called 'factor', which is a real multiplier times the window_size. Currently set at 1.0, this causes the program to discard any bursty region whose start time is within 1.0 * window_size of any bursty region captured so far .. unless the new region has a burstiness which is greater than the previously captured region, in which case the old region is replaced by the new region.

A second hard coded parameter is ,floor'. A floor of 1x requires that candidate regions must have more arriving bytes than the average number for the trace as a whole. The current value for floor is 2x, requiring that each region have at least twice the average byte arrivals.

Argument Descriptions

window_size: Defines the time span of target bursty regions.

bins/window: Refers to the granularity of the window (region). There must be in integral number of bins per window. "bins" all have the same duration, defined by the quotient of window_size divided by bins/window.

num_periods: The desired number of burstiest regions for the program to identify.

<trace_file>: The name of the LAN trace file which is to be scanned for bursty regions.

Example Input

scan_fltr.pl 60 60 50 OctExt.TL

Here, scan_fltr.pl will scan trace file OctExt.TL using windows of 60 seconds each (one minute), and having 60 one second bins. The program will identify the 50 burstiest regions found using Peak-to-Average ratio as the burstiness metric, and these regions will be written to file 'hot_spots'. See also: scan_trace.pl, and scan_var.pl.

Example Output

Output has been truncated for brevity.

```
owl.uccs.edu% scan_fltr.pl 60 60 50 OctExt.TL
   Max individual bin count in trace is      67096
   Average arrivals in a single bin is     1133.43
   Average sum of all bins in a window is 68005.95


   Please be patient while windows are being processed.
   This could take minutes or hours.  Parameter dependent!
   2500 bins, 100 sec windows, 100 bins/win .. took 4.25 min.
   Processed 10000 bins
   Processed 20000 bins
   Processed 30000 bins
   Processed 40000 bins

   Bursty and Quiet periods stored to file 'hot_spots'.

owl.uccs.edu% more hot_spots

************************************************************
************************************************************

   Output from 'scan_fltr.pl' is file: hot_spots

************************************************************
************************************************************

   Window size (sec): 60
   Bins per window:   60
   Num. of hot spots: 50
   LAN Trace file:    OctExt.TL
```

Max individual bin count in trace is 67096
Average arrivals in a single bin is 1133.43
Average sum of all bins in a window is 68005.95

The 50 most bursty windows in OctExt.TL are:

Window Start Time	Peak/Average Burstiness
30120.00	23.46
7740.00	18.42
40046.00	18.33
29368.00	16.22
40142.00	14.38
:	:
37553.00	7.80
41331.00	7.75
37698.00	7.64
4953.00	7.62

4.2.9. Script scan_var.pl

scan_var.pl is essentially the same as script scan_fltr.pl, but here variance is used as a burstiness metric instead of peak-to-average.

4.2.10. Script extract.pl

extract.pl is a utility which does a file copy of a user selected region in a larger trace. The extracted file can then be used as input to the simulator.

Name

extract.pl - a Perl script

Syntax

extract.pl <trace_file> T_advance T_start T_span <out_file>

Program Description

Perl script, extract.pl, extracts a segment of the trace file specified by the 'T' parameters, and copies that segment to the output file named by the user.

Argument Descriptions

trace_file: The name of the LAN trace file from which a traffic segment is to be extracted.

T_advance: An optional period of time before T_start, intended to "prime" the queues of a network switch.

T_start: The nominal start time of interest to the user, which should be less than the maximum trace time less T_span.

T_span: The time span of interest selected from the trace file.

out_file: The name of the file which is to receive the extracted trace segment.

Example Input

extract.pl BC-pAug89.TL 0.01 0.05 0.02 BC_extr

Here, extract.pl will extract a 0.03 second segment from trace file BC-pAug89.TL beginning at t=0.04. This includes a 0.01 second warm-up time as well as the 0.02 second interval of interest which follows the intended start time of 0.05 seconds. The user has specified that file BC_extr should receive the trace segment.

Example Output

The first 14 trace file records have been deleted to save space.

```
blanca.uccs.edu% head -50 BC-pAug89.TL_short       0.031608   162
   0.035844   174
   0.038468   162
   0.042524   174
   0.044044   150
   0.045324   162
   0.047296    90
   0.049248   174
   0.050360   150
   0.052184   162
   0.053820    90
   0.056356   174
   0.059044   162
   0.063028   174
   0.065900   162
   0.070032   174
   0.072760   162
   0.076728   174
   0.079616   162
   0.083732   174

blanca.uccs.edu% extract.pl BC-pAug89.TL_short  0.01 0.05 0.02 BC_extr

Please wait .. extracting bursty region from LAN trace file

Bursty region from "BC-pAug89.TL_short" extracted to "BC_extr"

blanca.uccs.edu% more BC_extr
   0.042524   174
   0.044044   150
   0.045324   162
   0.047296    90
```

```
0.049248   174
0.050360   150
0.052184   162
0.053820    90
0.056356   174
0.059044   162
0.063028   174
0.065900   162
```

4.2.11. Script gen_cbr.pl

gen_cbr.pl is a utility capable of generating constant bit rate (CBR) traffic. This script is useful in verifying proper simulator operation with table-driven input.

Name

gen_cbr.pl - a Perl script

Syntax

gen_cbr.pl start_time period(sec) byte_count intervals

Program Description

Perl script, gen_cbr.pl, generates a table of (time, byte_count) records in the Bellcore format. The purpose of this utility is to generate traffic which is constant bit rate, simulating packets which arrive at equal intervals, and having constant byte_count. The trace file so generated is named: CBR_trace.

Argument Descriptions

start_time: The time value for the first record in CBR_trace.

period: Time interval (seconds) between arriving packets.

byte_count: The total number of bytes arriving in each packet.

intervals: The total number of records to be generated in CBR_trace. Note that the time span covered by CBR_trace is simply: period * intervals.

Example Input / Output

Here, gen_cbr.pl will generate the equivalent of 10 packets, arriving at 5 millisecond intervals. The packet stream will start at t=36300 seconds and have a duration of 0.05 seconds. Each arriving packet contains 64 bytes. The output is written to file CBR_trace.

```
blanca.uccs.edu% gen_cbr.pl   36300    0.005    64   10

CBR trace CBR_trace created

blanca.uccs.edu% head -10 CBR_trace
36300.000000   64
36300.005000   64
```

36300.010000 64
36300.015000 64
36300.020000 64
36300.025000 64
36300.030000 64
36300.035000 64
36300.040000 64
36300.045000 64

4.2.12. Modified NIST ATM Simulator

sim is a ATM network simulator for analyzing the behavior of ATM networks.

Name

sim - an ATM simulator

Syntax

sim [-x] [-s seed] [configfile [stoptime]]

Program Description

sim provides X-window GUI for creating network topologies, controlling component parameters, measuring network activities, and logging data from simulation runs. The simulator can be run with the X-window GUI turned off. It was developed by Nada Golmie, Alfred Koening, and David Su at National Institute of Standards and Technology. We modified one of their component modules to be able to read in network trace and generate the corresponding cell arrival events. The cell data structure was modified to include cell ID information so that the cell delay, delay jitter, and cell loss statistics can be easily extract from the log file.

The current version of NIST ATM simulator uses the 4 byte integer for the simulation time (tick) and it is not adequate for longer ranger simulation. It currently has a segmentation fault triggered by __shtab() and the simulation runs for the hot region were terminated prematurely. Although we have several cells arrived at the destination of the VP connections, the samples are not big enough for reporting the cell delay and delay jitter statistics. We are working with the designers of NIST ATM simulator and hope the bug can be fixed in their version 2.0 release.

Argument Descriptions

-x	Used for running the simulator in background mode (without using X windows). With this option the configfile must be specified. Also, the configfile specified should be a "snapshot" that some parameters logged to disk so that the simulator run produces some results.
-s	Allow the user to specify the seed for the random number generator. If this option is omitted the current time (in UNIX format) is used as the seed. Specifying a particular seed is useful if identical results are expected from successive simulator runs.
Configfile	A file describing the configuration of the network to simulate. Such a file is produced by the SAVE and SNAP commands in the simulator.

Stoptime Length of time (in microseconds of simulated time) for the simulation run. Most useful when running no-interactive with the -x option. When the simulator stops, it will automatically produce a "snap" of its current state.

Example Input

sim -x hr1 60000000

Here, sim will take a network configure file, hr1, as input, run the ATM network simulation for 60 seconds, and produce a log file, sim_<n>, where n is the ID of process created to run the simulation. The hr1 file contains the description of an ATM network topology, several VP connections, a network trace file, oe_178.hr, of a hot region.

Example Output

Here is the sample output of the log file, sim_log.301. The record in the log file has the following fields: LogID Timestamp ComponentID VCI CellID. The logID=999 is used for ATMROS QoS analysis.

```
# 1 'link1' 'Link rate (Mbit/s) to switch1'
# 2 'BB1' 'Link rate (Mbit/s) to switch2'
# 3 'link3' 'Link rate (Mbit/s) to bte3'
1 0 0
2 0 0
3 0 0
MSG 0 oe_178.hr open trace file oe_178.hr

MSG 0 oe_178.hr total number of cells generated for this trace=3223

999 0 oe_178.hr 0 0
999 0 oe_178.hr 0 1
999 0 oe_178.hr 0 2
999 1335599 oe_178.hr 0 3
999 1335599 oe_178.hr 0 4
999 2002838 endconn 0 0
999 2003384 endconn 0 1
999 2003930 endconn 0 2
...
999 2327299 oe_178.hr 0 5
999 2327299 oe_178.hr 0 6
999 2501299 oe_178.hr 0 7
...
999 3328799 endconn 0 3
999 3329082 endconn 0 4
999 3329365 endconn 0 5
999 3502700 endconn 0 6
999 3502983 endconn 0 7
0 4274644 switch1  BB1 2 0
999 4274907 link5 2 0
0 4276828 switch1  BB1 2 0
999 4277091 link5 2 0
999 4317472 link5 2 0
0 4317855 switch1  link5 2 0
```

999 4318041 BB1 2 0

The above sample file shows that the network trace file oe_178.tr was read and generated 3223 cell arrival event as one of the traffic source of the ATM network simulation. The first 3 cells all arrived at time 0 and belonged to the same packet. The endconn is the component ID of the destination of the LAN VP connection. The first 3 cells departed at 2.002838, 2.003384, and 2.003930 seconds. The cells 4 and 5 are part of another packet and arrived at 1.335599 seconds. They departed at 3.328799 and 3.329082 seconds. This network has two ATM switches connected by long OC-3 transmission trunks (causing part of the long delay).

4.2.13.Script qos.pl

qos.pl This is a simulator post-processing tool which calculates cell delays across the network, delay distributions, and cell loss ratios as measures of each run's QoS.

Name

qos.pl - a Perl script

Syntax

qos.pl <log_file> <trace_file>

Program Description

Perl script, qos.pl, operates on the log file which results from a normal NIST ATM simulation, and generates output relating to Quality of Service (QoS) of that simulation run. Log file names begin with 'sim_log' and have a numeric extension, as in sim_log.2409. The two output files are generated by qos.pl are called: qos.dat and <trace_file>.dat. qos.dat reports the average cell delay and delay variance. <trace_file>.dat reports the arrival time and cell payload size (at 48 bytes/cell) using the same Bellcore trace format.

Argument Descriptions

log_file: The name of the sim_log.abcd file generated as a normal part of the NIST simulation. Note, the NIST simulator has been modified to provide additional QoS related cell arrival and departure times, which permits calculation of QoS related statistics.

trace_file: The name of the LAN trace used to drive the NIST simulator. The trace file is assumed to have standard Bellcore (time, byte_count) format.

Example Input

qos.pl sim_log.2409 OctExt.TL

Here, qos.pl will operate on NIST log file sim_log.2904, and produce a QoS statistics report on the run which generated the log file. The name of the trace file, OctExt.TL, is used to label the QoS report. Reports will be generated in files: OctExt.TL.out and qos.dat.

Example Output

total number of cells generated by input trace=8633

.

Average Delay = 1496294.88571429.

Delay Variant = 7562922641.97265.
Delay Standard Deviation = 86965.065641168.
No. of Cells Lost = 0.

4.3. Important observations and Future work

Establishing a Burstiness Metric is an interesting but time-consuming effort. In this research we have found that the Hurst constant, H, is not an effective metric for identifying local bursty regions in a network trace file. Local bursty areas can be identified effectively using either Peak/Average with floor or with variance with floor. However, there are unanswered questions. For example, no real customers or networks evaluated. Burstiness metrics still do not identify some areas which are very bursty (refer to 9-set plot of OctExt.TL.)

We conjecture that a scheme which segments the entire trace interval to be simulated and weights these sections according to their local burstiness will result in a burstiness metric for long trace intervals. Such a metric should exhibit some of the characteristics of the Hurst constant, H. Specifically, a segmented/weighted burstiness metric would rise and fall incrementally, similar to H, according to the burstiness of segments which are appended to some base interval. This type of metric would not be good for identifying burstiness of local regions, however, for the same reasons given in the evaluation of the Hurst constant.

One useful future work would be to generate families of curves of needed switch bandwidth vs. traffic load/burstiness. This may not be possible on a "per simulation" basis, but as a result of summarizing the results of multiple similar simulation runs. The other direction would be to have a detailed comparison analysis of Segmented/weighted burstiness metric with Hurst constant, H. By improving the modified NIST ATM simulator for longer range simulation and integrating efficient search routine, ATMROS can be used to suggest cost-effective bandwidth for a network customer subscribing an ATM network service.

5. Evaluation

The success of the proposed project can be assessed by the usefulness and performance of software modules provided in ATMROS. Based on the results we presented in Section 4, the project is quite successful since we now have a set of scripts for visualizing and analyzing the network traces, and for identifying and extracting the hot regions in a trace. They can assist network planners or managers to analyze network traffic. We also contribute the understanding of the burstiness property in the network traces by investigating the effectiveness of several metrics, including the Hurst constant. We lay down ground work for a modified ATM simulator for reading in a real network trace and produced the scripts for analyzing the QoS statistics of the selected trace over the ATM connections. By improving the modified NIST ATM simulator for longer range simulation and integrating efficient search routine, ATMROS can be used to suggest optimal bandwidth for a network customer subscribing an ATM network service. They augment the library of tools that can foster the research, education, and development efforts in the area of network resource optimization and traffic management.

US West will provide feedback on the ATMROS usage in their research and network management organizations and the userfulness of the report. Those feedback will further indicate the degree of success of the project.

6. Intellectual property developed under sponsorship of this grant.

We have designed and implemented ATMROS software system with modules that can be used to analyze the network traces and extract hot regions for further study. They can be licensed to companies in telecommunications industry that operate or plan networks. To obtain the source code of the ATMROS system, send email to chow@quandary.uccs.edu.

7. Technology Transfer

We have given presentation of our research results to researcher at US West Advanced Technologies twice and got valuable feedback from them. Based on that, we have emphasized on the analysis of network traces and build tools for visualizing and extracting hot regions. We have delivered the ATMROS software to US West Advanced Technologies.

Acknowledgment

We would like to thank Dr. Nada Gomie of NIST for helping with NIST ATM simulator, Dr. Vern Paxon of Lawrence Berkeley National Lab for giving us pointers on self-similar traffic and whittle code for estimating the Hurst parameter. Dr. Walter Willinger of Bellcore for sharing information on self-similar traffic. Heikki Julkunen helped with the installation and debugging of NIST ATM simulator on the Alpha Workstation.

8. Appendix - A Bibliography

8.1. *Hardware/Software Stacks*

[Traw93] C. Traw, J. Smith, "Hardware/Software Organization of a High-Performance ATM Host Interface", IEEE Journal on Selected Areas in Communications, vol. 11, no.2, 1993, pp. 240-253.

[Moor93] T. Moors, A. Cantoni, "ATM Receiver Implementation Issues", IEEE Journal on Selected Areas in Communications, vol. 11, no. 2, 1993, pp. 254-263.

[Veni93] I. Venieris, J. Angelopoulos, G. Stassinopoulos, "Efficient Use of Protocol Stacks for LAN/MAN-ATM Interworking", IEEE Journal on Selected Areas in Commun., vol. 11, no. 8, 1993, pp. 1160-1171.

8.2. *Performance Modeling*

[Kawa95] K. Kawahara, Y. Oie, M. Murata, H. Miyahara, "Performance Analysis of Reactive Congestion Control for ATM Networks", IEEE Journal on Selected Areas in Communications, vol. 13, no. 4, 1995, pp. 651-661.

[Bian95] R. Bianchini, H.S. Kim, "The Tera Project: A Hybrid Queueing ATM Switch Architecture for LAN", IEEE Journal on Selected Areas in Communications, vol. 13, no. 4, 1995, pp. 673-???.

[Reib95] A. Reibman, A. Berger, "Traffic Descriptors for VBR Video Teleconferencing Over ATM Networks", IEEE/ACM Transactions on Networking, vol. 3, no. 3, 1995, PP. 329-339.

[Cao95] Xi-Ren Cao, Don Towsley, "A Performance Model for ATM Switches with General Packet Length Distributions", IEEE/ACM Transactions on Networking, vol. 3, no. 3, 1995, pp. 299-309.

[Yege94] F. Yegenoglu, B. Jabbari, "Maximum Likelihood Estimation of ATM Traffic Model Parameters", IEEE GLOBECOM Mini-Conference, CTMC01 01.7, 1994, pp. 34-38.

[Rubi94] I. Rubin, A. Ratkovic, "Throughput Analysis of Input Queueing ATM Switches Under Imbalanced Input Traffic Loading", IEEE GLOBECOM Mini-Conference, CTMC01 04.4, 1994, pp. 127-131.

[Koll94] E. Kollias, G. Stassinopoulos, "ATM Performance Evaluation Under Transparencies of a Distributed System Environment (DSE)", IEEE Journal on Selected Areas in Communications, vol. 12, no. 6, 1994, pp. 1059-1071.

[Sale94] M. Saleh, M. Atiquzzaman, "Queueing Analysis of Shared Buffer Switches for ATM Networks", IEEE GLOBECOM, vol. 2, 1994, pp. 1070-1074.

[Tass94] L. Tassiulas, Y. Chung Hung, S. Panwar, "Optimal Buffer Control During Congestion in an ATM Network Node", IEEE/ACM Trans. on Networking, vol. 2, no. 4, 1994, pp. 374-386.

[Wang93] Q. Wang, V.S. Frost, "Efficient Estimation of Cell Blocking Probability for ATM Systems", IEEE/ACM Trans. on Networking, vol. 1, no. 2, 1993, pp. 230-235.

[Deve93] M. Devetsikiotis, J.K. Townsend, "Statistical Optimization of Dynamic Importance Sampling Parameters for Efficient Simulation of Communication Networks", IEEE/ACM Trans. on Networking, vol. 1 no. 3, 1993, pp. 293-305.

[Bian93] G. Bianchi, J.S. Turner, "Improved Queueing Analysis of Shared Buffer Switching Networks", IEEE/ACM Trans. on Networking, vol. 1, no. 4, 1993, pp. 482-490.

[Card93] M. Cardona, T. Satake, S. Tsujii, "An Algebraic Model for Computing the Maximum Throughput of Piplined Protocol Processors", IEEE GLOBECOM, vol. 3, 1993, pp. 1827-1833.

[Turn93] J.S. Turner, "Queueing Analysis of Buffered Switching Networks", IEEE Trans. on Communications, vol. 41, no.2, 1993, pp. 412-420.

8.3. *Quality of Service (QoS) and other Metrics*

[Besh94] M. Beshai, R. Kositpaiboon, J. Yan, "Interaction of Call Blocking and Cell Loss in an ATM Network", IEEE Journal on Selected Areas in Communications, vol. 12, no. 6, 1994, pp. 1051-58.

[Jung93] J. Jung, A. Gravey, "QoS Management and Performance Monitoring in ATM Networks", IEEE GLOBECOM, vol. 2, 1993, pp. 708-712.

[vand93] K. vanderWal, M. Dirksen, D. Brandt, "Implementation of a Policing Criterion Calculator based on the Leaky Bucket Algorithm", IEEE GLOBECOM, vol. 2, 1993, pp. 713-718.

[Chen95] W-T Chen, C-F Huang, S-C Ding, "A Scheme for QOS Control in ATM Switching Systems", IEEE Int. Conf. on Commun., vol. 1, 1995, pp. 196-200.

[Shio95] S. Shioda, H. Saito, "Satisfying QOS Standard with Combined Strategy for CAC and UPC", IEEE Int. Conf. on Commun., vol. 2, 1995, pp. 965-969.

[Tran95] Vu Tran, "An Adaptive Model for the Real-Time Management of Quality-of-Service in the OSI Reference Model", IEEE Int. Conf. on Commun., vol. 2, 1995, pp. 975-981.

[Mill96] G.J. Miller, S. Dastangoo, The MITRE Corp., translated Whittle's mean likelihood estimator for the Hurst constant (*H*) from S-Plus to C-code.
URL: ftp://foghorn.ie.org/pub/selfsim/whittle.tar.Z.

8.4. *Switch Architecture*

[Chao95] H. Jonathan Chao, Byeong-Seog Choe, "Design and Analysis of a Large-Scale Multicast Output Buffered ATM Switch", IEEE/ACM Transactions on Networking, vol. 3 no. 2, 1995, pp. 126-138.

[Lee95] M. Lee, D. Ahn, "Cell Loss Analysis and Design Trade-Offs of Nonblocking ATM Switches with Nonuniform Traffic", IEEE/ACM Transactions on Networking, vol. 3 no. 2, 1995, pp. 199-210.

[Patt94] A. Pattavina, C Bison, "Connectionless Switching by Asynchronous Banyan Networks", IEEE GLOBECOM, vol. 1, 1994, pp. 441-447.

[Chan94] C. Chang, A. Paulraj, T. Kailath, "A Broadband Packet Switch Architecture with Input and Output Queueing", IEEE GLOBECOM, vol. 1, 1994, pp. 448-452.

[Vish94] M. Vishnu, J.W. Mark, "ATM Switching Node Design Based on a Versatile Traffic Descriptor", IEEE GLOBECOM, vol. 1, 1994, pp. 111-116.

[Nade94] Nader Mirfakhraei, "Performance Analysis of a Large-Scale Switching System for High-Speed ATM Networks", IEEE GLOBECOM, vol. 1, 1994, pp. 144-149.

[Paol94] Paolo Giacomazzi, "Channel Grouping Techniques in the Tandem Banyan Switching Fabric", IEEE GLOBECOM, vol. 1, 1994, pp. 302-308.

[Chow94] S. Chowdhury, B. Sengupta, "Threshold-based Load Balancing in an ATM Switch Consisting of Parallel Output Buffered Switch Elements", IEEE GLOBECOM, vol. 1, 1994, pp. 463-469.

[Tao94] Z. Tao, S. Cheng, "A New Way to Share Buffer .. Grouped Input Queueing in ATM Switching", IEEE GLOBECOM, vol. 1, 1994, pp. 475-479.

[Park94] Young-Keun Park, V. Cherkassky, Gyungho Lee, "Omega Network-Based ATM Switch with Neural Network-Controlled Bypass Queueing and Multiplexing", IEEE Journal on Selected Areas in Communications, vol. 12, no. 9, 1994, pp. 1471-1480.

[Chen93] D.X. Chen, J.W. Mark, "SCOQ: A Fast Packet Switch with Shared Concentration and Output Queueing", IEEE/ACM Trans. on Networking, vol. 1, no. 1, 1993, pp. 142-151.

[Patt93] A. Pattavina, G. Bruzzi, "Analysis of Input and Output Queueing for Nonblocking ATM Switches", IEEE/ACM Trans. on Networking, vol. 1, no. 3, 1993, pp. 314-327.

[Copp93] P. Coppo, M. D'Ambrosio, R. Melen, "Optimal Cost/Performance Design of ATM Switches", IEEE/ACM Trans. on Networking, vol. 1 no. 5, 1993, pp. 566-575.

[Merc93] G. Mercankosk, Z. Budrikis, A. Cantoni, "Extended Distributed Queueing for Integrated Services", IEEE Journal on Selected Areas in Communications, vol. 11, no. 8, 1993, pp. 1193-1201.

[Ding93] J. Ding, H. Jiang, "Modeling a Class of Priority-Based ATM Communication Switch Designs", IEEE GLOBECOM, vol. 2, 1993, pp. 729-733.

[Wrig93] G. Wrigley, M. Izzo, S. Farkouh, "Multilayer Operations Functional Analysis for Broadband ATM Networks and Services", IEEE GLOBECOM, vol. 3, 1993, pp. 1536-1542.

[Toki93] I. Tokizawa, H. Ueda, K. Kikuchi, "ATM Transport System Architecture and Field Trial",

IEEE GLOBECOM, vol. 3, 1993, pp. 1449-1453.
[Kuma96] Sanjeev Kumar, Dharma P. Agrawal, "On Multicast Support for Shared-Memory-Based ATM Switch Architecture", IEEE Network, Jan/Feb, 1996, pp. 34-39.
[Simc95] Robert J. Simcoe, Tong-Bi Pei, "Perspectives on ATM Switch Architecture and the Influence of Traffic Pattern Assumptions on Switch Design", ACM SIGcomm Computer Commun. Review, vol. 25, no. 2, 1995, pp. 93-105.
[Vela93] R. Velamuri, P. Landsberg, C. Zukowski, "A Multi-Queue Flexible Buffer Manager Architecture", IEEE GLOBECOM, vol. 3, 1993, pp. 1401-1405.
[Puli95] A. Puliafito, M. Balakrishnan, K. Trivedi, I. Viniotis, "Buffer Sizing for ABR Traffic in an ATM Switch", IEEE Int. Conf. on Commun., vol. 1, 1995, pp. 316-320.
[Ahn95] D. Ahn, Sang Lee, M.J. Lee, "Effective Cell Loss Analysis of a Nonblocking ATM Switch with Nonuniform Traffic", IEEE Int. Conf. on Commun., vol. 1, 1995, pp. 321-325.
[Chen95] W-T Chen, W-Y Hwang, "A Simple High-Performance Priority Scheme for ATM Switching Systems", IEEE Int. Conf. on Commun., vol. 2, 1995, pp. 955-959.
[Hart95] F. Hartanto, H. Sirisena, K. Pawlikowski, "Protective Buffer Policies at ATM Switches", IEEE Int. Conf. on Commun., vol. 2, 1995, pp. 960-964.

8.5. *Traffic Analysis*

[Armi95] G.J. Armitage, K.M. Adams, "How Inefficient is IP over ATM Anyway?", IEEE Network, Jan/Feb, 1995, pp. 18-26.
[Bono95] F. Bonomi, K. Fendick, "The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service", IEEE Network, Mar/Apr, 1995, pp. 25-39.
[Kung95] H. Kung, R. Morris, "Credit-Based Flow Control for ATM Networks", IEEE Network, Mar/Apr, 1995, pp. 40-48.
[Rama95] K. Ramakrishnan, P. Newman, "Integration of Rate and Credit Schemes for ATM Flow Control", IEEE Network, Mar/Apr, 1995, pp. 49-56.
[Roma95] A. Romanow, S. Floyd, "Dynamics of TCP Traffic over ATM Networks", IEEE Journal on Selected Areas in Communications, vol. 13, no. 4, 1995, pp. 633-641.
[Paxo94] V. Paxson, "Empirically Derived Analytic Models of Wide Area TCP Connections", IEEE/ACM Trans. on Networking, vol. 2, no. 4, 1994, pp. 316-336.
[Roma93] Allyn Romanow, "TCP over ATM: Some Performance Results", ATM Forum Traffic Management Sub-Working Group, ??, 1993. ATM Forum/93-784 July 1993
[Vill94] Curtis Villamizar, Cheng Song, "High Performance TCP in ANSNET", ACM SIGcomm Computer Commun. Review, vol. 24, no. 5, 1994, pp. 45-60.
[Papa93] C. Papadopoulous, G.M. Parulkar, "Experimental Evaluation of SUNOS IPC and TCP/IP Protocol Implementation", IEEE/ACM Trans. on Networking, vol. 1, no. 2, 1993, pp. 199-216.
[Lela94] W. Leland, Taqqu, W. Willinger, D. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)", IEEE/ACM Trans. on Networking, vol. 2, no. 1, 1994, pp. 1-15.
[Garr94] Mark W. Garrett, Walter Willinger, "Analysis, Modeling and Generation of Self-Similar VBR Video Traffic", ACM SIGCOMM Computer Commun. Review, vol. 24, no.4, 1994, pp. 269-280.
[Bera95] J. Beran, R. Sherman, Taqqu, W. Willinger, "Long-Range Dependence in Variable-Bit-Rate Video Traffic", IEEE Trans. on Communications, vol.43, no.2/3/4, 1995, pp. 1566-1579.
[Lee90] D-S Lee, S-Q Li, K-H Tzou, "Analysis of Video Packet Loss in ATM Networks", IEEE GLOBECOM, vol. 2, 1990, pp. 857-861.
[Claf95] K. Claffy, Hans-Werner Braun, G. Polyzos, "A parameterizable methodology for Internet traffic flow profiling", IEEE Journal on Selected Areas in Communications, vol. 13, no. 8, 1995, pp. 1481-94.
[Schm93] A. Schmidt, R. Campbell, "Internet Protocol Traffic Analysis with Applications for ATM Switch Design", ACM SIGcomm Computer Commun. Review, vol. 23, no. 2, 1993, pp. 39-52.
[Paxo94] Vern Paxon, "Growth Trends in Wide-Area TCP Connections", IEEE Network, July/August, 1994, pp. 8-17.
[Li94] San-qi Li, "New Concepts and Approaches for Traffic Analysis and Network Control in ATM Networks", Slide Presentation via *US*West (1994).

[Alqa95] A. M. Alqaed, C.H. Chang, "Traffic Description Using Spectral Characterization of Wide-band Input Processes in ATM Networks", IEEE Int. Conf. on Commun., vol. 1, 1995, pp. 182-185.
[Au95] Tat-Ming Au, Hassan Mehrpour, "On the Worst Data Loss Behavior in Statistical Multiplexing", IEEE Int. Conf. on Commun., vol. 1, 1995, pp. 309-315.

8.6. *Traffic Management*

[Floy94] Sally Floyd, "TCP and Explicit Congestion Notification", ACM SIGcomm Computer Commun. Review, vol. 24, no. 5, 1994, pp. 8-23.
[Ozve95] C. Ozveren, R. Simcoe, G. Varghese, "Reliable and Efficient Hop-by-Hop Flow Control", IEEE Journal on Selected Areas in Communications, vol. 13, no. 4, 1995, pp. 642-650.
[Jong94] S. Jong, Y. Chin, "Congestion Control with the Double and Hysteresis Threshold in ATM Networks", IEEE GLOBECOM, vol. 1, 1994, pp. 595-599.
[Tipp94] Tipper, Sharma, Khetan, Balakrishnan, Menon, "An Analysis of the Congestion Effects of Link Failures in Wide Area Networks", IEEE Journal on Selected Areas in Communications, vol. 12, no. 1, 1994, pp. 179-192.
[Ndou94] Thomas D. Ndousse, "Fuzzy Neural Control of Voice Cells in ATM Networks", IEEE Journal on Selected Areas in Communications, vol. 12, no. 9, 1994, pp. 1488-1494.
[Kola94] A. Kolarov, G. Ramamurthy, "Comparison of Congestion Control Schemes for ABR Service in ATM Local Area Networks", IEEE GLOBECOM, vol. 2, 1994, pp. 913-918.
[Fang94] C. Fang, H. Chen, J. Hutchins, "A Simulation Study of TCP Performance in ATM Networks", IEEE GLOBECOM, vol. 2, 1994, pp. 1217-1223.
[Jens94] D. Jensen, "B-ISDN Network Management By A Fuzzy Logic Controller", IEEE GLOBECOM, vol. 2, 1994, pp. 799-804.
[Veci94] G. de Veciana, "Leaky Buckets and Optimal Self-Tuning Rate Control", IEEE GLOBECOM, vol. 2, 1994, pp. 1207-1211.
[Wu94] J.C. Wu, T. Huang, "Dynamic Priority Schemes to Improve the QOS in ATM Networks", IEEE GLOBECOM, vol. 2, 1994, pp. 1195-1199.
[Abde94] M. Abdelaziz, I. Stavrakakis, "Open- and Closed-Loop Traffic Regulation Schemes for ATM Networks", IEEE GLOBECOM, vol. 2, 1994, pp. 919-923.
[Oser94] T.M. Oser, X. Gu, D.R. Vaman, "Routing with Admission Control in ATM Networks", IEEE GLOBECOM, vol. 2, 1994, pp. 1212-1216.
[Cido94] I. Cidon, R. Guerin, A. Khamisy, "On Protective Buffer Policies", IEEE/ACM Trans. on Networking, vol. 2, no. 3, 1994, pp. 240-246.
[Bond94] A. Bonde, Jr., S. Ghosh, "A Comparative Study of Fuzzy Versus 'Fixed' Thresholds for Robust Queue Management in Cell-Switching Networks", IEEE/ACM Trans. on Networking, vol. 2, no. 4, 1994, pp. 337-344.
[Rose94] C. Rosenberg, B. Lague, "A Heuristic Framework for Source Policing in ATM Networks", IEEE/ACM Trans. on Networking, vol. 2, no. 4, 1994, pp. 387-397.
[Khan93] Irfan Khan, Victor O. K. Li, "A Traffic Control Mechanism for ATM Networks", IEEE GLOBECOM, vol. 2, 1993, pp. 1122-1126.
[Veci94] G. de Veciana, G. Kesidis, "Bandwidth Allocation for Multiple Qualities of Service Using Generalized Processor Sharing" , IEEE GLOBECOM, vol. 3, 1994, pp. 1550-1554.
[Zuke94] M. Zukerman, S. Chan, "Congestion Control by Maintaining Fairness in High Speed Data Networks", IEEE GLOBECOM, vol. 3, 1994, pp. 1576-1580.
[Kons94] T. Konstantopoulos, V. Anatharam, "Optimal Flow Control Schemes for ATM Networks", IEEE GLOBECOM, vol. 3, 1994, pp. 1763-1767.
[Vaki93] F. Vakil, "A Capacity Allocation Rule for ATM Networks", IEEE GLOBECOM, vol. 1, 1993, pp. 406-416.
[Elwa93] A. Elwalid, I. Widjaja, "Efficient Analysis of Buffered Multistage Switching Networks Under Bursty Traffic", IEEE GLOBECOM, vol. 2, 1993, pp. 1072-1078.
[Lee93] M. Lee, D. Ahn, "Packet Loss Analysis of Nonblocking ATM Switches with Nonuniform Traffic and Performance Improvement by Output Buffer Sharing", IEEE GLOBECOM, vol. 2, 1993, pp. 1079-1084.
[Yin93] N. Yin, M. Hluchyj, "Simple Models for Statistically Multiplexed Data Traffic

in Cell Relay Networks" IEEE GLOBECOM, vol. 2, 1993, pp. 824-829.
[Chou93] A. Choudhury, E. Hahne, "Space Priority Management in a Shared Memory ATM Switch", IEEE GLOBECOM, vol. 3, 1993, pp. 1375-1383.
[Coll93] B. Collier, H. Kim, "Effect of Multiplexed Sub-ATM Rate Inter-LAN Traffic on Input Queueing ATM Switches", IEEE GLOBECOM, vol. 3, 1993, pp. 1861-1866.
[Vama93] D. Vaman, X. Gu, S. Kumar, X. Qian, T. Oser, "A Flow Control Strategy for ATM Networks Based on a Unified Performance Parameter", IEEE GLOBECOM, vol. 3, 1993, pp. 1822-1826.
[Thom93] G. Thomas, "On High Speed Packet Switches with Windowed Input Buffers", IEEE GLOBECOM, vol. 3, 1993, pp. 1406-1410.
[Chen95] Song Cheng, San-qi Li, Joydeep Ghosh, "Predictive Dynamic Bandwidth Allocation for Efficient Transport of Real-Time VBR Video over ATM", IEEE Journal on Selected Areas in Communications, vol. 13, no. 1, 1995, pp. 12-23.
[Turn92] J. Turner, "Managing Bandwidth in ATM Networks with Bursty Traffic", IEEE Network, September, 1992, pp. 50-58.
[Crow95] J. Crowcroft, Z Wang, A. Smith, J. Adams, "A Rough Comparison of the IETF and ATM Service Models", IEEE Network, Nov./Dec., 1995, pp. 12-16.
[Lefe96] C. Lefelhocz, B. Lyles, S. Henker, L. Zhang, "Congestion Control for Best-Effort Service: Why We Need a New Paradigm", IEEE Network, Jan./Feb., 1996, pp. 10-19
[Ohsa95] H. Ohsaki, M. Murata, H. Suzuki, C. Ikeda, H. Miyahara, "Rate-Based Congestion Control for ATM Networks", ACM SIGcomm Computer Commun. Review, vol. 25, no. 2 , '95, pp. 60-72.
[Kola95] A. Kolarov, G. Ramamurthy, "End-to-end Adaptive Rate Based Congestion Control Scheme for ABR Service in Wide Area ATM Networks", IEEE Int. Conf. on Commun., vol. 1, 1995, pp. 138-143.
[Harm95] Janelle Harms, Hong Qian, "A Call Admission Scheme for ATM Networks Based on Refinement of Traffic Measures", IEEE Int. Conf. on Commun., vol. 1, 1995, pp. 191-195.
[Tarr95] A. Tarraf, I.Habib, T. Saadawi, "Congestion Control Mechanism for ATM Networks Using Neural Networks", IEEE Int. Conf. on Commun., vol. 1, 1995, pp. 206-210.
[Park95] H-S Park, D-Y Kwak, W-S Rhee, M-Y Jeon, "Adaptive CAC Algoritnms Using Measurement and Estimation", IEEE Int. Conf. on Commun., vol. 1, 1995, pp. 211-215.
[Elwa95] Anwar I. Elwalid, "Analysis of Adaptive Rate-Based Congestion Control for High-Speed Wide-Area Networks", IEEE Int. Conf. on Commun., vol. 3, 1995, pp. 1948-1953.
[Char95] A. Charny, D. Clark, Raj Jain, "Congestion Control with Explicit Rate Indication", IEEE Int. Conf. on Commun., vol. 3, 1995, pp. 1954-1963.
[Abde95] M. Abdelaziz, I. Stavrakakis, "Study of an Adaptive Rate Control Scheme under Unequal Propagation Delays", IEEE Int. Conf. on Commun., vol. 3, 1995, pp. 1964-1973.
[Kris95] R. Krishnan, J. Silvester, "The Effect of Variance Reduction on the Performance of the Leaky Bucket", IEEE Int. Conf. on Commun., vol. 3, 1995, pp. 1974-1980.
[Ko95] C.P. Ko, O.W.W. Yang, H.T. Mouftah, "Bursty Traffic Control using Dynamic Token Allocation Method", IEEE Int. Conf. on Commun., vol. 3, 1995, pp. 1981-1985.
[Rich95] I.E.G. Richardson, M.J. Riley, "Usage Parameter Control Cell Loss Effects on MPEG Video", IEEE Int. Conf. on Commun., vol. 2, 1995, pp. 970-974.
[Doul95] C. Douligeris, G. Develekos, "A Fuzzy Logic Approach to Congestion Control in ATM Networks", IEEE Int. Conf. on Commun., vol. 3, 1995, pp. 1969-1973.
[Pate96] B.V. Patel, C.C. Bisdikian, "End-Station Performance under Leaky Bucket Traffic Shaping", IEEE Network, vol. 10 no.5, Sep/Oct 1996, pp. 40-47.
[Cope96] David Copelend, "Closed-loop flow control manages ATM channel bandwidth", EDN Nov 21, 1996, pp. 117-122.

8.7. *Recent Papers on Self-Similar Traffic*

[Norr95] Ilkka Norros, "On the Use of Fractional Brownian Motion in the Theory of Connectionless Networks", IEEE Journal on Selected Areas in Communications, vol. 13, no. 6, 1995, pp. 953-962.
[Huan95] C. Huang, M. Devetsikiotis, I. Lambadaris, A. Kaye, "Fast Simulation for Self-Similar Traffic in ATM Networks", IEEE Int. Conf. on Commun., vol. 1, 1995, pp. 438-444.
[Prut95] Parag Pruthi, "Heavy-Tailed ON/OFF Source Behavior and Self-Similar Traffic",

IEEE Int. Conf. on Commun., vol. 1, 1995, pp. 445-450.
[Norr95] Ilkka Norros, "The Management of Large Flows of Connectionless Traffic on the Basis of Self-Similar Modeling", IEEE Int. Conf. on Commun., vol. 1, 1995, pp. 451-455.
[Slim95] S. Ben Slimane, T. Le-Ngoc, "A Doubly Stochastic Poisson Model for Self-Similar Traffic", IEEE Int. Conf. on Commun., vol. 1, 1995, pp. 456-460.
[Addi95] R. G. Addie, M. Zukerman, T. Neame, "Performance of a Single Server Queue with Self-Similar Input", IEEE Int. Conf. on Commun., vol. 1, 1995, pp. 461-465.
[Lau95] W-C Lau, A. Erramilli, J. L. Wang, W. Willinger, "Self-Similar Traffic Generation: The Random Midpoint Displacement Algorithm and Its Properties", IEEE Int. Conf. on Commun., vol. 1, 1995, pp. 466-472.
[Duff95] N. Duffield, J. Lewis, N. O'Connell, "Predicting Quality of Service for Traffic with Long-Range Fluctuations", IEEE Int. Conf. on Commun., vol. 1, 1995, pp. 473-477.

8.8. *General References*

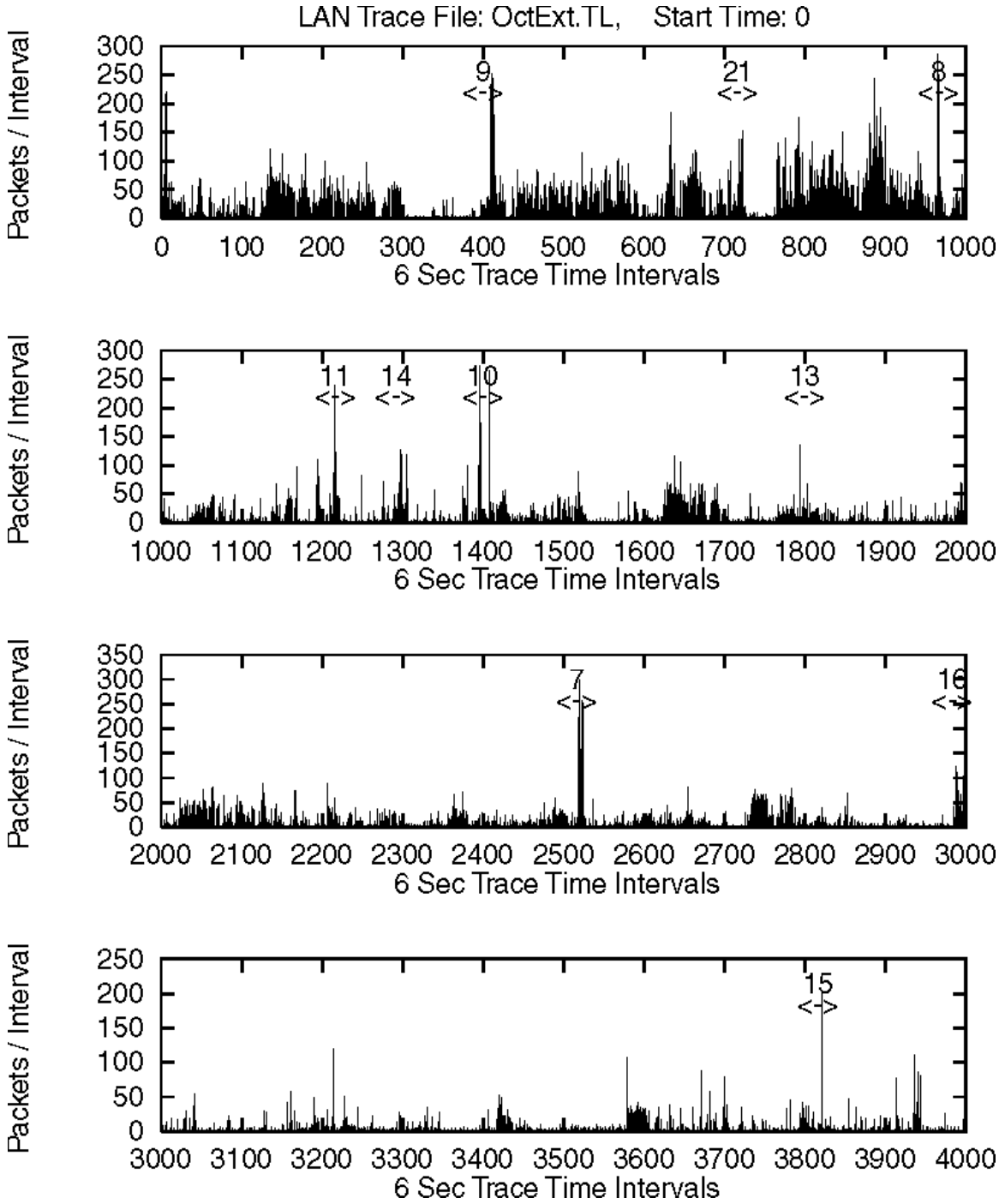
[Stal94] W. Stalllings, "ISDN and Broadband ISDN with Frame Relay and ATM", third edition, Prentice Hall, 1994. See especially Chapter 16, "ATM Traffic and Congestion Control".
[ATMF94] The ATM Forum, "ATM User-Network Interface Specification", Version 3.1, September 1994.
[Bera94] Jan Beran, "Statistics for Long-Memory Processes", Chapman & Hall, New York, 1994.

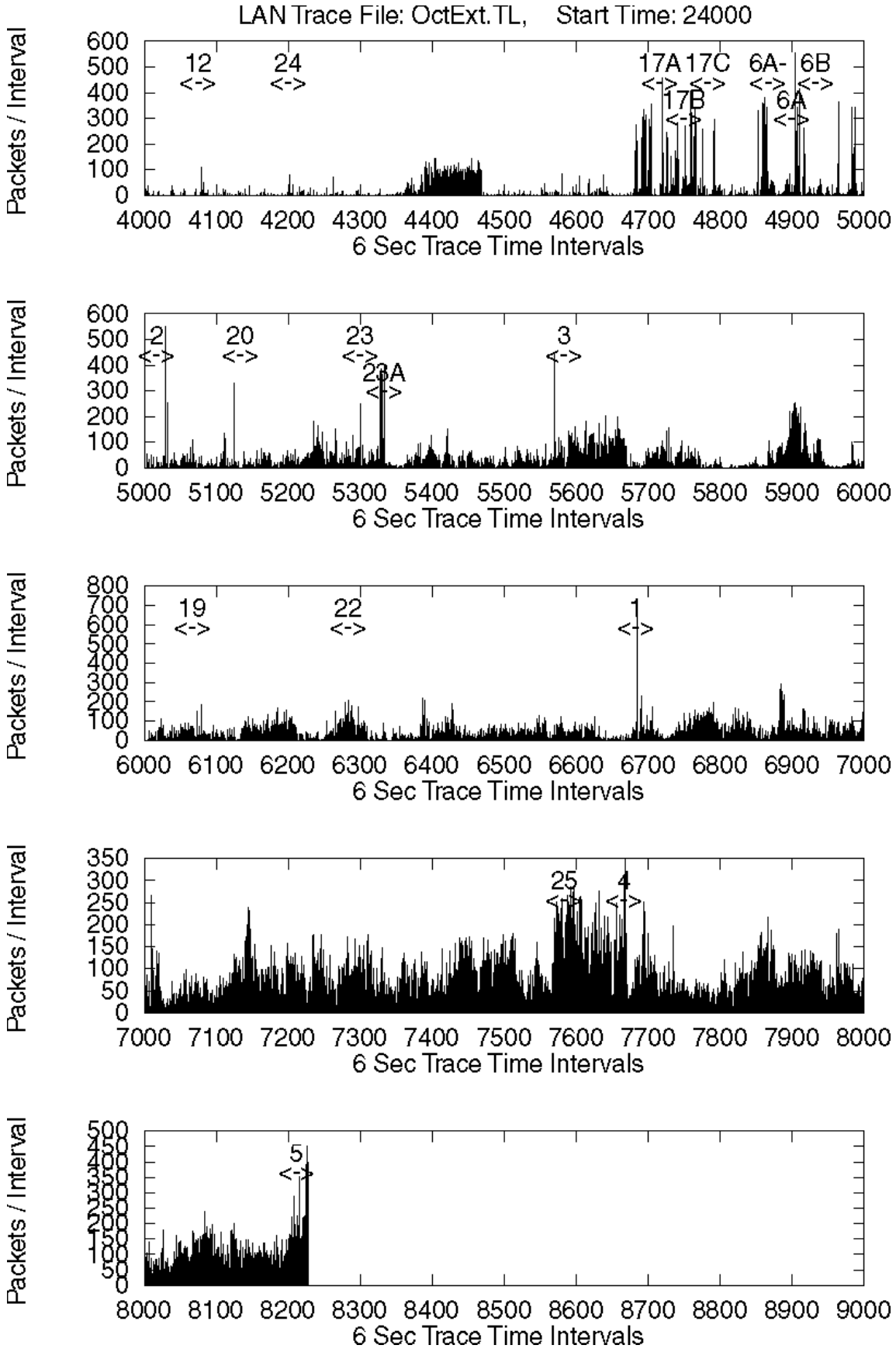
8.9. *Traffic Billing*

[Low93] S.H. Low, P.P. Varaiya, "A New Approach to Service Provisioning in ATM Networks", IEEE/ACM Trans. on Networking, vol. 1 no. 5, 1993, pp. 547-553.

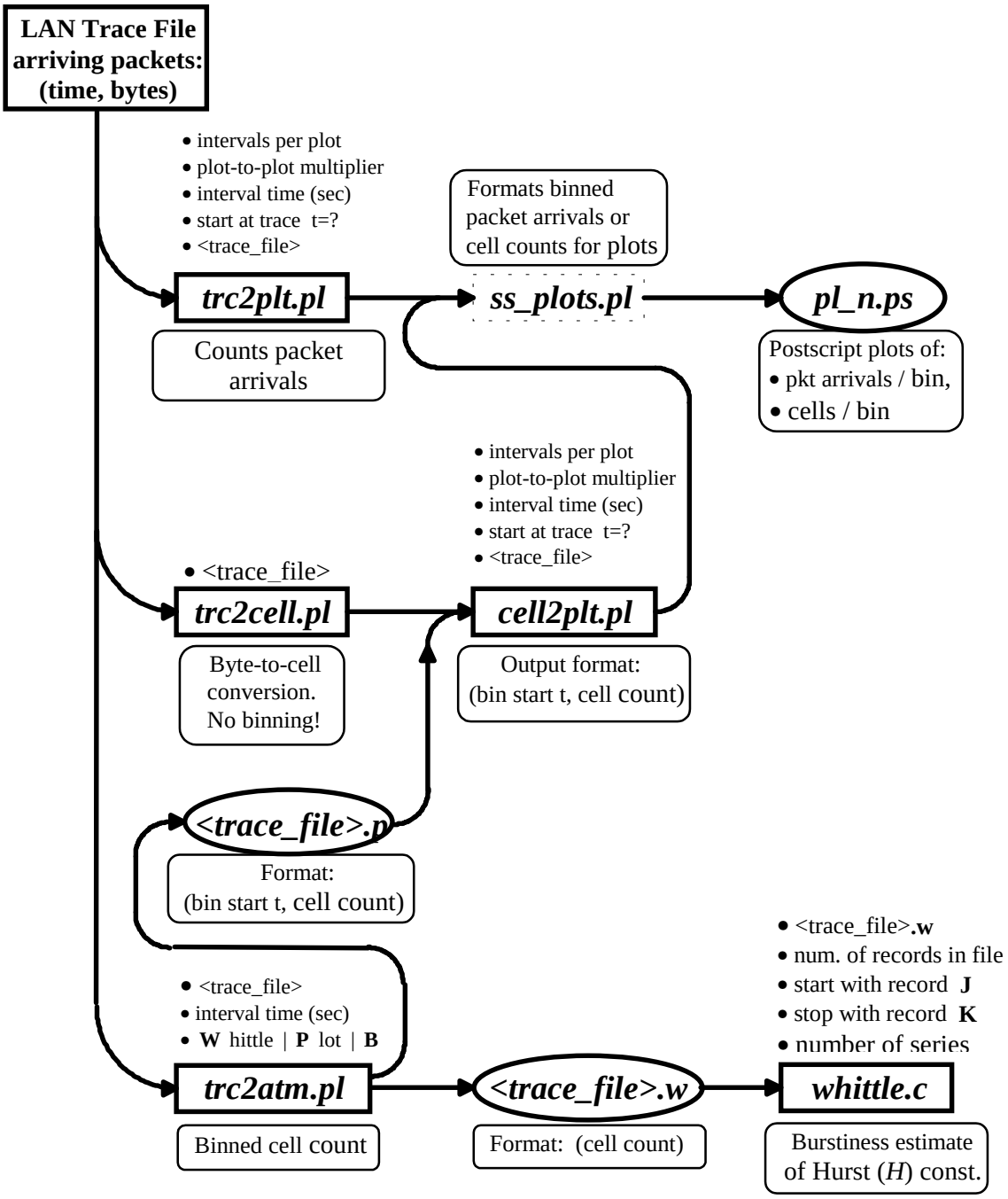
9. Appendix - B Plot of Bellcore Trace: OctExt.TL

LAN trace file OctExt.TL is shown below in 9 segments on a constant time scale of six (6) seconds per interval. To get actual trace time, multiply the indicated abscissa by 6. Bursty regions discussed in Section 4 entitled "Analysis of Bellcore Trace File: OctExt.TL" are identified by region reference number.

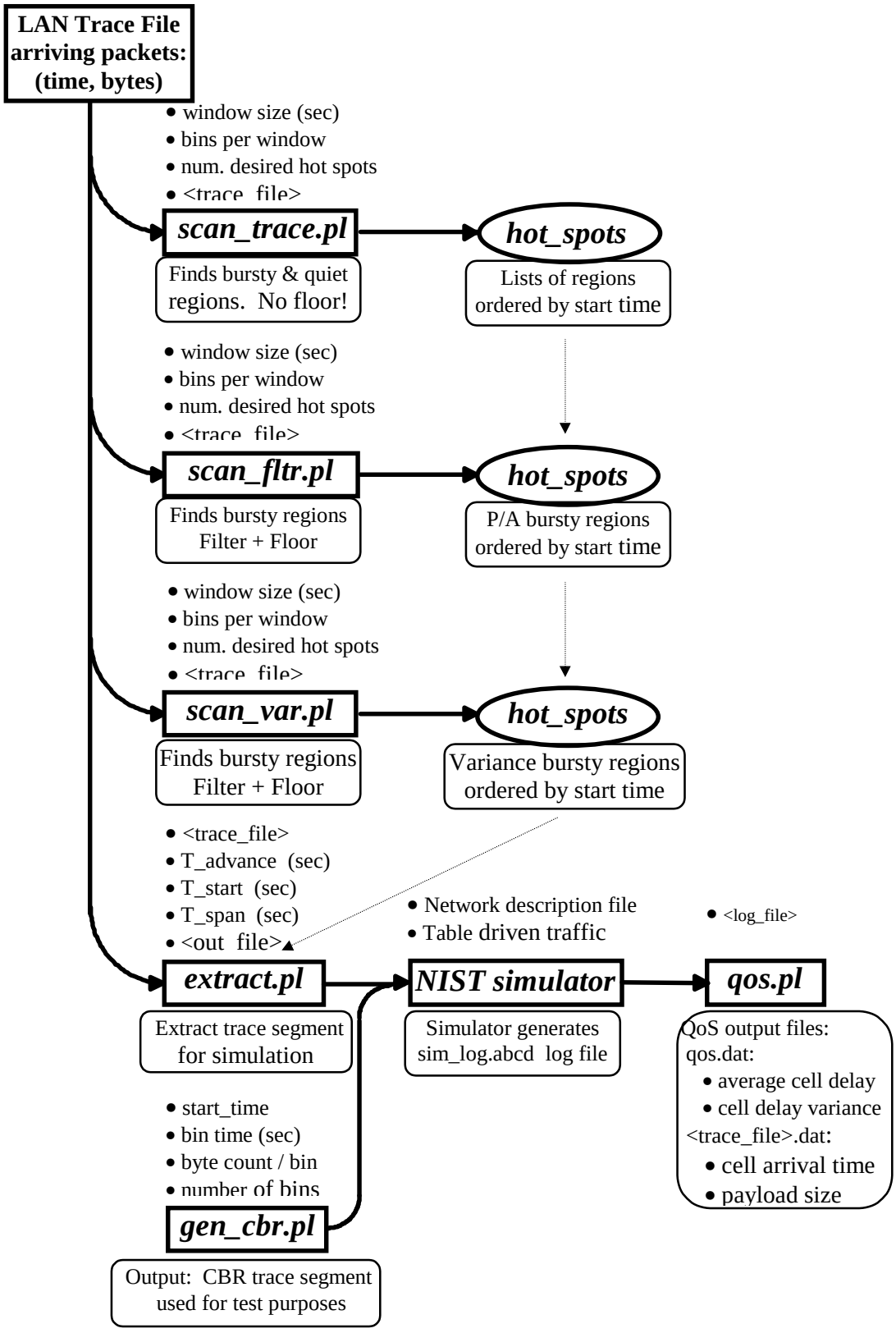



LAN Trace File: OctExt.TL, Start Time: 24000
Packets / Interval
6 Sec Trace Time Intervals
12 <->
24 <->
17A <->
17C <->
6A- <->
6B <->
17B <->
6A <->
2 <->
20 <->
23 <->
23A <->
3 <->
19 <->
22 <->
1 <->
25 <->
4 <->
5 <->

10. Appendix - C Trace Plot & Hurst Burstiness Programs



11. Appendix - D Programs to Facilitate / Verify Simulation



12. Appendix - E Install ATMROS from Archive File

This section provides the basic instructions for installing **ATMROS** system. It is assumed that the compressed tar file of **ATMROS** software, *atmros.tar.gz*, is available via ftp or cartridge. The *atmros.tar.gz* is about 8.6 MBytes. The installation requires about 61 MBytes disk space. Before attempting installation, read the README file.

12.1. *Installation Procedure*

Following these steps to install **ATMROS**:

- Create a directory for building the **ATMROS** system and assume that the path name to this directory is *$ATMROS*.
- Copy the compressed tar file, *atmros.tar.gz,* to *$ATMROS* directory.
- Change the working directory by issuing "**cd $ATMROS**" command.
- Uncompress *atmros.tar.Z* by using the "**gunzip atmros.tar.gz**" command.
- Restore the directory structure by issuing the "**tar xvf atmros.tar**" command.
- Due to the copyright reason, we can not directly distribute the whittle code implemneted by Sia Dastangoo and Greg Miller of MITRE corporation. You can ftp their whittle code from ftp://foghorn.ie.org/pub/selfsim/whittle.tar.Z, to $ATMROS/src directory and uncompress it by "uncompress whittle.tar.Z" and restore the directory by "tar xvf whittle.tar". Go to the whittle directory and execute "make". If you find it complains about the missing of /usr/include/floatingpoint.h, just delete the MAKE DEPEND section in the Makefile and recompile it. Copy the *whittle* code to *$ATMROS/bin* directory.
- Change to the *$ATMROS/src/nistsim* directory and follow the instruction in the INSTALL file. For the basic case, you need to create a directory with the architecture type of the computer, such as mips or alpha. Copy the Makefile in the *linux* directory to that directory. Modify the Makefile to include your the include or library directories. For example, the Makefile template in linux directory specifies the X11 library location as -L/usr/X11R6/lib. You may have to change it, if make complains that it can not find the X11 library. Once modified, execute "make copysim" which will compile the sim code and copy them to *$ATMROS/bin*. After verifying sim execution, you may execute "make clean" to remove the .o, .a, and .bak files.
- ***It is very important that*** after installation is complete, include the *$ATMROS/bin* in the $PATH environment variable and

12.2. *Installation Notes*

The Makefile in each src directory assumes the use of cc. If your system uses gcc, change the line "CC=cc" in the Makefile to "CC=gcc". It would be best to create a separate network directory to contain the results for a network design, similar to the arrangement under the *net* subdirectory here.

12.3. *ATMROS DIRECTORY ORGANIZATION*

The *$ATMROS* archive structure contains the following directories:

data/gnuplotoutput
    /scriptoutput

doc

src/nistsim
/perlscripts
/qostool
/whittle

traces

The "data" directories contain script program output either in the form of "plots" (gnuplotoutput), or files which aid in identifying bursty regions (scriptoutput). Each directory contains a readme file.

The "doc" directory contains the powerpoint file of the presentation at CASI sympossiuml, this final report in MS Word format, and files used to assemble the CASI final report, including figures.

The "src" directory contains source code. Simulator code is in *nistsim*. Perl scripts used to evaluate burstiness of the composite trace or of trace segments is in the *perlscript* directory. Perl scripts used to calculate trace QoS based on simulator log files are in *qostool*, along with some output files. Finally, *whittle* contains information on how to obtain the archieve of the source code for the Whittle estimator, which computes approximations of the Hurst burstiness constant, H.

The "traces" directory contains various LAN traces used in this work.