

The Performance of TCP Over ATM ABR and UBR Services

[Xiangrong CAI {cai@cis.ohio-state.edu}](mailto:cai@cis.ohio-state.edu)

This is a survey paper on the performance of TCP over ATM ABR and UBR services. Using throughput and fairness as our performance metrics, the effect of the following factors to the performance of TCP over ATM ABR and UBR is discussed here: (1)the switch dropping policies when overflow happens(simple discard, Tail Drop, Drop From Front, Early Packet Drop, Per-VC queuing, Fair Buffer Allocation); (2)the TCP end system policies, such as FRR and SACK; (3)buffer requirements in order to maintain no cell loss or high throughput; and (4)Guarantee Rate. We also make a comparison between the performance of TCP over ABR and TCP over UBR.

[Other Reports on Recent Advances in Networking](#)

[back to Raj Jain's Home Page](#)

Contents

- [Introduction](#)
 - [TCP Features](#)
 - [Slow Start and Congestion Avoidance](#)
 - [Fast Retransmit and Recovery](#)
 - [Selective Acknowledgments](#)
 - [ATM Features](#)
 - [ABR](#)
 - [UBR](#)
 - [TCP Over ATM](#)
 - [Performance Metrics](#)
 - [TCP over UBR](#)
 - [Switch Dropping Policies](#)
 - [End System Policies](#)
 - [Buffer Requirement](#)
 - [Guaranteed Rate](#)
 - [TCP Over ABR](#)
 - [Buffer Requirement](#)
 - [ABR With Varying Bandwidth](#)
 - [Comparison Between TCP Over ABR and UBR](#)
 - [Summary](#)
 - [Appendix of Acronyms](#)
 - [References](#)
-

Introduction

With increasing requirement for more bandwidth and high speed networks, the Asynchronous Transfer Mode(ATM) has become the main consideration of many academic and industrial entities. Not only many research work has been done on this by lots of academic institutions, but also many kinds of ATM products are produced and supplied by lots of network vendors. Among various kinds of services provided by ATM, the Available Bit Rate(ABR) and Unspecified Bit Rate(UBR) are used for data transmission [\[ATMF 4.0\]](#).

As a transfer mode, ATM develops very fast on both hardware and lower layer software(such as switch scheduling software and end system policy software). However, it lacks higher layer application software. As a result, lots of work about ATM is done theoretically. In order to see the real effect of ATM services, higher layer application software is needed to run over ATM. However, developing new higher layer ATM application software is a very time consuming task. As a solution, a lot of work is done to run existing software over ATM. This is normally called LAN Emulation(LANE).

Currently, the internet is developing at an extremely fast speed. Among the factors that make this happen, the protocol suite the internet uses -- TCP/IP, which stands for Transmission Control Protocol and Internet Protocol -- is obviously the most important one. In fact, as a statistic shows, more than 85% of networks are running TCP/IP. A natural idea is: Can TCP/IP run over ATM? The answer is yes. In this paper, we will talk about how to run TCP over ATM and the performance of TCP running over ATM(in another paper of this series, [IP over ATM](#) is introduced).

The paper is organized as follow: first, we introduce some features of TCP and ATM. More specifically, the slow start and congestion avoidance, fast retransmit and recovery, and selective acknowledgments of TCP and the ABR and UBR services of ATM are introduced. Then, we give our performance metrics, which are throughput and fairness. Using these metrics, we present the results of the performance of TCP over ATM, which include the following factors for both ABR and UBR: the switch dropping policies, the end system policies and the buffer requirements of TCP over ATM. We also compare the performance of TCP over ABR and TCP over UBR.



[Return to Contents](#)

TCP Features

The TCP/IP protocol suite consists of a lot of protocols for different purposes. For the TCP over ATM purpose, because the most important thing is to do congestion control, we only need to check the features that connected to the TCP end system flow control mechanisms. This includes at least these three parts: the slow start and congestion avoidance, Fast Retransmit and Recovery(FRR), and Selective ACKnowledgments(SACK). We will see how these end system mechanisms affect the performance of TCP over ATM.

Slow Start and Congestion Avoidance[\[Jacobson's congavoid\]](#)

TCP uses a window based protocol for flow control. The sender maintains a variable congestion window(CWND) to control how many segments can be put into the network at a specific time, while the receiver maintains a receiver window(RCVWND) to tell the sender how many segments it can afford. Initially, CWND is set to one segment and increased by one segment whenever it receives a new acknowledgment from the receiver until it reaches the minimum of RCVWND and maximum CWND(i.e: $\min\{\text{RCVWND}, \text{max}(\text{CWND})\}$, which is normally 65535 bytes). This is called the slow start stage of TCP(in fact, this is a misnomer, since the congestion window increases exponentially during this stage). It can be shown that during the slow start stage, the CWND doubles every Round Trip Time(RTT).

When a segment is lost, the receiver will send duplicate ACKnowledgments(ACK) on receiving subsequent segments.

The sender maintains a retransmission timer for the last unacknowledged packet. If the timer times out, it is assumed that the network is suffering congestion. Using this assumption, some further actions should be done to lower the load of the network. When this happens, the sender saves half of the CWND in a variable called SSTHRESH and resets the CWND to 1. The sender then retransmits segments starting from the lost one. CWND is increased by one on the receipt of each new ACK until it reaches SSTHRESH (slow start again). After that, CWND increases by one segment every round trip time. This results in a linear increase of CWND. This is called the congestion avoidance stage. Figure 1 shows the slow start and congestion avoidance stages for a typical TCP connection.

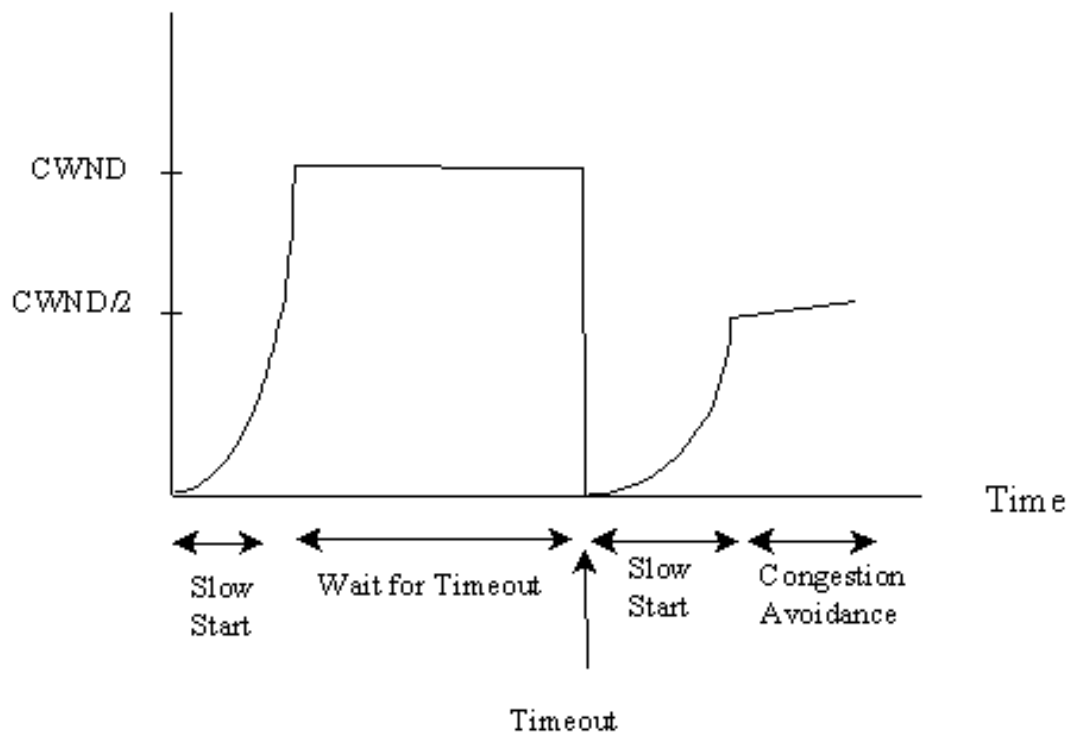


Figure 1: TCP Slow Start and Congestion Avoidance

Fast retransmit and recovery [\[RFC 1323\]](#)

TCP uses a coarse granularity (typically 500ms) timer for the retransmission timeout. As a result, the TCP connection can lose a lot of time waiting for the timeout when it suffers segment loss. During this waiting period, the TCP neither sends new packets nor retransmits the lost packets. Moreover, once the timeout happens, the CWND is set to 1 segment, which means the connection will take another several round trip time to make full use of the network link. This will definitely result in low efficiency. As an improvement, fast retransmit and recovery (FRR) was proposed to enable the connection recover from isolated segment loss quickly.

When the receiver receives an out-of-order segment, it sends a duplicate ACK to the sender immediately. When the sender receives three duplicate ACKs, it assumes that the segment indicated by the ACKs is lost. Therefore, new actions are taken. These actions are: the sender retransmits the lost segment immediately; the sender reduces the CWND to half (plus 3 segments, which correspond to the 3 duplicate ACKs), and saves the original CWND value in SSTHRESH. Now, for each subsequent duplicate ACK, the sender increases the CWND by one and tries to send a new segment. The effect of these actions is that the sender maintains the connection pipe at half of its capacity when it is in fast retransmit.

Approximately one round trip time after the missing segment is retransmitted, its ACK is received if the network recovered properly. At this time, instead of setting the CWND to one segment and doing the slow start again until

CWND reaches SSTHRESH, the TCP sets CWND to SSTHRESH directly, and then does the congestion avoidance. This is called fast recovery. Figure shows a case with 2 fast retransmissions.

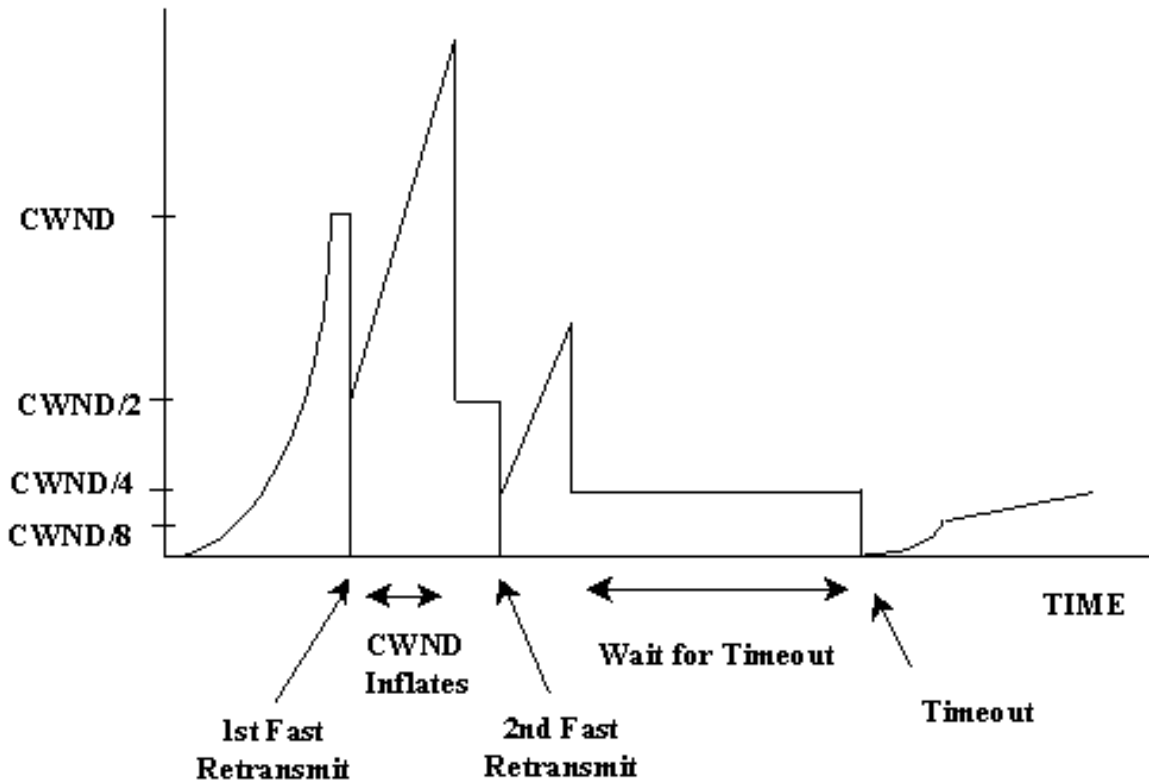


Figure 2: TCP Fast Retransmit and Recovery

Selective Acknowledgments [\[RFC 2018\]](#)

FRR works well for just isolated losses. If several losses happen in a short period of time, it performs badly. A new proposal, the selective acknowledgment(SACK) works well even under multiple packet losses. A SACK TCP acknowledgment contains additional information about the segments have been received by the destination. When duplicate SACKs are received from the destination, the sending TCP can reconstruct information about the segments not received at the destination. When the sender receives three duplicate ACKs, it retransmits the first lost segment, and increases the CWND by one for each duplicate ACK it receives. After that, whenever it is allowed to send another segment, it uses the SACK information to retransmit lost segments before sending any new segments. As a result, the sender can recover from multiple dropped segments in about one round trip time. Figure 3 shows a case of selective acknowledgment.

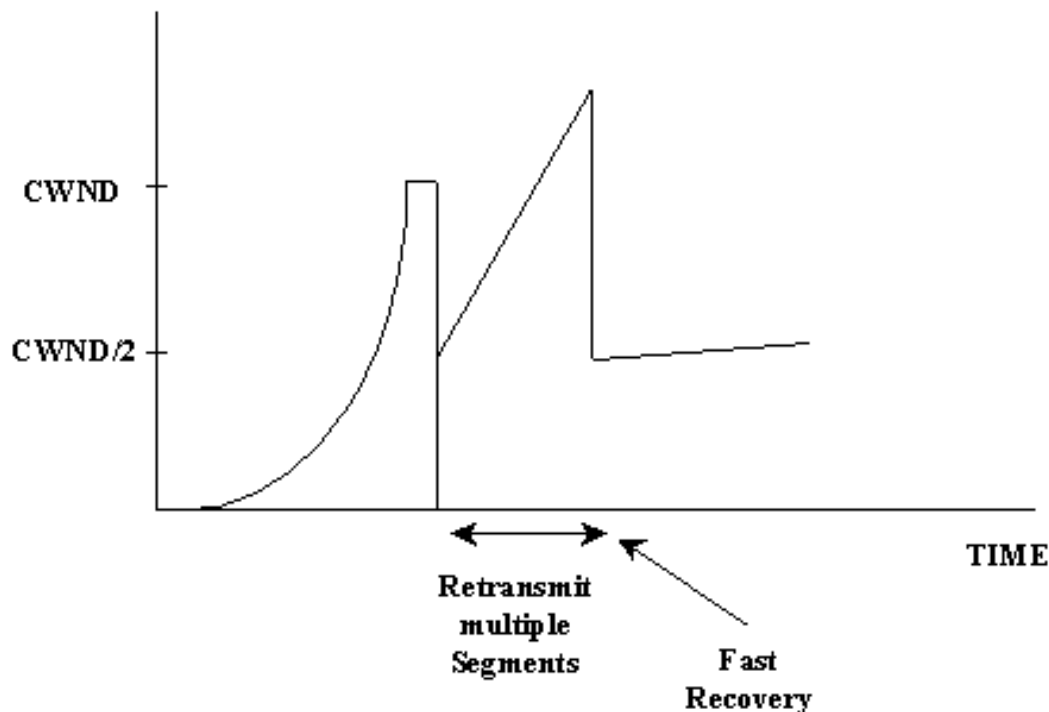


Figure 3: TCP Selective acknowledgment

So far we have some knowledge about TCP. This is not the whole of TCP. For more protocols of TCP, see RFC-793(Transmission Control Protocol), RFC-813(Window and Acknowledgment strategy in TCP), RFC-1122(Requirement for Internet Hosts), and RFC-1323(TCP extension for High Performance). However, the features of TCP mentioned in the above section are the main features that affect the flow control most. In the following section, we will see their effect to the performance of TCP over ATM.



[Return to Contents](#)

ATM Features

Compared to many traditional transfer modes, which provide "best effect" service(i.e. service supplier gives all the bandwidth it has to a SINGLE application), ATM supports multiple Quality of Services(QoS), which include Constant Bit Rate(CBR), Variable Bit Rate(VBR), Available Bit Rate(ABR) and Unspecified Bit Rate(UBR). These services share a common link according to preassigned rules. Therefore, not all of them can get the bandwidth they require. Since most data applications cannot predict their own bandwidth requirements, they usually need a service that dynamically shares the available bandwidth among all active users. ATM QoS and congestion control are used for this purpose. In ATM networks, the ABR service and UBR service are used to support non-delay sensitive data applications, and they are two kinds of "best effect" service [\[ATMF 4.0\]](#).

ABR

As the name shows, ABR normally uses the available bandwidth. This is often the left-over of the higher priorities services, which are CBR and VBR. Though the current standards for ABR service do not require the cell transfer delay and cell loss ratio to be guaranteed, it is desirable for switches to minimize the delay and loss as much as possible. The

ABR service requires network switches to constantly monitor their load and feed this information back to the sources, which in turn dynamically adjust their input into the network. This is mainly done by inserting Resource Management(RM) cells into the traffic periodically and getting the network congestion state feedback from the returned RM cells, which may contain congestion information reported by the switches and destination. Depending upon the feedback, the source is required to adjust its transmission rate. Figure 4 shows an ABR traffic Management model. The RM cell contains an Explicit Rate(ER) field. The switches along the path put some information to indicate the rate that the source should use after the receipt of the RM cell.

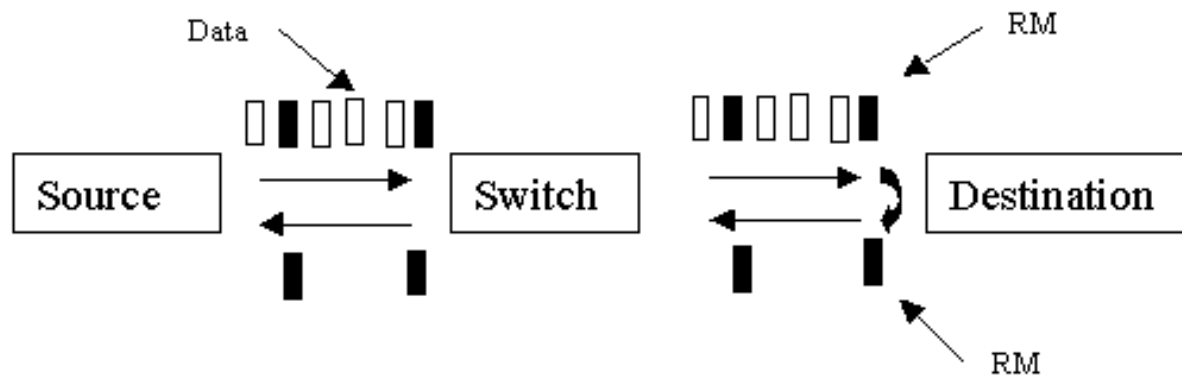


Figure 4: ABR Traffic Management Model

ABR users are allowed to declare a Minimum Cell Rate(MCR), which is guaranteed to the virtual connection(VC) by the network. Most VCs use zero as the default MCR value. However, for an ABR with higher MCR, the connection may be denied if sufficient bandwidth is not available.

It should be mentioned that both ABR data traffic and the available bandwidth for ABR are variable. Even though the average bandwidth is enough for an ABR connection in the long run, if there is not enough buffer to buffer the bursty traffic(either from VBR, which reduces the available bandwidth, or ABR itself, which requires more bandwidth), too many losses will result in low performance. For TCP over ABR, we will discuss the buffer requirements in order to maintain low cell lose rate and high performance.

UBR

UBR service is designed for those data applications that want to use any left-over capacity and are not sensitive to cell loss or delay. Such connections are not rejected on the basis of bandwidth shortage and not policed for their usage behavior. The switches only do very little thing to monitor the queues and simply discard cells or packets if the connection is overloaded. Several cell dropping policies are used for UBR. For TCP over UBR, these cell discarding policies are very important factors that influence the performance.



[Return to Contents](#)

TCP Over ATM

Now it is time for us to give the results of TCP over ATM. In this part, we mainly focus on the *performance* of TCP over ATM. This can be further divided into two main groups for both ABR and UBR: switching policies and end system policies. But before we give the results, let's first define what our performance is.

Performance Metrics

In our terms, **the performance of TCP over ATM is measured by two factors: the efficiency and the fairness.**

Efficiency is defined as follows:

$$\text{Efficiency} = (\text{Sum of TCP throughputs}) / (\text{Maximum possible TCP throughput})$$

The TCP throughputs are measured at the destination TCP layers and is defined as the total number of bytes delivered to the destination application, divided by the total simulation time.

Because of transmission overhead, the throughput is normally less than the link bandwidth. Here is an example to calculate the valid throughput for link bandwidth 155.52 Mbps and segment size 512 bytes:

For 512 bytes of data, the ATM layer receives:

512 bytes of data
 + 20 bytes of TCP header
 + 20 bytes of IP header
 + 8 bytes of LLC header
 + 8 bytes of AAL5 trailer
 = 568 bytes

Since $11 \times 48 < 568 < 12 \times 48$, these are padded to produce 12 ATM cells. Thus each TCP segment results in $12 \times 53 = 636$ bytes at the ATM layer. For this configuration, the maximum possible throughput = $512 / 636 = 80.5\% \Rightarrow 125.2$ Mbps.

Fairness is measured by the fairness index, which is defined as:

$$\text{Fairness Index} = \frac{\text{sqr}(\text{sum of } X_i)}{(N * \text{sum of } \text{sqr}(X_i))}$$

where X_i = throughput of the i th TCP source, and N is the number of TCP sources.

The reasonability of our definition is obvious. For a good service supplier who supplies different kinds of QoS to many users in the same time, the efficiency and the fairness are the same important. It is a misconception that throughput or efficiency is the only measure of performance. With multiple users sharing the same link, it is obvious that we should consider the fairness among the contending users.



[Return to Contents](#)

TCP over UBR

We will discuss the performance of TCP over UBR based on the following four factors: Switch dropping policies, end system policies, buffer requirements and guaranteed rate.

Switch dropping policies

As mentioned above in the UBR section, UBR switches almost do nothing for congestion control. They only drop cells whenever there is overflow. **Initially large buffers without congestion control plus TCP was proposed for TCP over UBR.** It was argued that whenever the buffer is big enough, there will be no loss. However, this is incorrect both theoretically and practically. No matter how big the buffer is, with the increase of traffic, it almost always overflows. Also, large buffers means longer delays. Though UBR services do not guarantee any kind of delay, practically one cannot wait for something forever.

To solve this problem, many dropping policies are proposed. **The earliest and easiest drop policy is called Tail**

Drop(TD)[\[Fang's simulation\]](#). It just drop any incoming cells whenever the switch buffer is full, leaving the higher layer such as TCP to do the recovery. This is also called TCP over plain ATM. However, simulation results show that it has very poor performance. The reason is that when ATM switch drops a cell, the rest of the higher-level packet is still transmitted, clogging the congested link with useless data. Thus some additional control mechanisms are required to achieve acceptable performance for TCP in ATM networks.

There is also an interesting phenomenon due to tail drop reported by [\[Kalyanaraman's TCP over ABR\]](#):

In AAL5, the source marks the last cell of each message by End-of-Message(EOM) bit. If the EOM cell is dropped at the switch, the retransmitted packet gets merged with previous partial packet at the destination. The merged packet fails the CRC test and is dropped at the destination by AAL5. The source will have to retransmit two packets.

After the first retransmission, the Ssthresh is set to half the previous window size and the window is set to one(see the above slow start and congestion avoidance of TCP). When the second retransmission occurs, the window is one and hence Ssthresh is set to 2(the minimum value). The window remains at one. TCP hence increase the window linearly resulting in low throughput for this resource. Since the EOM cells of the other TCP sources may not have been dropped, they do not experience this phenomenon and get high throughput.

Opposite to drop from tail, **drop from front was proposed by [\[Lakshman's Drop from front\]](#) to make the use of TCP fast retransmit mechanism**. The idea of drop from front is when a cell arrives to a full buffer, it is allowed in, with space being created by discarding the cell at the front of the buffer. This is reasonable because in all versions of TCP, the window is at least halved upon a detected loss. Halving the window causes the source to stop sending packets for about half a typical RTT. With drop from front, some sources receive earlier congestion notification and thus the FRR can react earlier. The rate reduction of these sources allows other sources to successfully enter and also leave the buffer. The shortened congestion episode with fewer sources losing packets greatly reduces or eliminates later over reaction by more sources. As a result, throughput is improved.

[\[Lakshman's Drop from front\]](#) gives some simulation results which compare the pure cell drop at the tail(Cell-Tail), pure cell drop at the front(Cell-Front), partial cell drop at the tail(Frame-Tail) and partial cell drop at the front(Frame-Front). It concludes that Frame-Front is better than Cell-front and Frame-Tail. Frame-Tail is better than Cell-Front for small buffer sizes but Cell_Front becomes better than Frame-Tail for larger buffer sizes.

The disparity in throughput results in unfairness among sources. As two improvements, **Partial Packet Discard(PPD) and Early Packet Discard(EPD) are proposed in [\[Romanow's TCP over ATM\]](#)**. In PPD, if a cell is dropped from a switch buffer, the subsequent cells in the higher layer protocol data unit are discarded. In EPD, when the switch buffer queues reach a threshold level, entire higher level data units are dropped. In [\[Kawahara's selective cell discard\]](#), EPD is further divided into EPD1 and EPD2. In EPD2, the node sets up variable thresholds depending on the number of accepted packets. While in EPD1, the threshold is fixed according to a calculation or assumption. A better performance is obtained by using EPD2 than using EPD1.

In EPD, an important decision to be made is the choice of the threshold for discarding packets. With a threshold close to the buffer size, the buffer can be used more efficiently, but it also means more drops and retransmission. On the other hand, if the threshold is much less than the buffer size, very few drops can happen, but the buffer utilization rate is low. There are two general approaches in setting the buffer threshold. **The first one is to set the threshold as a percentage of the total buffer size**. For example, in [\[Kalampoukas's comparison\]](#), the threshold is set to $\max\{0.8, [\text{buffer size} - 3 * \text{segment size}]/[\text{buffer size}]\}$, which is a percent value. This approach can be very conservative sometimes. Given 80% as the percentage, it may be proper for a buffer size of 50k, which will reserve 10k. But for a buffer size of 500k, 100k will be wasted. This is far more than what is necessary for the ATM-EPD scheme in most case. **The alternative approach is to set the threshold to a value that will always reserve a certain amount of space**. This can be further divided into two ways: Fix-sized and Variable-sized according to the configuration, such as number of sources. The above mentioned EPD1 and EPD2 are examples of these two divisions. [\[Goyal's UBR+\]](#) gave $[\text{buffer size}] - [\text{Maximum segment size(MSS)}] * [\text{number of active sources}]$ as the threshold. It is assumed the worst case is that every connection except one has its first cell in the buffer when the threshold is reached. Thus, the buffer

should reserve at least $\{MSS * [\text{number of active sources}]\}$ space for the coming packet of every suspending connection. However, in [\[Kalampoukas' comparison\]](#), simulation results show that reserving $\{3 * \text{pkts} * [\text{number of sources}]\}$ space gives the best result.

Studies of EPD show that it does improve the throughput, but it does not attempt to improve fairness. This is explainable. Based on EPD, it is highly possible that some of the VCs happen to finish transferring all the cells of one packet and get a chance to further increase its CWND. In the meantime, some unlucky VCs suffer cell loss and are required to reduce its CWND. As a result, the lucky ones get more chance to transmit, while the unlucky ones are further restricted. In the long run, the fairness is badly maintained.

A further improvement to packet dropping policy is **Selective Packet Dropping (SPD)**. It is also called **packet dropping based upon per-VC accounting**. The basic idea of SPD is that it keeps track of the activity of each VC by counting the number of cells from each VC in the buffer. A VC is said to be active if it has at least one cell in the buffer. A fair allocation is calculated as the current buffer occupancy divided by number of active VCs.

Let the buffer occupancy be denoted by X , and the number of active VCs be denoted by N_a . Then Fair allocation = X/N_a . The ratio of the number of cells of a VC in the buffer to the fair allocation gives a measure of how much the VC is overloading the buffer, i.e., by what ratio it exceeds the fair allocation. Let Y_i be the number of cells from VC_i in the buffer. Then the Load Ratio of VC_i is defined as:

$$\text{Load Ratio of } VC_i = (\text{number of Cells from } VC_i) / (\text{Fair allocation}) = Y_i * N_z / X$$

If the load ratio of a VC is greater than a parameter A , then new packets from that VC are dropped in preference to packet of a VC with load ratio less than Z . Figure 5 shows the buffer management of the SPD. K stands for buffer size in cells, R stands for the minimum threshold, and X stands buffer occupancy.

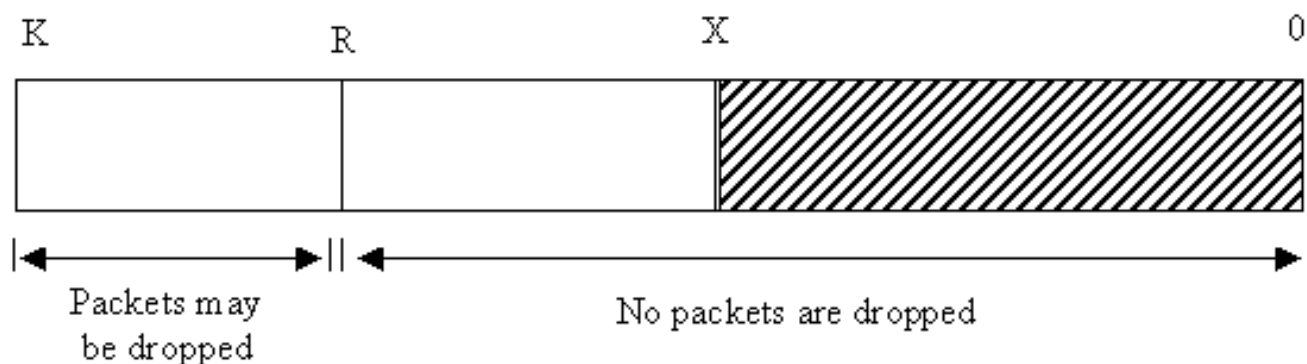


Figure 5: SPD and FBA: Buffer Occupancy for Drop

[\[Li's per-VC queuing\]](#) gives a pseudo code for per-VC queuing which makes this easy to understand. In the following code, Q_j is the number of cells from VC_j in the switch buffer, Q is the number of cells in the switch buffer, Th is the threshold, and K is a control parameter with a typical value between 1 and 2.

When a cell in VC_j comes to an ATM switch:

if the cell is the first cell of a packet

*calculate $Th = K * Q/N$*

if $Q \geq Th$ and $Q_j \geq Th$

discard the cell

else

```

        accept the cell into the single FIFO queue
         $Q_j = Q_j + 1$ 
    else
        if any cell of the packet has been discarded
            discard the cell
        else
            if  $Q \geq Q_{max}$ 
                discard the cell
            else
                accept the cell into the single FIFO queue
                 $Q_j = Q_j + 1$ 

```

Simulation results show that SPD using per-VC accounting improves the fairness of TCP over UBR + EPD, fairness and efficiency increase with increase in buffer size, and fairness decrease with increasing number of sources. See [\[Goyal's UBR+\]](#) and [\[Li's per-VC queuing\]](#).

The final improvement to switch dropping policy is called **Fair Buffer Allocation(FBA)** [\[Heinanen's FBA\]](#). It is also called **per-VC queuing**. It uses a smooth form of the parameter of Z and compares it with the load ratio of VC. For the same set of parameters as SPD, a packet from VC_i is dropped if:

$$(X > R) \text{ and } (Y_i * N_a / X > Z((K - R) / (X - R)))$$

Note that when the current buffer occupancy X exceeds the minimum threshold R, it is not always the case that a new packet is dropped. The load ratio in the above equation determines if VC_i is using more than a fair amount of buffer space. X/N_a is used as a measure of a fair allocation for each VC, and Z*((K-R)/(X-R)) is a drop threshold for the buffer. If the current buffer occupy(Y_i) is greater than this dynamic threshold times the fair allocation(X/N_a), then the new packet of that VC is dropped.

And also, this is the pseudo code from [\[Li's per-VC queuing\]](#):

```

When a cell in VCj comes to an ATM switch:
    if the cell is the first cell of a packet
        calculate  $Th = K * Q/N$ 
        if  $Q \geq Th$  and  $Q_j \geq Th$ 
            discard the cell
        else
            accept the cell into VCj's queue
             $Q_j = Q_j + 1$ 
    else
        if any cell of the packet has been discarded
            discard the cell
        else
            if  $Q \geq Q_{max}$ 
                discard the cell
            else
                accept the cell into VCj's queue
                 $Q_j = Q_j + 1$ 

```

Simulation results show that FBA improves both fairness and efficiency of TCP over UBR. See [\[Goyal's UBR+\]](#) and [\[Li's per-VC queuing\]](#).

So far we have introduced many dropping policies. Now lets make a short review:

- The earliest proposal is to use large buffer size without any dropping policies. This is impractical;
- Tail Drop just drops any incoming cells whenever the buffer is full. This results in the sharp throughput degradation of TCP over ATM compared with the conventional packet TCP;
- Drop from front makes use of TCP fast retransmit mechanism and has a better throughput than Tail Drop;
- Partial Packet Discard, which discards the subsequent cells of a cell-losing packet, has a better throughput than TD;
- Early Packet Drop, which uses a threshold in the buffer to ensure no partial packet be sent to the upper layer, is a milestone for improving the throughput of TCP over ATM; It can be further divided into EPD1 and EPD2, which choose different threshold setting mechanisms;
- All of the above policies have the unfairness problem. Selective Packet Dropping(or dropping based upon per-VC accounting) + EPD increases the fairness while maintains the high throughput;
- Fair Buffer Allocation(or per-VC queuing) + EPD further improves the fairness of TCP over ATM; however, sometimes it has a little bit lower throughput than SPD+EPD or EPD only [[Fang's EPD/EPD+FBA](#)];



[Return to Contents](#)

End system policies

As mentioned above in the TCP features section, the end system policies are the basic slow start and congestion avoidance and two improvements: FRR and SACK. Simulation results show that the following statements hold for these policies [[Morris' measurements](#)], [[Goyal's FRR UBR](#)], and [[Goyal's SACK UBR](#)].

For TCP Fast Retransmit and Recovery:

- For long latency connections(WAN), fast retransmit and recovery hurts the efficiency. This is because congestion typically results in multiple packets being dropped. FRR cannot recover from multiple losses and slow start is triggered, which is less efficient;
- FRR improves the efficiency of TCP over UBR for the LAN configuration. This is because the effect of multiple packet losses is much less visible in low latency connections;
- The addition of EPD with FRR results in a large improvement in both fairness and throughput.

For TCP Selective Acknowledgments:

- For most cases, for a given drop policy, SACK TCP provides higher efficiency than the corresponding drop policy in slow start TCP;
- For LANs, the effect of drop policies is very important and can dominate the effect of SACK;
- The throughput improvement provided by SACK is significant for WANs;
- The performance of SACK TCP can be improved by intelligent drop policies like EPD and Selective drop;
- The fairness values for selective drop are comparable to the values with the other TCP versions;

Buffer requirements

Though infinite buffer for the switches is not a way to solve the congestion control problems of TCP over ATM, the amount of a switch's buffer is definitely very important in order to maintain a high performance. If the buffer is too small and result a high Cell Loss Rate(CLR), according to either the slow start and congestion avoidance TCP, or the FRR TCP, or the SACK TCP, the CWND will be reduced at least by half. This results in a lower utilization of the link bandwidth and thus has bad performance.

One of the goal of studying buffer requirements of TCP over ATM is to find the minimum buffer requirements which ensure a zero CLR. Simulation results show that:

- TCP achieves maximum throughput when there are enough buffers at the switches;
- When the number of buffers is smaller, there is a large reduction in throughput even though CLR is very small;
- When capacity is varied, CLR exhibits high variance and is not related to TCP throughput. In general, CLR is not a good indicator of TCP level performance;

The above result can be explained as follows [\[Kalyanaraman's TCP over ABR\]](#):

CLR does not reflect TCP performance since higher CLR does not necessarily mean lower TCP throughput. The effect of cell loss depends not upon the number of cells lost but upon the number of timeouts. If a large number of cells are lost but there is only one timeout, the throughput degradation may not be that severe. On the other hand, even if a few cells are lost but the losses happen far apart triggering multiple timeouts, the throughput will severely degraded;

How many buffers are required for a UBR switch? [\[Kalyanaraman's buffer requirements\]](#) states that TCP using UBR requires network buffers equal to the sum of the maximum window sizes of all TCP connections to avoid cell loss. In this respect, UBR is not scalable for TCP. However, [\[Goyal's UBR buffer Requirements\]](#) shows that with sufficiently large buffers(0.5RTT or more), the performance of TCP-SACK over UBR with per-VC accounting is scalable with respect to the number of sources.

Guaranteed Rate

TCP performance over UBR can be degraded when high priority VBR uses up 100% of the link. In order to solve this problem, it is suggested the link reserves some bandwidth for UBR, this is called guaranteed rate UBR(GR). Providing a rate guarantee to the UBR class can ensure a continuous flow of TCP packets in the network. The guarantees provided are for the entire UBR class, and per-VC guarantees are not provided.

For a normal UBR, the GR is zero. When GR is grate than 0, it means that the UBR service class can be guaranteed that percent of total bandwidth. This parameter has the same function as the MCR of ABR service.

Comparing the simulation results got from TCP over pure UBR and TCP over GR UBR shows the following result [\[Goyal's GR UBR\]](#):

- For LANs, the dominating factor that effects the performance is the switch drop policy;
- For WANs, the dominating factor is the GR, and a GR of 0 hurts the TCP performance;
- For satellite networks, the TCP congestion control mechanism makes the most difference; SACK TCP produces the best results, and FRR TCP results in the worst performance;



[Return to Contents](#)

TCP over ABR

Since both TCP and ABR have their own mechanisms to do the flow control(in terms of TCP) or congestion control(in terms of ATM), running TCP over ABR is normally controlled by two sets of mechanisms: In TCP, the maximum traffic is controlled by the CWND, while in ABR, the traffic is controlled by MCR, Peak Cell Rate(PCR) and Allowed Cell Rate(ACR). A key factor that impacts user and network performance for TCP over ABR is how well the traffic management mechanism used in TCP end system and ATM end system and switch mesh together in providing good end to end performance.

Since ABR has more control to both the source/destination and switches than UBR has, the dropping policies, which are very important in TCP over UBR, are not so important here in TCP over ABR. In fact, a lot of TCP over ABR

work is done without taking any serious considerations for the above mentioned dropping policies. However, the TCP end system policies, which include the slow start and congestion avoidance, FRR, and SACK, are the same important as they are in TCP over UBR.

Buffer requirements

Simulations show the following statement about buffer requirement for TCP over ABR services [[Kalyanaraman's TCP over ABR](#)]:

- TCP achieves maximum throughput when there are enough buffers at the switches;
- When maximum throughput is achieved, the TCP sources are rate-limited by ABR rather than window-limited by the TCP;
- When throughput is reduced, the TCP sources are window-limited by TCP rather than rate-limited by ABR;
- When ABR capacity is varied, CLR exhibits high variance and is not related to TCP throughput;
- Large number of window-limited source increase TCP throughput. This is because the sum of the windows is larger when there are more sources;
- ABR buffer requirements is scalable in terms of the number of TCP sources;

It is proposed by [[Kalyanaraman's buffer requirements](#)] that in order to keep zero CLR, the buffer requirements of TCP over ABR is approximately $\{(a \cdot RTT + c \cdot \text{feedback delay}) \cdot \text{link bandwidth}\}$ for low coefficients a and c . And in most case, $a=3$ is big enough. Here the feedback delay is defined as the sum of delay from the bottleneck back to source and from source to the bottleneck.

It should be noted that though the maximum ABR network queues are small, the queues at the sources are high. Specifically, the maximum sum of the queues in the source and the switches is equal to the sum of the TCP window sizes of all TCP connections. In other words the buffering requirement for ABR becomes the same as that for UBR if we take the source queues into consideration. However, if the ABR is an end-to-end network, the source end system can directly flow control the TCP sources and can therefore use the local disk instead of use the source buffers.

Whenever we talk about buffer requirements for TCP or ATM switch, it seems that this statement is always correct: the more the buffer, the higher the throughput. However, [[Comer's TCP buffering](#)] shows us an anomalous behavior of TCP over ATM.

In that study, simulation results show that when we use 9188 (RFC 1626 proposes that the default IP maximum transmission unit(MTU) for use over ATM AAL5 is 9180) as the IP's MTU, for some fix-sized sender buffer, the larger the receiver buffer size, the lower the throughput. Further study show that large MTU size, as compared with the TCP buffer size, and mismatched TCP sender and receiver buffer sizes are the main cause of this anomalous behavior. TCP's "silly window syndrome" and the delayed ACKs (see RFC 813) and ATM's large MTU(9188) create *"a circular-wait situation not previously observed on a conventional LAN. The circular-wait, which can only be broken by the receiver's delayed ACK timer, creates a lock-step interaction in which the sender repeatedly experiences unnecessarily long delay before sending additional data. Thus, the network remains idle while data is waiting to be transmitted. As a result, TCP throughput decreases dramatically"*. A solution provided by the same study is to use a sender buffer size no smaller than $3 \cdot \text{MSS}$.

ABR With Varying Bandwidth

Another kind of experiment of TCP over ATM is to study the performance of ABR with a background application which has a higher priority and share the bandwidth with ABR [[Kalyanaraman's TCP over ABR2](#)] and [[Siu's TCP over ATM](#)]. This is normally the VBR. Normally, the VBR is simulated by an ON-OFF way. This is to say, the background is ON for a short period of time and then OFF for another period of time. During these two stages, it has different bandwidth requirements.

The simulation results show that the ON-OFF times, the feedback delays, and a switch scheme sensitive to variance in ABR load and capacity may combine to create worst case conditions where the ABR queues diverge.



[Return to Contents](#)

Comparison Between TCP Over ABR and UBR

So far we have list some results for TCP over UBR and TCP over ABR. Since normally we do not use both of them simultaneously(in fact, there is a discussion about whether we should support ABR or UBR), we need to make a comparison between them. This is also important since existing ABR rate-based congestion control schemes are more expensive to be implemented in ATM switches than UBR service with EPD schemes. We should know the degree of improvement in TCP performance by using a more expensive ABR congestion control in comparison with a UBR service.

[\[Li's TCP over ABR/UBR\]](#), [\[Siu's TCP over ATM\]](#) and [\[Kalyanaraman's buffer requirements\]](#) all give some results about comparing TCP over ABR and UBR + EPD. Here are the results:

- TCP achieves maximum throughput over both ABR and UBR when there are enough buffer at the switch and no cells are lost in the ATM networks;
- TCP connections can achieve similar throughput over UBR+EPD and ABR, but the degree of fairness is much better in ABR than in UBR+EPD. The buffer requirement in UBR+EPD is much larger that in ABR. Moreover, fairness in UBR+EPD degrades as the buffer size decreases.
- When congestion is moderate and no buffer overflow occurs at the ATM switches, UBR+EPD has similar performance as ABR. However, compared with ABR, the good performance of UBR+EPD comes at the expense of larger switch buffer size;
- With an effective congestion control switch algorithm in ABR, TCP can achieve good performance in terms of throughputs and fairness, even if the available bandwidth is time-varying;
- TCP over UBR+EPD can suffer severe degradation in throughputs and may not be able to provide fair bandwidth allocations among TCP sessions even with simple peer-to-peer configurations in an LAN environment, especially when the networks are very congested. Furthermore, TCP over UBR+EPD may experience the "beat down" problem -- a phenomenon that packets belonging to one TCP connection which traverses multiple congested links tend to have more chance to be dropped, resulting in unfairness;

It seems that we can draw the conclusion that generally speaking, TCP over ABR is better than TCP over UBR+EPD, both in efficiency and fairness. However, the degree of this difference is a topic that need further study.



[Return to Contents](#)

Summary

TCP is a very popular higher level network protocol, and ATM is a quickly developing high bandwidth transfer mode. In this paper, the combination of these two -- TCP over ATM -- is discussed.

We first introduce some of the TCP end system features, which include the slow start and congestion control, fast retransmit and recovery, and selective acknowledgment. These features play very important roles in improving the performance of TCP over ATM. We also discuss the ATM features, especially the ABR and UBR services. We know that UBR normally does very little congestion control and only drops the overflow data. Thus in order to improve the

performance of TCP over UBR, it is very important to have good dropping policies implemented in the UBR switches. For ABR, it has its own congestion control mechanism. While the switch dropping policies is important, the more important thing is how to mesh together TCP end system and ATM end system and switch.

Then, we give our performance metrics, which are efficiency and fairness. A short explanation is given for this choice.

Finally, we present the theoretical and observed simulation results of TCP over ABR and UBR. For TCP over UBR, switch dropping policies, TCP end system policies, guaranteed rate, and buffer requirements are discussed. Several dropping policies are introduced. Based on our performance metrics, these policies are compared. Buffer requirements for no cell loss is analyzed and simulation verified. For TCP over ABR, we focus mainly on the TCP end system policies and the buffer requirements for no cell loss or high throughput. We also make a comparison between TCP over UBR and TCP over ABR.



[Return to Contents](#)

Appendix of Acronyms

AAL5: ATM Adaptation Layer 5
ABR: Available Bit Rate
ACR: Allowed Cell Rate
ATM: Asynchronous Transfer Mode
CBR: Constant Bit Rate
CLR: Cell Loss Ratio
CWND: Congestion WiNDow
ER: Explicit Rate
EOM: End of Message
EPD: Early packet Drop
FBA: Fair Buffer Allocation
FRR: Fast Retransmit and Recovery
GR: Guarantee Rate
IP: Internet Protocol
LAN: Local Area Network
LANE: LAN Emulation
MCR: Minimum Cell Rate
MSS: Maximum Segment Size
MTU: Maximum Transmission Unit
PCR: Peak Cell Rate
PPD: Partial Packet Drop
QoS: Quality of Service
RCVWND: Receiver window
RFC: Request For Comments
RM: Resource Management
RTT: Round Trip Time
SACK: Selective ACKnowledgment
SPD: Selective Packet Drop
TCP: Transmission Control Protocol
TD: Tail Dropping
UBR: Unspecified Bit Rate
VBR: Variable Bit Rate

VC: Virtual Circuit

WAN: Wide Area Network



[Return to Contents](#)

References

1. *[Li's TCP over ABR/UBR]*

H. L. K. Siu, H. Tzeng, C. Ikeda and H. Suzuli

TCP performance over ABR and UBR services in ATM

Proceedings of IPCCC'96, March 1996

Summary: No EDP has much lower performance than EDP, but the later is unfair; EPD has lower performance than EPD + per-VC accounting, the later is also fairer; EPD + per-VC accounting has lower performance than EPD + per-VC queuing. The later is fair but sometime suffers from lower throughput.

2. *[Kalyanaraman's TCP over ABR]*

Shiv Kalyanaraman, Raj Jain, Sonia Fahmy, Rohit Goyal, Fang Lu, Saragur Srinidhi

Performance of TCP/IP over ABR Service on ATM Networks

Globecom'96, London, November 1996, <http://www.cis.ohio-state.edu/~jain/papers/globecom.htm>

Summary: It is possible to get maximum TCP throughput when there are enough buffers at the switches; However, when the number of buffers is smaller, there is a large reduction in TCP throughput even though the cell loss ratio is very small. The effect of factors which affect TCP throughput and fairness are studied. These factors include: TCP timer granularity, switch buffering, ABR parameters, and the cell drop policy at the switches.

3. *[Romanow's TCP over ATM]*

Allyn Romanow and Sally Floyd

Dynamics of TCP Traffic over ATM Networks

Computer Communication Review 1994; Vol 24; Number 4; pp. 79-88

Available from: ftp://ftp.ee.lbl.gov/papers/tcp_atm.ps.Z

Summary: The low performance of TCP over plain ATM is analyzed and PPD and EPD are proposed for solving this problem.

4. *[Fang's EPD/EPD+FBA]*

Chien Fang and Arthur Lin

On TCP Performance of UBR with EPD and UBR-EPD with a Fair Buffer Allocation Scheme

ATM-FORUM/95-1645

Summary: EPD and FBA are compared. It states that the throughput of FBA is a little lower than or at most the same as that of EPD, but the former has higher fairness. Also ABR is compared with these two, and it states that the throughput of FBA/EPD is at least the same good as ABR.

5. *[Heinanan's FBA]*

Juha Heinanan and Kalevi Kilkki

A fair buffer allocation scheme

Unpublished Manuscript

Summary: FBA is introduced and analyzed.

6. *[Li's per-VC queuing]*

H. Li, K. Siu, H. Tzeng, C. Ikeda and H. Suzuki

On TCP performance in ATM networks with per-VC early packet discard mechanisms

Computer Communication 19(1996), pp. 1065-1076

Summary: Per-VC accounting and per-VC queuing EPD are compared with the pure EPD.

7. [*Siu's TCP over ATM*]

Kai-Yeung Siu and Hong-Yi Tzeng

Performance of TCP over ATM with time-varying available bandwidth

Computer Communication 19(1996), pp. 927-936

Summary: Performance of TCP over ABR and UBR+EPD with background on-off VBR is studied.

8. [*Comer's TCP buffering*]

Douglas E. Comer and John C. Lin

TCP Buffering and Performance Over an ATM Network

Internetworking: Research and Experience, Vol. 6, 1995.

Summary: Sometimes more buffers result in lower throughput due to ATM's MSS and TCP's delayed ACK. One solution is to make sure the source has a $3 \times \text{MSS}$ buffer at least.

9. [*Lakshman's Drop from front*]

T.V.Lakshman, Arnold Neidhardt, and Tenuis J. Ott

The Drop from Front Strategy in TCP and in TCP over ATM

IEEE INFOCOM'96 Mar. 1996 pp.1242-1250

Summary: Pure/Partial drop from tail are compared with drop from front.

10. [*Goyal's UBR buffer Requirements*]

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

UBR Buffer requirements for TCP/IP over Satellite Networks

ATM-FORUM/97-0616

Summary: Experiments with both LEO and GEO satellite delays along with various buffer sizes and number of sources are done and conclude that with sufficiently large buffers(0.5RTT or more), the performance of TCP-SACK over UBR with per-VC accounting is scalable with respect to the number of sources.

11. [*Kalyanaraman's buffer requirements*]

Shiv Kalyanaraman, Raj Jain, Sonia Fahmy, and Rohit Goyal

Performance and Buffer Requirements of Internet Protocols over ATM ABR and UBR services

Available at: <http://www.cis.ohio-state.edu/~jain/papers/ieee-mag.htm>

Summary: This paper analyzes the performance of TCP/IP over ABR and UBR services and shows that ABR pushes congestion to the edges of the ATM network while UBR leaves it inside the ATM portion.

12. [*Kalyanaraman's TCP over ABR2*]

Shiv Kalyanaraman, Raj Jain, Sonia Fahmy, Rohit Goyal, and Jiangping Jiang

Performance of TCP over ABR on ATM backbone and with various VBR traffic patterns

Available from: http://www.cis.ohio-state.edu/~jain/papers/tcp_vbr.htm

Summary: Two studies are done here:(1)The performance of TCP over ABR in an ATM backbone; and (2)The effect of various patterns of VBR background traffic.

13. [*Goyal's SACK UBR*]

Rohit Goyal, Raj Jain, Shiv Kalyanaraman, Sonia Fahmy, Bobby Vandalore, and Xiangrong Cai

Selective Acknowledgments and UBR+ Drop Policies to Improve TCP/UBR Performance over Terrestrial and Satellite Networks

ATM-FORUM/97-0423

Summary: SACK TCP is seen to improve the performance of TCP over UBR. UBR+ drop policies are also essential to improve the performance of TCP over UBR.

14. [*Goyal's FRR UBR*]

Rohit Goyal, Raj Jain, Shiv Kalyanaraman and Sonia Fahmy

Further Results on UBR+: Effect of Fast Retransmit and Recovery

ATM-FORUM/96-1761

Summary: Effect of FRR over UBR is discussed;

15. [*Goyal's GR UBR*]

Rohit Goyal, Raj Jain, Shiv Kalyanaraman, Sonia Fahmy, Bobby Vandalore, and Xiangrong Cai

Guaranteed Rate for Improving TCP Performance on UBR+ over Terrestrial and satellite Networks

ATM-FORUM/97-0424

Summary: The effect of providing a guaranteed rate to the UBR service class on the TCP performance over UBR is analyzed here.

16. [*Goyal's UBR+*]

Rohit Goyal, Raj Jain, Shiv Kalyanaraman and Sonia Fahmy

UBR+: Improving Performance of TCP over ATM-UBR service

Proc. ICC'97, June 1997. Available from:

<ftp://ftp.netlab.ohio-state.edu/pub/jain/papers/icc97.htm>

Summary: Effect of EPD, SPD and FBA are discussed and compared.

17. [*Jacobson's congavoid*]

V. Jacobson

"Congestion Avoidance and Control"

Proceedings of SIGCOMM'88 Symposium, pp.314-332, August 1988

Available from: <ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z>

Summary: The original idea of TCP slow start and congestion avoidance is presented here.

18. [*RFC 1323*]

V. Jacobson, R. Braden, D. Broman

TCP Extensions for High Performance

Internet RFC 1323, May 1992

Summary: The idea of TCP fast retransmit and recovery is originally introduced here.

19. [*RFC 2018*]

M. Mathis, J. Madhavi, S. Floyd, A. Romanow

TCP Selective Acknowledgment Options

Internet RFC 2018, October 1996

Summary: The idea of TCP selective acknowledgment is introduced here.

20. [*Kawahara's selective cell discard*]

K. Kawahara, K. Kitajima, T. Takine, and Y. Oie

Performance Evaluation of Selective Cell Discard Schemes in ATM Networks

IEEE Infocom. Vol3, 1996. pp.1054-1061

Summary: Tail Drop, EPD1 and EPD2 are analyzed and compared.

21. [*Kalampoukas' comparison*]

Lampros Kalampoukas and Anujan Varma

Performance of TCP over Multi-Hop ATM Networks: A Comparative Study of ATM-Layer Congestion Control Schemes

International Conference on Communications, June 1995 pp.1472-1477

Summary: Plain ATM, EPD and FCVC are compared. Two approaches in setting the buffer threshold for EPD are discussed.

22. *[Fang's simulation]*

Chien Fang, Helen Chen and Jim Hutchins

A simulation Study of TCP Performance in ATM Networks

GLOBECOM Nov. 1994. Number 1. pp. 1217-1223

Summary: Comparison between pure UBR and Drop Tail, and between Drop Tail and Drop Whole.

23. *[Morris' measurements]*

R. Morris and H. T. Kung

Impact of ATM Switching and Flow Control on TCP performance: Measurements on An Experimental Switch

GLOBECOM '95 Number 2, pp. 888-892

Summary: TCP performance with and without ATM flow control are compared and analyzed; Why FRR does not work well with ATM is explained.

24. *[ATMF 4.0]*

ATM Forum

ATM Forum Traffic Management Specification Version 4.0, April 1996

<ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0056.000.ps>



[Return to Contents](#)

[Xiangrong Cai](#) Last modified: Mon Jul 19 13:52:02 EDT 1999