Simulation study of World Wide Web traffic over the ATM ABR service

Bobby Vandalore, Shivkumar Kalyanaraman, Raj Jain, Rohit Goyal, and Sonia Fahmy

Raj Jain is now at Washington University in Saint Louis, jain@cse.wustl.edu http://www.cse.wustl.edu/~jain/

ABSTRACT

Asynchronous transfer mode (ATM) is the technology chosen for implementing the Broadband Integrated Services Digital Network (B-ISDN). The performance of Internet protocols over ATM is an extremely important research area. As web traffic forms a major portion of the Internet traffic, we model world wide web (WWW) servers and clients running over an ATM network using the available bit rate (ABR) service. The WWW servers are modeled using a variant of the *SPECweb96*[1] benchmark, while the WWW clients are based on a model proposed in [2]. The traffic generated is typically bursty, having active and idle transmission periods. A timeout occurs after a certain idle interval. During idle periods, the underlying TCP congestion windows remain large until the timer expires. When the application becomes active again, these large windows may be used to send data in a burst. This raises the possibility of large queues at the switches, if the source rates are not controlled by ABR. We study this problem and show that ABR scales well to a large number of bursty TCP sources in the system.

Keywords: ATM, WWW model, TCP/IP, ABR service

1. INTRODUCTION

ATM is a high speed, connection-oriented, cell switching technology. It uses small fixed-size packets (also called cells) to transport its traffic. It is designed to provide an integrated service for supporting audio, video and data traffic. ATM provides multiple categories of service to support different quality of service (QoS) requirements. The current set of service categories specified are: the constant bit rate (CBR), real-time variable bit rate (rt-VBR), non-real time variable bit rate (nrt-VBR), available bit rate (ABR), and unspecified bit rate (UBR). The CBR service is aimed at transporting voice and synchronous data applications. The VBR (rt- and nrt-) services provide support for video and audio applications which do not require isochronous transfer. The ABR service transports data traffic. It guarantees a minimum cell rate (MCR) allocation and uses closed-loop feedback to control source rates. Cell loss is small in ABR, though ABR does not give bounded cell loss guarantees. UBR provides "best-effort" delivery for data applications and does not give any guarantees.

The ATM technology is already being used in the backbone of the Internet and many campus backbones. The transport control protocol/internet protocol (TCP/IP) suite of protocols are the most important protocols of the Internet. The performance of TCP/IP over ATM has been addressed in [3–5].

As large ATM networks are built, it is important to study the performance of real-world applications like the World Wide Web over ATM. Such applications typically have low average bandwidth demands from the network, but desire good response time when they become active. It is interesting from the traffic management perspective to study the aggregate effect of hundreds of such applications browsing or down-loading large documents over an ATM backbone. This study focuses on the performance of WWW traffic over the ATM ABR service.

Section 2 gives an overview of the TCP congestion control algorithm, the congestion control mechanism in the ATM ABR service, and the ERICA+ algorithm. Section 3 discusses the issues involved in transporting WWW traffic over ATM. Section 4 discusses the implications of using the HTTP/1.1 standard and presents the WWW server model and the WWW client model used in this study. Section 5 explains the simulation configurations, and section 6 gives the TCP/IP and the ERICA+ parameter values used in the simulations. Section 7 discusses the simulation results. Finally, section 8 provides a summary of the paper.

Proceedings of SPIE Symposium on Voice, Video and Data Communications, Vol. 3530, Conference on Performance and Control of Network Systems II, Boston, MA November 1998, pp. 415-422. ©SPIE.
Further author information:
Tel: (614) 688-4482; Fax: (614) 292-2911; E-mail: {vandalor, jain}@cis.ohio-state.edu; WWW: http://www.cis.ohio-state.edu/~jain/

2. CONGESTION CONTROL

In this section, we give a brief overview of TCP congestion control, the ABR flow control mechanism, and the ERICA+ rate allocation algorithm.

2.1. TCP Congestion Control

TCP provides a reliable, connection-oriented service. TCP connections provide window-based end-to-end flow control. The receiver's window (*rcvwnd*) is enforced by the receiver as a measure of its buffering capacity. The congestion window (*cwnd*) is used at the sender as a measure of the available capacity of the network. The sender cannot send more than the minimum of *rcvwnd* and *cwnd*. The TCP congestion control scheme[6] consists of the "slow start" and "congestion avoidance" phases. In the "slow start" phase, *cwnd* is initialized to one TCP segment. The *cwnd* is incremented by one segment for each acknowledgment received, so the *cwnd* doubles every round trip. The *cwnd* can reach up to *ssthresh* (initialized to 64K bytes).

In the "congestion avoidance" phase, the *cwnd* is incremented by $1/cwnd$ for every segment acknowledged. If an acknowledgment is not received by the timeout period, the segment is considered to be lost, and "slow-start" phase is entered. The *cwnd* is set to one, and *ssthresh* is set to *max(2, min(cwnd/2, rcvwnd))*. Figure 1 shows the different phases of the TCP congestion control mechanism. It has been shown that TCP/IP performance can be improved with the *fast retransmit and recovery*[7] and *selective acknowledgments*[8] options.

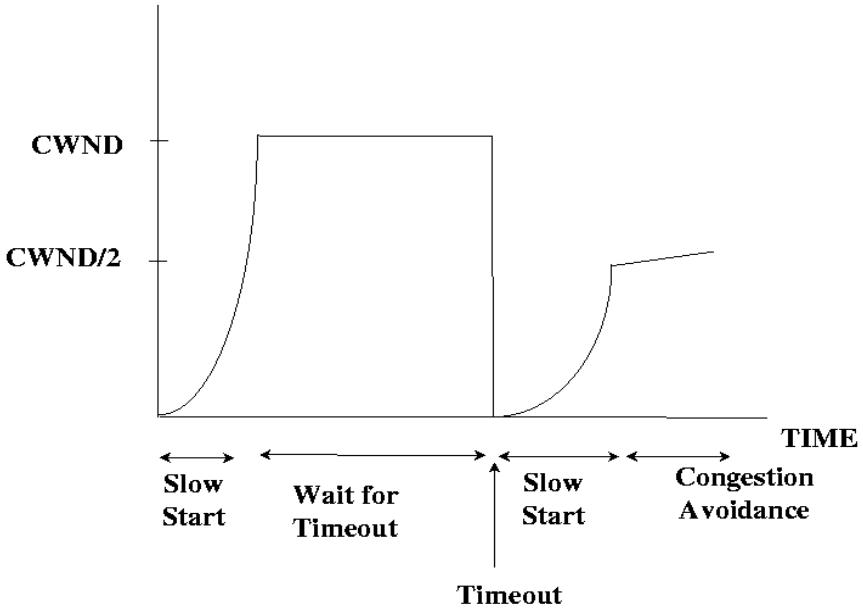

Figure 1. TCP CWND vs Time

2.2. ABR Flow Control

The ABR service uses closed-loop feedback control to advise the sources about the rate at which they should be transmitting data. The switches monitor their load, compute the available bandwidth and divide it fairly among the competing sources. The feedback from the switches to the sources is sent in Resource Management (RM) cells which are sent periodically (one RM cell after every $Nrm - 1$ data cells[9]) by the sources and turned around by the destinations (see figure 2).

The RM cells traveling in the forward direction are called forward RM cells (FRMs) while those returning from the destination to the source are called backward RM cells (BRMs). When a source receives a BRM cell, it computes its allowed cell rate (ACR).

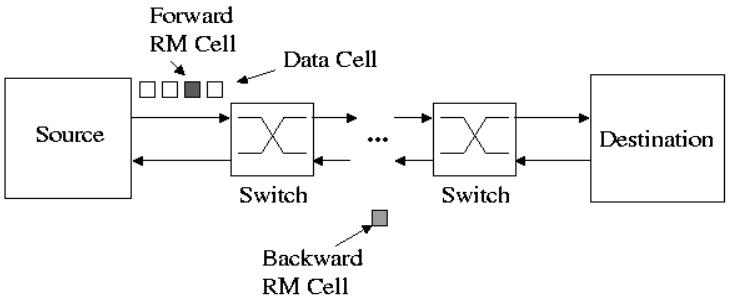

Figure 2. RM cell path

2.3. The ERICA+ Switch Algorithm

The ERICA+ switch algorithm operates at each output port of a switch. The switch periodically monitors the load on each link and determines a load factor, the available ABR capacity, and the number of currently active VCs. The measurement period is called the "averaging interval". The measurements are used to calculate the feedback rate which is indicated in the BRM cells. Measurements are performed in the forward direction, while feedback is given in the reverse direction. ERICA+ uses a queue control function to drain queues at the switch by varying the target rate dynamically. The complete description of the ERICA+ algorithm is given in [10].

3. WWW TRAFFIC OVER ATM

The WWW application sets up TCP connections for its data transfers.[11] The WWW application differs from a large file transfer application in that, while the latter looks like an *infinite* or *persistent* application to TCP, the former appears as a *bursty* application with active and idle transmission periods. The effect of this on traffic management is described below.

TCP increases its *congestion window* as it receives acknowledgments for segments correctly received by the destination. If the application (such as file transfer or WWW server/client) has data to send, it transmits the data. Otherwise, the window remains open until either the application has data to send or TCP times out. The timer set by TCP's round trip time (RTT) estimation algorithm. The timer can go off for two reasons: (i) retransmission timeout: if an acknowledgment is not received within the timeout period, (ii) idle timeout: if there is no activity at the source for a given amount of time. The timeout period is at least as much as the timer granularity, which is typically 100-500 milliseconds. If the timer goes off, TCP reduces the congestion window to one segment, and rises exponentially (*slow start* phase) once the source becomes active again.

On the other hand, if the application remains idle for a period smaller than the timeout, the window is still open when the source becomes active again. If acknowledgments (corresponding to the data sent) are received within this idle interval, the window size further increases. Since no new data is sent during the idle interval, the usable window size is large. The large window maybe used by the application to send a large burst of traffic.

When TCP transporting such a WWW application is transferred over ATM, the burst of data is simply transferred to the NIC (network interface card). Assuming that each TCP connection is carried over a separate ABR virtual circuit (VC), the data burst is sent into the ATM network at the VC's allowed cell rate (ACR). Since this VC has been idle for a period shorter than the TCP timeout (typically 500 ms for ATM LANs and WANs), it is an *ACR retaining* VC. Source End System (SES) Rule 5[9] specifies that the ACR of such a VC be reduced to ICR (initial cell rate) if the idle period is greater than parameter ADTF (ACR decrease time-out factor), which defaults to 500 ms. With this default value of ADTF, and the behavior of the TCP application, the ACR is not reduced to ICR. This situation can be potentially harmful to the switches if ACRs are high and sources simultaneously send data after their idle periods.

Observe that a persistent application using TCP over ABR does not send data in such sudden bursts. As discussed in our previous work,[12] the aggregate TCP load at most doubles every round trip time (since two packets are inserted

Table 1. Class, Files sizes and Frequency of Access

Class	File Sizes	Frequency of Access
Class 0	0 – 1KB	20%
Class 1	1KB – 10KB	28%
Class 2	10KB – 100KB	40%
Class 3	100KB – 1MB	11.2%
Class 4	1MB – 10MB	0.8%

into the network for every packet transmitted, in the worst case). Bursty TCP applications may cause the aggregate load to more than double in a round trip time.

In this paper, we show that such worst case scenarios are avoided in practice due to the nature of WWW applications and the ABR closed-loop feedback mechanism. In other words, ABR scales well to support a large number of bursty WWW sources running over TCP.

4. THE WWW SYSTEM MODEL

The world wide web uses the Hypertext Transfer Protocol (HTTP), which uses traditional TCP/IP for communication between the WWW clients and the WWW servers.[13] Modeling of the WWW traffic is a difficult task since the nature of traffic is changing due to the development of new HTTP standards, new WWW servers, WWW clients, and changes in user behavior. In this section, we outline our model and the inherent assumptions.

4.1. Implications of the HTTP/1.1 standard

The main difference between the latest version of the HyperText Transfer Protocol, HTTP/1.1,[11] and earlier versions is the use of persistent TCP connections as the default behavior for all HTTP connections. In other words, a new TCP connection is not set up for each HTTP request. The HTTP client and the HTTP server assume that the TCP connection is persistent until a *Close* request is sent in the HTTP Connection header field.

Another important difference between HTTP/1.1 and earlier versions is that the HTTP client can make multiple requests without waiting for responses from the server (called *pipelining*). The earlier models were *closed-loop* in the sense that each request needed a response before the next request could be sent.

4.2. WWW Server Model

We model our WWW servers as infinite servers getting file requests from WWW clients. The model is an extension of that specified in the SPECweb96 benchmark.[1] The file requests fall into five classes (Class 0 through Class 4) which are shown in table 1.

There are nine discrete sizes in each class (e.g., 1 KB, 2 KB, up to 9 KB, then 10 KB, 20 KB, through 90 KB, etc). We assume that the accesses within a class are evenly distributed. The model of discrete sizes in each class is based on the SPECweb96 benchmark.[1] The key differences from the SPEC model are: (i) the assumptions of an infinite server, and (ii) a new distribution of file sizes, which allows us to model file sizes larger than those in the SPEC benchmark.

Specifically, the average file size in our distribution is approximately 120 KB, compared to an average file size of about 15 KB in SPECweb96. Our distribution introduces an extra class of file sizes (1 MB - 10 MB) which models the down-loading of large software distributions, and off-line browsing search results. We justify the weight assignments for the various classes in the next subsection.

4.3. WWW Client model

The HTTP model proposed in[2] describes an empirical model of WWW clients based on observations in a LAN environment. Specifically, a typical client is observed to make, on the average, four HTTP GET requests for a single document. Multiple requests are needed to fetch in-line images, if any. With the introduction of JAVA scripts in web pages, additional accesses maybe required to fetch the scripts. Therefore, we use five as the average number of HTTP GET requests. The caching effects at the clients are ignored.

HTTP/1.1 uses persistent TCP connections and sends multiple requests without waiting for the results of the earlier requests (*open-loop mode*). But, the *open-loop* mode is not used for the first request because the additional requests have to be made based on the results of the first response.

Typically, the first HTTP request from an HTTP client accesses the index page (plain text), which is of size 1 KB or less. Since every fifth request is expected to be an index page access, we assign a weight of 20% (= 1/5) for the file size range of 1 KB or less.

Additional requests may require larger responses, as modeled by our distribution of file sizes at servers, taking into consideration the possibility of larger file sizes in the future.

We also model a time lag between batches of requests (presumably for the same document), which corresponds to the time taken by the user to request a new document, as a constant, 10 seconds. While this may be too short a time for a human user to make decisions, it also factors in the possibility of off-line browsing where the inter-batch time is much shorter.

We do not attempt to model user behavior across different servers. The main purpose of using this simplistic model is to approximate the small loads offered by individual web connections, and to study the effects of aggregation of such small loads on the network.

5. THE *K-N CLIENT-SERVER* CONFIGURATION

The *K-N Client-Server* configuration shown in figure 3 has a single bottleneck link (LINK1) shared by the N Web servers and K clients per server or a total of N × K clients). Each client (e.g., a netscape browser) sets up a TCP connection which goes through an ATM VC on a separate NIC. The VC goes through the switch (SW1) to the NIC corresponding to its server. Every server has a NIC card connected to it. Therefore there are N NICs, one for each of the N servers. The ATM VCs from the clients reach the server through the server NIC and a separate TCP entity. Hence, there are K TCP entities corresponding to the K clients per server. In our simulations, K = 15, and N = 1, 2, 5, 10. All link lengths are 1000 kms. The bottleneck link (LINK1) speed is 45.0 Mbps, modeling a T3 link, while the remaining link speeds are 155.52 Mbps.

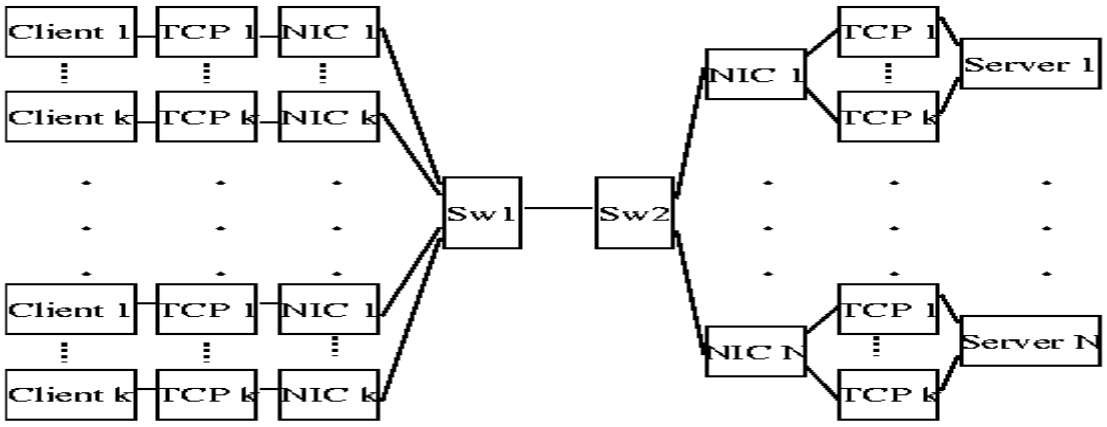

Figure 3. K-N Client-Server configuration

We define feedback delay as sum of the time required for feedback from the bottleneck switch to reach the source and the delay for the new load to be sensed at the switch. In our configuration, the feedback delay is 10 ms, which corresponds to 3680 cells at 155.52 Mbps. The round trip time is 30 ms, which corresponds to 11040 cells.

6. TCP AND ERICA+ PARAMETERS

We use a TCP maximum segment size (MSS) of 512 bytes. The window scaling option is used to obtain larger window sizes for our simulations. For our WAN simulations, we use a window of 16 × 64 KB or 1024 KB, which is greater than the product of the round trip time (RTT) and the bandwidth, yielding 454,875 bytes at 121.3 Mbps TCP payload rate (defined below) when the RTT is 30 ms.

TCP data is encapsulated over ATM as follows. First, a set of headers and trailers are added to every TCP segment. We have 20 bytes of TCP header, 20 bytes of IP header, 8 bytes for the RFC1577 LLC/SNAP encapsulation, and 8 bytes of AAL5 (ATM adaptation layer) information, a total of 56 bytes. Hence, every MSS of 512 bytes becomes 568 bytes of payload for transmission over ATM. This payload with padding requires 12 ATM cells of 48 data bytes each. The maximum throughput of TCP over raw ATM is (512 bytes/(12 cells × 53 bytes/cell)) = 80.5%. Further, in ABR, we send FRM cells once every $Nrm - 1$ (31) cells. Hence, the maximum throughput is 31/32 × 0.805 = 78% of ABR capacity. For example, when the ABR capacity is 45 Mbps, the maximum TCP payload rate is 35.1 Mbps. Note that higher efficiency can be achieved by using larger MSS values.

We are interested in comparative efficiency, so we normalize our efficiency measurements. We define *efficiency* to be the ratio of the TCP throughput achieved to the maximum throughput possible. As defined above, the maximum throughput possible is 0.78 × the mean ABR capacity.

In our simulations, we have not used the *fast retransmit and recovery* and *selective acknowledgment* options of TCP/IP. Since there is no loss, these mechanisms are not exercised.

The ERICA+ algorithm[10] uses five parameters. The algorithm measures the load and number of active sources over successive averaging intervals and tries to achieve 100% utilization with queuing delay equal to a target value. The averaging intervals end either after a specified period of time, or after a specified number of cells has been received, whichever happens first. In our simulations, the values used are 500 ABR input cells or 5 ms. The other parameters are used to define a function which scales the ABR capacity in order to achieve the desired queuing delay goals. These include a target queuing delay, T0, set to 500 microseconds, two curve parameters (a = 1.15 and b = 1.05), and a factor which limits the amount of ABR capacity allocated to drain the queues (queue drain limit factor, QDLF = 0.5).

7. SIMULATION RESULTS

In our simulations, we use 15 clients per server (K=15). The number of servers in the system is varied from one to fifteen. Given the average file size of 120 KB, five requests per batch, and a constant inter-batch time of 10 seconds, the average load generated per client is 0.48 Mbps. With N servers (and K=15 clients per server), the expected load on the system is 7.2 × N Mbps. As we vary N from 1 to 15, the expected load varies from 7.2 Mbps to 72 Mbps over a bottleneck link of speed 45 Mbps.

The simulation results are presented in table 2.

Observe that the maximum switch queue (which corresponds to the buffer requirement at the bottleneck switch) initially *increases linearly* as a function of input load (i.e., number of servers; see rows 1 through 5). The efficiency also increases linearly as a function of load in this range. The efficiency value is high (greater than 85%), and the variation in efficiency is due to use of random numbers (used for generating client requests) in the simulation.

However, as the average load on the network exceeds the bottleneck link capacity (N = 7,...,15), the buffer requirements do not increase correspondingly. In other words, the *maximum queue stabilizes* under overload conditions due to ABR feedback while the efficiency remains high. This is explained as follows.

Initially, when the network is lightly loaded, switches allocate high rates to sources. Due to the bursty nature of the WWW applications, and the use of persistent TCP connections, the sources may send bursts of cells into the network (seen as queues at the bottleneck switch). This leads to a *linear increase* in maximum queue lengths for low average loads. The maximum queue length is bounded by 3 × RTT delay cells.

Table 2. Effect Number of Servers on Efficiency and Maximum Queue length

Sources	ABR Metrics		
Number of servers	Max Switch Q (cells)	Total TCP Throughput	Efficiency (% of Max throughput)
1	3677 (0.33×RTT Delay)	6.11 Mbps	17.4%
2	6154 (0.56×RTT Delay)	14.16 Mbps	40.3%
3	7533 (0.68×RTT Delay)	21.11 Mbps	60.1%
4	10217 (0.93×RTT Delay)	28.46 Mbps	81.1%
5	14057 (1.27×RTT Delay)	34.08 Mbps	97.1%
6	16342 (1.48×RTT Delay)	34.28 Mbps	97.6%
7	13846 (1.25×RTT Delay)	34.05 Mbps	97.0%
8	19399 (1.76×RTT Delay)	34.05 Mbps	97.0%
9	23471 (2.13×RTT Delay)	33.04 Mbps	94.1%
10	17269 (1.56×RTT Delay)	32.72 Mbps	93.2%
11	17134 (1.55×RTT Delay)	33.02 Mbps	94.1%
12	17956 (1.63×RTT Delay)	32.60 Mbps	92.9%
13	27011 (2.45×RTT Delay)	32.45 Mbps	92.4%
14	17652 (1.60×RTT Delay)	30.03 Mbps	85.6%
15	16268 (1.47×RTT Delay)	30.10 Mbps	85.8%

However, as the average load on the bottleneck switch increases, the switch allocates lower rates to the contending sources. Now, even if the WWW applications send bursts of cells, they are not admitted into the ATM network since the source rate allocations are low. The low rate allocations limit the sudden load seen by the network under overload conditions with bursty TCP sources.

8. SUMMARY

In this paper, we have investigated the performance of ABR when bursty applications like WWW clients and servers using TCP run over ATM networks. The problem with such applications is that they might utilize the large TCP congestion windows to send bursts of data on the underlying ATM network, resulting in large queues. Though the burst sizes (captured by file size distributions) used by these applications may be large, the average rate per connection is small. Hence, it takes a large number of connections to load the network. However, as the load on the network increases, the ABR switch algorithm controls the source rates to low values, and restricts the burstiness in load seen by the network. As a result, the maximum queue lengths in the network are bounded while the efficiency remains high. Hence, ABR responds to load increases and scales well to realistic workloads.

ACKNOWLEDGMENTS

This research was sponsored in part by NSF Grant/Contract #NCR-9628438.

REFERENCES

1. SPEC benchmark organization, *An Explanation of the SPECweb96 Benchmark*, http://www.specbench.org/osg/web96/webpaper.html.
2. B. Mah, "An Empirical Model of HTTP Network Traffic," in *Proc. of IEEE INFOCOM'97*, April 1997.
3. H. Li, K. Y. Siu, H. T. Tzeng, C. Ikeda, and H. Suzuki, "TCP over ABR and UBR Services in ATM," in *Proc. IPCCC'96*, March 1996.
4. A. Romanov and S. Floyd, "Dynamics of TCP traffic over ATM Networks," *IEEE JSAC* , 1995.
5. R. Goyal, R. Jain, S. Kalyanaraman, S. Fahmy, and B. Vandalore, "Improving the Performance of TCP over the ATM-UBR service," *To appear in Computer Communications* , 1998.

6. V. Jacobson, "Congestion Avoidance and Control," in *Proc. of the SIGCOMM'88*, pp. 314–32, August 1988.
7. W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," *Internet RFC 2001* , January 1997.
8. M. Mathis, J. Madhavi, S. Floyd, and A. Romanow, "Tcp selective acknowledgment options," *Internet RFC 2018* , 1997.
9. S. S. Sathaye, *ATM Forum Traffic Management Specification Version 4.0*, ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0056.0000.pdf, 1996.
10. S. Kalyanaraman, R. Jain, R. Goyal, S. Fahmy, and B. Vandalore, "The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks," *Submitted to IEEE/ACM Transactions on Networking* , November 1997.
11. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," *Internet RFC 2068* , 1997.
12. S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, and S. Kim, "Performance and Buffering Requirements of Internet Protocols over ATM ABR and UBR Services," in *IEEE Communications Magazine*, June 1998.
13. T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext Transfer Protocol – HTTP/1.0," *Internet RFC 1945* , May 1996.