Preserving Bandwidth Using A Lazy Packet Discard Policy in ATM Networks

Azer Bestavros[a] and Gitae Kim[b]

[a]Computer Science Dept, Boston University, Boston, MA 02215
[b]Telcordia Technologies (Bellcore), Red Bank, NJ 07701

ABSTRACT

A number of recent studies have pointed out that TCP's performance over ATM networks tends to suffer, especially under congestion and switch buffer limitations. Switch-level enhancements and link-level flow control have been proposed to improve TCP's performance in ATM networks. Selective Cell Discard (SCD) and Early Packet Discard (EPD) ensure that partial packets are discarded from the network "as early as possible", thus reducing wasted bandwidth. While such techniques improve the achievable throughput, their effectiveness tends to degrade in multi-hop networks.

In this paper, we introduce Lazy Packet Discard (LPD), an AAL-level enhancement that improves effective throughput, reduces response time, and minimizes wasted bandwidth for TCP/IP over ATM. In contrast to the SCD and EPD policies, LPD delays as much as possible the removal from the network of cells belonging to a partially communicated packet. LPD preserves network bandwidth by keeping such cells alive and by ensuring that additional cells, obtained through Reed-Solomon block coding at the sender's AAL, are eventually transmitted to salvage the packet in question. We outline the implementation of LPD and show the performance advantage of TCP/LPD, compared to plain TCP and TCP/EPD through analysis and simulations.

Keywords: ATM networks; ATM Adaptation Layer (AAL); TCP/IP; performance evaluation.

1. INTRODUCTION

Asynchronous Transfer Mode (ATM) networks aim to: (1) minimize transmission latency through fast packet switching, and (2) deliver guaranteed QoS for a wide range of communication applications. Achieving the above goals while ensuring high utilization of network resources has proven to be quite challenging. To that end, The use of ATM's best-effort Available Bit Rate (ABR) service—as opposed to the guaranteed QoS through Constant Bit Rate (CBR) service, for example—has emerged as a potentially effective way to maximize the utilization of resources in ATM networks.

IP over ATM: Interest in ATM's ABR service is furthered by its compatibility with the best-effort philosophy underlying TCP/IP. TCP, which manages end-to-end feedback-based flow and congestion control algorithms for reliable data communications, has demonstrated its robustness in the Internet environment—an unreliable IP-based global network. Recent studies have shown that TCP/IP, when implemented over ATM networks, is susceptible to serious performance limitations, due mostly to the problem of packet fragmentation.

One of the defining characteristics of ATM technology is its use of a fixed 53-byte transfer unit—called a *cell*. The fragmentation of IP packet into cells at the boundaries of ATM networks raises significant challenges with regard to bandwidth preservation. In particular, when a portion of a packet is lost due to bit corruption or congestion, the whole packet must be discarded at reassembly time.

Azer Bestavros is Associate Professor of Computer Science at Boston University. He can be reached via email at `bestavros@cs.bu.edu`. Gitae Kim is a Research Scientist at Telcordia Technologies (formerly Bellcore Labs). He can be reached via email at `kgt@research.telcordia.com`. This research was partially supported by NSF grant CCR-9706685. Work was performed while second author was at Boston University.

Previous Work: A number of techniques have been proposed to improve the performance of TCP/IP over ATM. Selective Cell Discard (SCD)[1]—also called Partial Packet Discard (PPD) in[2]—and the Early Packet Discard (EPD)[2] propose switch-level packet discard to improve the effective throughput for TCP/IP traffic over ATM. Link-level flow control schemes including $N23Scheme$[3–5] have been introduced to prevent congestion at switches, and a transport layer enhancement, called TCP-Boston, has been introduced to enhance the performance of TCP over ATM.[6]

In SCD, once a cell c is dropped at a switch, all subsequent cells from the packet to which c belongs are dropped by the switch. In EPD, a more aggressive policy is used, whereby all cells from the packet to which c belongs are dropped, including those still in the switch buffer (*i.e.* preceding cells that were in the switch buffer at the time it was decided to drop c). The simulation results described in[2] and[7] show that both SCD and EPD improve the effective throughput of TCP/IP over ATMs. In particular, it was shown that the effective throughput achievable through the use of EPD approaches that of TCP/IP in the absence of fragmentation, but that it can still provide unacceptable performance when switch buffer sizes are small. Furthermore, in more realistic, multi-hop ATM networks, the cumulative wasted bandwidth (as a result of cells discarded through SCD or EPD) may be large, especially when the network resources are limited, and the impact of the ensuing packet losses on the performance of TCP is likely to be severe. To understand these limitations, it is important to realize that while dropping cells belonging to a packet at a congested switch preserves the bandwidth of that switch, it does not preserve the ABR bandwidth at all the switches preceding that (congested) switch along the virtual channel for the TCP connection. Moreover, any cells belonging to a corrupted packet, which have already been transmitted out of the congested switch, will continue to waste the bandwidth at all the switches following that (congested) switch.[1] Obviously, the more hops separating the TCP/IP source from the TCP/IP destination, the more wasted ABR bandwidth one would expect under the presence of congestion or resource limitation, even when these techniques are used.

Contribution: In this paper, we present an ATM Adaptation Layer (AAL) enhancement that alleviates the problem of IP packet fragmentation in ATM networks. A block coding technique—Information Dispersal Algorithm (IDA)[8]—coupled with AAL-to-AAL cell retransmissions allows recovery from partial packet delivery without any unnecessary waste in resource utilization, even under severe congestion or switch buffer limitations.

2. ATM ADAPTATION LAYER (AAL)

The main goal of AAL is to provide an ATM network with the ability to support a multi-application environment in accordance with the applications' specific QoS requirements.

The ATM standards groups divide the functions of AAL into two sub-layers: the Convergence Sub-layer (CS) and the Segmentation and Reassembly sub-layer (SAR) as shown in Figure 1. The convergence sub-layer, which is positioned closer to the application layer, is further divided into two sub-layers: the Common Part of CS (CPCS)[2] and the Service Specific CS (SSCS)[3]. The convergence sub-layer performs clock synchronization, sequencing, and forward error correction (FEC)[4] functions.

There are currently four types of protocols for AAL (*i.e.,* AAL1, AAL2, AAL3/4, and AAL5) to support different types of user traffic. The ITU and the ATM Forum have approved five classes of traffic, with labels of A through D, and X. Class A applications, requiring constant bit rate (CBR) service, are supported by the AAL1 protocol. Class B, C, D, and X applications, requiring variable bit rate (VBR) service, are handled by the rest of the AAL protocols.[9] The ABR service is supported by the AAL5 protocol.

Upon the arrival of a user message, which could range from one byte to several thousand bytes, the CS at the source AAL attaches a header and a trailer to the message as shown in Figure 2. Next, the message is *segmented* by SAR into data units with a fixed size ranging from 44 to 47 bytes depending on the type of traffic to which the message belongs. Next, another header is added to each data unit. Thus the source AAL's final product (*i.e.,* the data unit), which emerges from this series of operations is always 48-byte long (*i.e.*, an AAL PDU). The last operation shown at the bottom of Figure 2 is performed by the ATM layer. It adds a 5-byte header to the 48-byte payload, resulting in a 53-byte ATM cell.

[1] Cell interleaving and multiplexing can make this problem worse by increasing inter-cell distances, and its effect becomes more severe when switch buffers become limited.

[2] The CPCS supports generic functions common to more than one type of data applications.

[3] The SSCS supports specific aspect of a data application.

[4] The convergence sublayer provide FEC on user payload for high-quality audio-visual applications and on the AAL1 header.

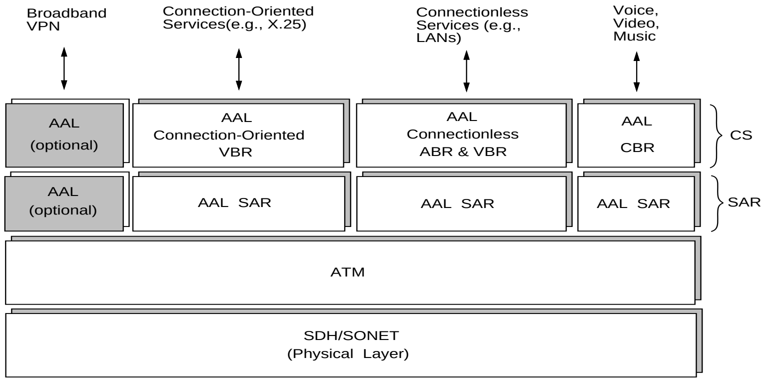

Figure 1. The ATM Layers

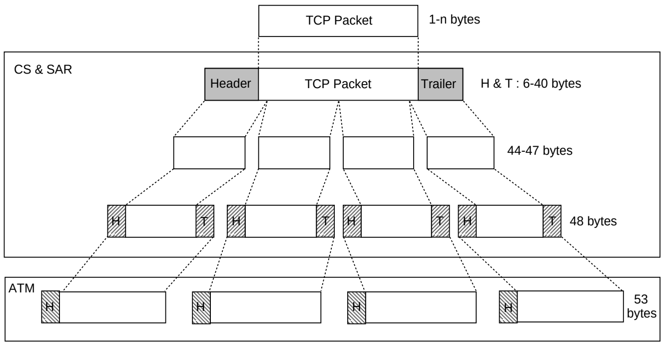

Figure 2. AAL Convergence and SAR

The sequence of operations at the source AAL is reversed to *reassemble* the original message at the destination AAL. When one or more cells are missing at the destination AAL, the SAR discards all the received cells that belong to the message whose cells were dropped in the network.

3. LAZY PACKET DISCARD (LPD)

In contrast to the Early Packet Discard (EPD) policy, which ensures that partial packets are discarded as early as possible, we propose the use of a Lazy Packet Discard (LPD) policy, which delays as much as possible the removal from the network of cells belonging to a partially communicated packet. LPD *preserves* network bandwidth by keeping such cells alive and by ensuring that additional cells, obtained through a block coding step at the sender's AAL, are eventually transmitted to salvage the packet in question. The block coding mechanism employed in LPD is based on the Information Dispersal Algorithm (IDA), which we present next.

3.1. Information Dispersal Algorithm (IDA)

There are two classes of redundancy-injecting coding techniques that are popularly used in data communication to handle transmission errors: convolutional and block coding. A subclass of block coding schemes that are known as Reed-Solomon codes—based on non-binary codes—does not distinguish between data and parity, *i.e.,* the redundancy is injected uniformly. The Information Dispersal Algorithm (IDA) of Michael O. Rabin[8] is an example of such techniques.

IDA has been previously shown to be a sound mechanism that considerably improves the performance of I/O systems, parallel/distributed storage devices,[10] and real-time broadcast disks.[11] The use of IDA for efficient routing in parallel architectures has also been explored in.[12]

To understand how IDA works in our context, consider a packet S. Let m be the size of S measured in "cell" units (*i.e.*, 48-byte units). Using IDA's *dispersal operation*, S could be processed to obtain a new N-cell packet S' (for any $N > m$), such that recombining *any* m of the N cells, using IDA's *reconstruction operation*, is sufficient to retrieve S.

3.2. Lazy Packet Discard: The Concept

The default assumption behind IDA is that the communication environment is not reliable, and thus, redundant information is injected at the time of packet transmission in anticipation of communication errors, such as bit corruptions and cell losses. In our strategy, we take the rather optimistic view that no cells will be corrupted or lost, and thus we do not inject any redundant cells at the time of initial transmission (*i.e.,* exactly m cells are transmitted). Instead, the unsent portion of the encoded message (*i.e.,* $N - m$ units) is kept by the sender for use *only when* communication failures occur and the transmission of additional cells is requested by the receiver. In particular, when cell losses are detected, the sender (upon a request from the receiver) transmits extra cells from its cache of unsent cells that belong to the packet in question. These additional cells are used by the receiver to recover the partially received packet. Thus, a partially received packet is "lazily discarded" by the receiver AAL only when its redemption through additional cell transmissions is not possible (*e.g.* is timed out).

The main reason for taking such an optimistic stance is to preserve network resources (*i.e.,* maximize effective throughout) by excluding redundancy from transmitted messages. Moreover, the ABR and UBR services in ATM networks are aimed at best-effort traffic (*e.g.,* TCP or UDP), which are not subject to the stringent latency or reliability QoS requirements that necessitate the use of FEC. The purpose of using FEC redundancy is to improve transmission reliability with minimum latency. In ATM networks, traffic classes that require such transmission characteristics are supported by CBR and VBR services. Since TCP/IP relies on the use of temporal redundancy (*i.e.,* retransmissions) to maintain data integrity over unreliable communication channels, our optimistic approach, which relies on message retransmissions for data integrity, could support TCP/IP traffic seamlessly.

As pointed out in,[13] packet fragmentation can introduce a waste in bandwidth, and it is indeed the main cause of low performance when TCP/IP packets are transmitted through the ATM networks. Fragmentation is inevitable when a packet traverses a network with a smaller MTU. There is a space efficiency issue in making packet sizes small in order to prevent fragmentation. The byte ratio of payload to the protocol overhead in current TCP/IP packet is approximately 10:1.[5] As the packet size becomes smaller, the ratio of bytes wasted by the protocol increases. Apart from the efficiency issue, under current TCP/IP protocol standards as well as IP*ng*, even the headers alone cannot fit into the 41-byte[6] constraint of ATM standards to prevent fragmentation.

Our technique is (in a sense) to emulate small TCP packet sizes within ATM environments by getting rid of the overhead of TCP/IP headers, and by taking advantage of the semantics of SAR in the AAL. As mentioned above, the SAR at the destination discards the cells if it concludes that there were cell drops—there is no way of making use of those successful cells in current AAL environment. The use of the IDA block coding scheme makes it possible for the SAR to redeem those successful cells through AAL-to-AAL cell requests and retransmissions. It assumes that flow and congestion control is done on an end-to-end basis by upper-layer protocols (*i.e.*, TCP in this case), and does not alter the traffic dynamics managed by TCP's flow and congestion control algorithms.

3.3. Lazy Packet Discard: Implementation Overview

As explained before, the implementation of LPD requires (1) the implementation of an IDA block coding module, (2) the implementation of data structures and buffer spaces at the source and destination AALs, and (3) the modification of the CS and SAR functionalities in AAL to enable additional cell transmissions for partially received packets. In this section, we discuss our implementation of the above requirements. In addition, we present some optimizations that LPD enables for TCP traffic streams. Figure 3 shows an overview of the protocol. For simplicity, the length of headers and trailers is assumed to be zero, and operations thereon are omitted (unless meaningful to our protocol).

Sending Packets: Upon the arrival of a new TCP packet of size m (cells), the sender AAL assigns an AAL-level sequence# to the packet, and then forwards the packet to the IDA encoder, where the packet is processed to produce N ($N > m$) cells. Of these N cells, m cells are injected into the ATM network and the remaining $(N - m)$ cells are set-aside for later transmission,[7] should less than m cells be received by the destination AAL. An identification of this packet, along with a pointer to the buffer space in which the $(N - m)$ "set-aside" cells are stored, is inserted into

[5] It is assumed that TCP segments are not fragmented in IP network whose MTU is 512 bytes, and the header size of TCP and IP is 24 bytes.

[6] This value comes from a simple calculation from Figure 2.

[7] Any $(N - m)$ cells out of the N cells produced by IDA can be chosen for that purpose.

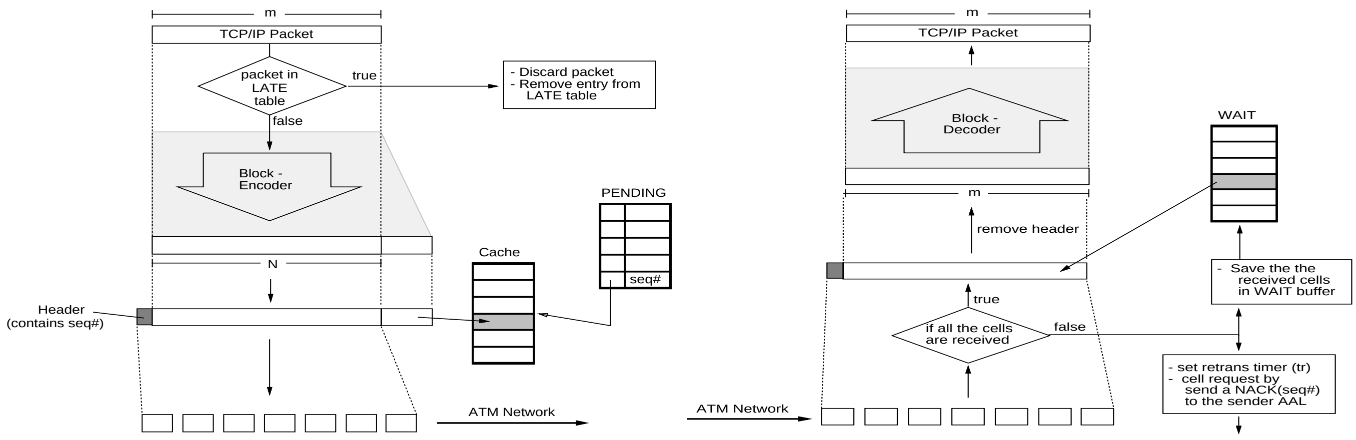

Figure 3. The Lazy Packet Discard (LPD) Protocol

the *PENDING* queue. Thus, an entry in the *PENDING* queue corresponds to a packet *en route* to the destination AAL. The *PENDING* queue is indexed by the AAL-level sequence number. When the *PENDING* queue is full, an entry is deleted in FIFO fashion. For the identification of a packet, a triplet, *<segment#, dest-port#, dest-IP-Addr>* is used to record the packet's TCP sequence#, destination TCP port#, and destination IP address.

Receiving Packets: When the destination AAL receives a cell from the ATM layer, it checks if enough cells (*i.e.* m cells) are received to be able to reconstruct the original packet. If so, the cells are reassembled to produce the original packet. If the destination AAL finds that cells are missing, then instead of discarding all the received cells, it saves them in a buffer called *WAIT*, and sends a request message to the source AAL. When the missing cells are received, cells in the *WAIT* buffer are forwarded to the IDA decoder along with the newly arriving cell(s), and the *WAIT* buffer space is freed.

Recovering Packets: Partially received packets are recovered through the transmission of additional "set aside" cells. The destination AAL requests these cells by communicating with the source AAL the number of cells missing and the AAL-level sequence#, to which the cells belong. Upon receiving such a request, the source AAL locates the cells in its storage (indexed by the AAL-level sequence#). This is done by locating the appropriate entry in the *PENDING* queue and using that entry to fetch the requested cells, which are then injected into the ATM network. If an entry in the *PENDING* queue corresponding to the requested AAL-level sequence# cannot be found, then no additional cell transmission is possible—and the recovery of the partially received packet will fail, resulting in the eventual "lazy" discarding of that packet.

3.4. Lazy Packet Discard: Implementation Issues

There are a number of important issues to be addressed when implementing the LPD protocol. These include: the segment numbering scheme to be employed, the blocking factor and redundancy levels in IDA, the buffer space management in sender and receiver AALs, the AAL-to-AAL additional cell transmissions needed by LPD, as well as optimization opportunities for specific traffic streams (e.g., TCP) enabled via LPD. In this section, we address some of these issues and present implementation options for efficient management of the LPD protocol.

AAL Segment Numbering: The AAL sequence# is needed for both the sender and receiver AALs. The sender AAL needs the AAL sequence# to be able to index and access the unsent portion of the encoded packet. Likewise, the receiver AAL needs a way of referring the attaché of cells that are requested for retransmission,[8] and it uses the AAL sequence# for that purpose. The purpose of the LPD header is to carry the sequence# to the sender AAL,

[8]The term "retransmission" is somewhat misleading since the sender never replies with cells that have been transmitted before. Nevertheless, for brevity, we use this term throughout when we refer to the AAL-to-AAL additional cell transmissions.

and thus, the size of header is mainly determined by the space occupied by the AAL sequence#. A 32-bit cyclic numbering scheme (similar to that commonly used for TCP sequence numbering) is used for that purpose.

Our implementation of LPD illustrated in Figure 3 and used in our simulation experiments, assumes that the LPD header containing the AAL sequence# is prepended to the encoded payload *before* the segmentation phase. Under such an implementation, if the cell that contains the LPD header is lost during transmission, the receiver AAL has no way of finding out the sequence numbers and thus has to discard all the cells for that TCP packet. An alternative implementation strategy is to prepend the LPD header *after* the segmentation phase. In this case, an LPD header is attached to each of the cells segmented, and thus, the segmentation phase needs to produce smaller units so that the resulting units' size after the header attachment would fit into the 53-byte cell size. Since the LPD header is attached to each of the cells, this approach gives less space efficiency, but has the advantage that any cell can be dropped during transmission since every cell carries the the LPD sequence#.

Blocking Factor and Redundancy Levels: The choice of blocking factor and redundancy level employed by the IDA encoder at the sender AAL has an important effect on the performance of LPD. The blocking factor refers to the number of packets to be used as input to the encoding process. In our implementation (and discussion above), the blocking factor was assumed to be 1. By increasing the blocking factor, the space overhead required by the LPD header[9] decreases, and the probability of cell discards due to header-cell loss decreases, accordingly. The disadvantages of the large blocking factor are increased average message response time, and increased coding complexity which could cause latency problems in high-speed networks.

The choice of redundancy level refers to the difference in size between the original block and the encoded block, *i.e.,* the unsent portion of the encoded block [= $(N - m)$] at the sender AAL. The sender AAL needs to set aside enough redundant cells so that it can use them when the receiver AAL requests additional cell transmissions. Obviously, increasing the redundancy level results in larger buffer sizes at the sender AAL.[10] But, when the network is congested, the receiver AAL is likely to request a large number of additional cell transmissions. In extreme cases, the receiver's request(s) for additional transmissions could result in exhausting the supply of set-aside cells at the sender AAL, making the recovery of partially transmitted packets impossible. Thus, the choice of redundancy level should be in accordance with the characteristics of the network under which the protocol is to be deployed, the classes of applications it is targeting, and buffer space availability.

Of course, it is possible for the encoding/decoding phases to have multiple blocking factors and redundancy levels, which can be adjusted dynamically on a per-virtual-channel basis. Such an approach is likely to complicate the encoding scheme (*e.g.*, using IDA, it will require the maintenance of multiple versions of the dispersal and reconstruction matrices).

Buffer Storage Management: As we eluded earlier, using LPD, the sender and receiver AAL's must manage a number of additional buffers. For example, the sender AAL requires the maintenance of a queue, called *PENDING*, to store indices to the set-aside (unsent) portions of encoded packet. In our implementation, we used a FIFO replacement discipline when inserting new entries to the *PENDING* queue. Other disciplines may be more appropriate. For example, it may be advantageous to use a replacement discipline that gives preference to packets that belong to a specific type of traffic (*e.g.*, TCP vs. UDP packets).

Another issue related to the management of buffer space is that of garbage collection. For example, the receiver AAL must have a mechanism whereby partially received packets that cannot be recovered are eventually discarded. In our implementation, the "lazy" discarding of partially received packets at the destination AAL is done through the use of a timeout mechanism. When a request for retransmission is issued, the destination AAL sets a timer with a value t_r. When the timer expires and, still, not enough cells are available to reconstruct the packet, the process of requesting retransmissions and of setting up the timer is repeated. When an upper bound on the number of such requests is reached,[11] the partially received packet is discarded.

---

[9] Assuming that AAL sequence numbers are attached *before* the segmentation phase as shown in Figure 3.

[10] Alternatively, if buffer space is constant, then increasing the redundancy level results in decreasing the number of packets in the *PENDING* queue, and hence decreasing the likelihood of packet recovery.

[11] This would be an indication that severe congestion exists, or that the source AAL has dispensed of the packet by removing it from the *PENDING* queue.

TCP-specific Optimizations: The implementation of LPD enables some subtle optimizations to the TCP protocol. In particular, using LPD, information pertaining to the successful delivery of packets could be used to prevent unnecessary TCP packet retransmissions (due to repeated acks or timeouts, for example).

To explain this point, consider the following scenario. A packet transmitted through a source AAL is partially received by the destination AAL. As a result, the destination AAL requests additional cells to recover the packet. In response, the source AAL sends additional cells and the packet is reassembled successfully. While this packet recovery was in progress, the source TCP may have received duplicate acknowledgments and, as a result, decides to retransmit that packet again. When such a packet reaches the source AAL, it is possible to conclude that it is a duplicate packet, whose transmission is *not* necessary, since an identical packet[12] was delivered successfully (albeit late). We have implemented this optimization for TCP over AAL/LPD using a simple data structure in which we keep the identity of "late" packets known to have been successfully transmitted to the destination AAL.[13]

4. PERFORMANCE MEASUREMENTS

In this section, performance evaluation is carried out using both analysis and simulation to study the effectiveness of our LPD AAL enhancement and to compare the performance of TCP Vegas over ATM with LPD enhancement (Vegas/LPD) against that of TCP Vegas over ATM with EPD enhancement (Vegas/EPD) and that of TCP Vegas over ATM with no enhancement (Vegas). The performance criteria we consider are effective throughput (*aka* goodput), cell failure rate, and response time.

4.1. Analysis of Wasted Bandwidth

The measurement of effective throughput *at a bottleneck link* in a network is widely used to characterize the useful utilization of various network resources (*e.g.*, link and switch buffers). However, this metric alone cannot characterize adequately the bandwidth wasted *throughout the network* due to the communication of non-useful payloads (*e.g.* partial or duplicate packet deliveries), especially in a multi-hop environment. In this section, we derive a simple, yet general formulation of the wasted bandwidth in a multi-hop network. Our formulation allows the estimation of wasted bandwidth throughout a multi-hop network, using measurements obtained (say) via simulation of a network with a single, fixed congested switch. The metric used in our analysis is the *Byte-Hop product* (BHops). When data (packet, cell, *etc.*) of size n bytes travels for h hops, its Byte-Hop product (*i.e.*, the bandwidth consumed by the data), is $n \times h$.

Consider a *single* source-sink TCP connection in a multi-hop network such that the number of hops between the source and the sink (*i.e.*, the length of the virtual channel) is h. Furthermore, assume that the virtual channel that connects the source and the sink traverses a congested switch after an average of δ hops.[14] For simplicity (and without loss of generality), we will assume that there is only one such congested switch along the virtual channel and that its state of congestion (*i.e.* the probability that a cell will be dropped at that switch) is independent of the TCP protocol used by the source-to-sink connection under consideration.[15] Let $(1-p)$ denote that probability (*i.e.* p denotes the probability that a cell will not be dropped at the congested switch).

Given the above definitions and assumptions, we are now ready to define the wasted BHops caused by cell drops for each of the three protocols, assuming that the total number of packets to be transferred is t and that the number of cells per packet is m.

Vegas/LPD: Since cells are dropped with a probability $(1-p)$ after an average of δ hops from the source, it follows that the total wasted BHops caused by the protocol is simply:

$$Waste_{LPD} \quad = \quad (1-p) \cdot \delta \cdot m \cdot t \tag{1}$$

[12] *i.e.*, with the same segment#, dest-port#, and dest-IP-addr

[13] Figure 3 reflects our TCP implementation, where the redundant TCP/IP packets are discarded by checking the LATE list in the sender AAL.

[14] Another way of stating this is that the average distance that a cell must travel until it is dropped (*en route* from source to sink) is δ.

[15] This assumption will hold if the volume of ABR traffic at the switch is much larger than the amount of traffic going through the connection under consideration. If this condition is not satisfied, then our assumption will simply favor Vegas and Vegas/EPD, and thus is a conservative assumption to establish LPD's superiority.

Vegas/EPD: Since all cells belonging to a packet are discarded when any cell from that packet is dropped at the congested switch, it follows that the total wasted BHops is given by:

$$Waste_{EPD} \quad = \quad (1-p^m)\cdot\delta\cdot m\cdot t \tag{2}$$

Vegas: Since no cells other than the dropped one(s) are discarded at the congested switch, it follows that the total wasted BHops is given by:

$$Waste_{Vegas} \quad = \quad (1-p^m)\cdot((1-p)\cdot\delta+p\cdot h)\cdot m\cdot t \tag{3}$$

When we assume that each switch on the virtual channel between the source and sink has an equal probability of being the congested switch, the average distance that a cell travels in the path before being dropped (if it is dropped) reduces to $\frac{1}{2}h$ hops (away from the source). Substituting in the above equations we get the following ratios of wasted BHops for the three cases.

$$Waste_{LPD} : Waste_{EPD} : Waste_{Vegas} \quad = \quad 1 : (1-p^m)\left(\frac{1}{1-p}\right) : (1-p^m)\left(\frac{1+p}{1-p}\right) \tag{4}$$

where $q = (1-p^m)$ represents the probability of a packet loss as a result of one or more cells being dropped at the congested switch.

For values of $p \approx 1$ (*i.e.* small cell drop rates), equation 4 reduces to:

$$Waste_{LPD} : Waste_{EPD} : Waste_{Vegas} \quad = \quad 1 : m : 2m \tag{5}$$

On the other hand, for smaller values of p (*i.e.* large cell drop rates), equation 4 reduces to:

$$Waste_{LPD} : Waste_{EPD} : Waste_{Vegas} \quad = \quad 1 : \frac{1}{1-p} : \frac{1+p}{1-p} \tag{6}$$

By measuring the cell loss rate $(1-p)$ at a congested switch (either in a real system or in a simulated system), one could predict, using the above equation, the ratio between the bandwidth wasted by each one of the three protocols in a multi-hop network. For example, if the cell drop rate is measured to be 10% (*i.e.* $p = 0.9$) and $m = 10$, we get a value of $q = 0.18$. Substituting in equation 4, we get:

$$Waste_{LPD} : Waste_{EPD} : Waste_{Vegas} \quad = \quad 1 : 6.51 : 12.38 \tag{7}$$

from which we infer that Vegas wastes almost twice as much bandwidth as Vegas/EDP, which in turn wastes more than six times as much bandwidth as that wasted by Vegas/LPD.

Equation 4 indicates that both Vegas and Vegas/EPD are highly susceptible to fragmentation, since the amount of BHops they waste increases with m, the fragmentation level. On the contrary, Vegas/LPD is tolerant to fragmentation and is only sensitive to the cell loss ratio $(1-p)$. Furthermore, equation 5 shows that Vegas and Vegas/EPD are especially susceptible to fragmentation when the network is not congested. When the network is congested, equation 6 indicates that, as expected, all three protocols perform poorly, with Vegas/LPD providing the least wasted BHops.

4.2. Simulation Experiments

In this section, we present the results of simulation experiments that were conducted to evaluate and compare the performance of plain Vegas, Vegas/EPD, and Vegas/LPD.

4.2.1. Simulation Set-up

Topology: Our network model simulates an ATM ABR-like environment, *i.e.,* cell switching without network-level flow and congestion control. The network topology used in our simulation is shown in Figure 4. 32 servers, each of which is competing for network resources, are evenly connected to four receivers via a bottleneck link of 155 Mbps. The average latency between a client and a server is 450μ sec, which translates to a radius of 30 miles.[16] The link bandwidth and latency are chosen to study the effectiveness of TCP's feedback-based flow and congestion control scheme under a high speed, large-scale Metropolitan Area Network (MAN).

[16] The propagation speed of the link used to estimate the radius is 2.15×10^5 km per second. The bandwidth-delay product of the network is approximately 330 cells.

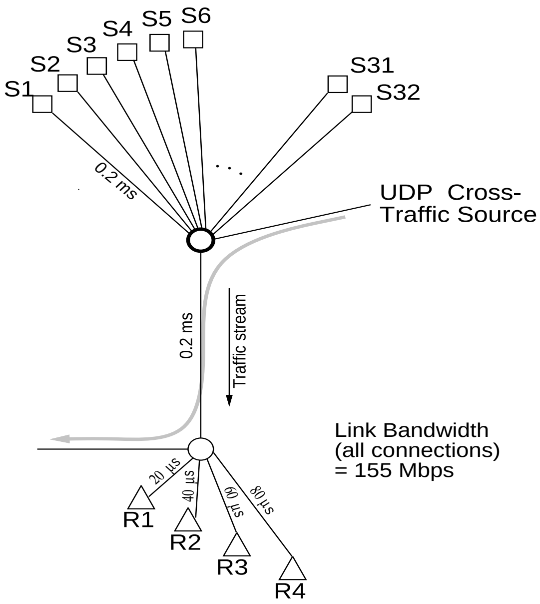

Figure 4. Configuration of simulated network.

Traffic: Our simulation experiments used a total of 32 TCP connections, each of which is established between a server and a receiver, and servers are assigned evenly to the four receivers. In each experiment, servers transmit infinite streams of data for a total of 20 seconds (to transfer a total of 350MB of data, on average). In addition to these TCP connections, a rate controlled UDP source generates data packets at a variable rate to emulate cross traffic passing through the bottleneck link. The UDP source transmits packets in an ON/OFF fashion. In particular, the transmission of a packet by a UDP source (the ON time) is followed by an OFF period, whose distribution is drawn from a Pareto function with $\alpha = 1.35$ and $k = 20$ to generate an average OFF time of 77 μ second. The purpose of adding the ON/OFF UDP cross traffic is to evaluate the effectiveness of TCP flow and congestion control algorithms when subjected to non TCP-controlled traffic. The use of Pareto-distributed OFF times allows us to model the self-similarity of network traffic as characterized in.[14,15]

Simulation Parameters: Our experiments were run over a range of bottleneck switch buffer sizes, TCP packet sizes, and TCP window sizes. Four different packet sizes were chosen to reflect the maximum transfer unit (MTU) of popular standards: 576 bytes for IP packets, 1,500 bytes for Ethernet, 4,470 bytes for FDDI link standards,[16] and 9,180 bytes which is the recommended packet size for IP over ATM.[17] The values chosen for the TCP window size are 8 kB, 32 kB, and 64 kB. Bottleneck ATM switch buffer sizes used are 64, 256, 512, 1,000, 2,000, and 4,000 cells. Note that the buffer size corresponding to the bandwidth-delay product of the simulated network is 330 cells. For non-bottleneck switches, the buffer spaces are set to be large enough to prevent any cell losses.

Simulation Engine: The LBNL Network Simulator (*ns*) is used as the base engine in our simulator. We modified *ns* extensively to add ATM functionalities. In particular, we added: (1) ATM's cell switching functionality, (2) the essential AAL5 functionalities (*e.g.,* packet fragmentation and reassembly), and (3) AAL5 extensions, including LPD (*i.e.,* cell retransmission and deferred packet discard) and EPD functionalities.

The ATM switch architecture we adopted in our simulation engine is a simple, 32-port, output-buffered single-stage switch.[18] When the output port is busy, a cell at the input port is queued into the output buffer of the simulated switch. When the output buffer is full, an incoming cell destined to the output port is dropped. The output buffer is managed using FIFO scheduling, and cells in input ports are served in a round-robin fashion to ensure fairness.

The ATM Adaptation Layer (AAL) implements the basic functions found in AAL5, namely fragmentation and reconstruction of IP packets.[19,20] AAL divides IP packets into 48-byte units for transmission as ATM cells, and appends 0 to 47 bytes of padding to the end of data. To support LPD, the destination AAL does not discard cells received even when cell losses are encountered. Incomplete packets are discarded by the destination AAL for non-LPD implementations. In our simulations, an AAL module is placed at each of the server and client sites.

The simulation package was also enhanced to facilitate the gathering of traffic measurements and for the evaluation of additional statistics. To complement the existing Reno-like TCP in *ns*, we added Vegas-like flow and congestion control with careful adjustments of timer granularity[17] and timeout timer for maximum compatibility with TCP in high-speed links.

4.2.2. Simulation Results

The performance metrics reported in this section are the effective throughput, packet response time, and cell failure rate. Each of the performance graphs presented in this section portrays one of these performance metrics (on the y-axis) as a function of the switch buffer size (on the x-axis). The function is shown as a family of curves, each corresponding to the performance of plain Vegas, Vegas/EPD, and Vegas/LPD.

Effective Throughput: The effective throughput (*aka,* goodput) refers to the throughput considering only the bytes that are useful at the upper layer. Figure 5 gives a comparison of goodput achieved by the three methods, under 64 kB window size for packet sizes of 1,500 bytes (left) and 9,180 bytes (right), where the effective throughput of the three different methods is plotted as a function of the switch buffer size. Vegas/LPD clearly outperforms both plain Vegas and Vegas/EPD under limited buffer space, with wider margins for large packet sizes. The two curves of Vegas/LPD and Vegas/EPD meet at a buffer size of 70 kB, after which the two methods maintain similar throughput levels. The throughput of plain Vegas starts very low at small buffer sizes, but continues to improve throughout the entire span of buffer sizes, approaching the performance of the other two methods at the maximum buffer size.

In our experiments, the effective throughput is closely related to the number of cells discarded by the receiver AAL or by the EPD switch, and the responsiveness of the TCP flow and congestion control scheme (i.e., link idle time). Our simulation results show that, under extreme buffer shortages, all three methods exhibit decreased performance, showing the vulnerability of TCP in an ATM environment. Both LPD and EPD improve the performance with a wide margin compared to plain Vegas, but LPD exhibits more resilience as the switch buffer size decreases, by showing the most graceful degradation in performance among the three methods. As the packet size gets larger, the throughput of plain Vegas becomes unacceptable. Our traces show that the low throughput of plain Vegas and of Vegas/EPD under large packet sizes with small buffer buffer spaces is caused by repeated packet discards that result from repeateded cell losses at the bottleneck switch, as well as the link idle time that also affects Vegas/LPD.

Cell Failure Rate: The cell failure rate is defined as the sum of *cell loss rate* and *cell discard rate.* In other words, it is a measure of the percentage of cells that waste the bandwidth by being discarded or being lost (in either the switches or the receiver AAL). Figure 6 shows the cell failure rates for the three methods. As the size of the switch buffer decreases, the cell failure rate for all three methods increases slowly until the buffer size reaches 70 kB for both 1,500-byte and 9,180-byte packet sizes. From that point on, the cell failure rate begins to grow steeply, and eventually grows super exponentially toward the marginal buffer size for Vegas/EPD and plain Vegas under 9,180-byte packet size.

The ratio of cell failure rates between Vegas/LPD and the other two methods increase toward marginal buffer sizes. This increase is more pronounced as the packet size increases. This is because, as the packet size increases, the number of cells per packet increases, and thus, the chance of a cell in a packet being lost or discarded increases in both plain Vegas and Vegas/EPD, which results in an increased number of cells that are being discarded or lost. In contrast, the cell failure rate of the Vegas/LPD is governed only by the cell loss rate at the bottleneck switch, thus, giving a smoother increase in the cell failure rate as the switch buffer space becomes limited.

[17] Coarse-grain timer is set to 5 msec.

Packet response time: We define the packet response time to be the average time taken from the point a TCP sender transmits a new packet to the time a TCP receiver receives it. If a packet is lost on the way, the time that the TCP sender transmits the initial packet is used as the start time. Thus, a packet response time is proportional to the number of packet transmissions/retransmissions necessary to deliver that packet. Figure 7 shows a partial view of the packet response time achieved by the three methods, under a 64 kB window size, for packet sizes of 1,500 bytes (left) and 9,180 bytes (right), where the packet response times are plotted as a function of the switch buffer size. In general, as the switch buffer size becomes larger, the response time increases accordingly, due to the fact that the average number of cells queued at each switch increases as the switch buffer size becomes large.

This phenomenon is well depicted in the two response time plots (and has been documented in other studies[15] as well). The plots also imply that the queuing delay increases linearly as the buffer size increases. The packet response time characteristics for the three methods show that Vegas/LPD outperforms both plain Vegas and Vegas/EPD, under the entire buffer range, though the gaps between each method are marginal especially under small TCP packet sizes. With a large packet size, the performance of the plain Vegas approaches that of Vegas/EPD at the extremely limited buffer region. On the other hand, the packet response time for Vegas/LPD continues to fall, creating a larger gap with the other two methods as the buffer size approaches the extreme limit. The main cause of this large gap is due to the increased chance of repeated cell losses for a packet at the bottleneck switch. A cell loss requires at least another RTT for packet retransmission, resulting in multiple RTTs for each packet transmission. In contrast, Vegas/LPD tend to accumulate the partial packets as the retransmission is repeated, instead of discarding them, resulting in improved packet delivery time.

So far, the results we have presented for the three methods were under a large TCP window size (*i.e.,* 64 kB). The performance characteristics for the three methods under smaller TCP window sizes (*i.e.,* 8 kB and 32 kB) show similar pattern as that of 64 kB, with an exception that the performance of the three methods gradually converges as the TCP window size decreases.

5. SUMMARY AND FUTURE WORK

In this paper we presented a novel method, Lazy Packet Discard (LPD), an AAL-level bandwidth preserving technique for datagram transmission in ATM networks. We presented the concept of LPD, along with its various implementation issues. We focused our performance evaluation on TCP/IP traffic over ATM networks. We have shown the performance superiority of LPD when compared to plain Vegas and Vegas/EPD, which are more vulnerable to fragmentation. Our current work includes studying the dynamic redundancy control algorithms that can be implemented to control the retransmission mechanism of LPD, so that the use of well-controlled FEC can further enhance its performance while minimizing space redundancy.

REFERENCES

1. G. Armitage and K. Adams, "Packet Reassembly During Cell Loss," *IEEE Network Mag.* **7**, pp. 26–34, September 1993.
2. A. Romanow and S. Floyd, "Dynamics of TCP Traffic over ATM Networks," *IEEE Journal on Selected Areas in Communication* **13**, pp. 633–641, May 1995.
3. H. T. Kung and A. Chapman, "The FCVC (Flow-Controlled Virtual Channels) Proposal for ATM Networks," in *Proceedings of the 1993 International Conference on Network Protocols*, pp. 116–127, October 1993.
4. H. T. Kung, T. Blackwell, and A. Chapman, "Credit-Based Flow Control for ATM Networks: Credit Update Protocol, Adaptive Credit Allocation, and Statistical Multiplexing," in *Proceedings of SIGCOMM'94*, September 1994.
5. G. Varghese, C. Ozveren, and R. Simcoe, "Reliable and Efficient Hop-by-Hop Flow Control," in *Proceedings of SIGCOMM'94*, September 1994.
6. A. Bestavros and G. Kim, "TCP Boston: A Fragmentation-tolerant TCP Protocol for ATM Networks," in *Proceedings of IEEE INFOCOM'97*, (Kobe, Japan), April 1997.
7. L. Kalampoukas and A. Varma, "Performance of TCP over Multi-Hop ATM Networks: A Comparative Study of ATM-Layer Congestion Control Schemes," Tech. Rep. UCSC-CRL-95-13, Computer Engineering and Information Sciences, University of California, Santa Cruz, Santa Cruz, CA, 1995.
8. M. O. Rabin, "Efficient Dispersal of Information for Security, Load Balancing and Fault Tolerance," *Journal of the Association for Computing Machinery* **36**, pp. 335–348, April 1989.
9. U. Black, *ATM: Foundation For Broadband Networks*, Prentice Hall, Inc., 1995.
10. A. Bestavros, "IDA-based Disk Arrays," Technical Memorandum 45312-890707-01TM, AT&T, Bell Laboratories, Department 45312, Holmdel, NJ, July 1989.
11. A. Bestavros, "AIDA-based Real-Time Fault-Tolerant Broadcast Disks," in *Proceedings of RTAS'96: The 1996 IEEE Real-Time Technology and Applications Symposium*, (Boston, Massachusetts), May 1996.

12. Y.-D. Lyuu, "Fast fault-tolerant parallel communication and on-line maintenance using information dispersal," Tech. Rep. TR-19-1989, Harvard University, Cambridge, Massachusetts, October 1989.
13. C. Kent and J. Mogul, "Fragmentation Cosidered Harmful," in *ACM SIGCOMM'87*, pp. 390–401, Sugust 1987.
14. W. Willinger, M. Taqqu, R. Sherman, and D. Wilson, "Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level," in *Proc. ACM SIGCOMM '95*, pp. 100–113, 1995.
15. K. Park, G. Kim, and M. E. Crovella, "The Effects of Traffic Self-Similarity on TCP Performance," tech. rep., Boston University Computer Science Department, 1996.
16. S. Mirchandani and R. Khanna, eds., *FDDI Technology and Applications*, John Wiley & Sons, Inc., 1993.
17. R. Atkinson, "Default IP MTU for use over ATM AAL5," in *RFC 1626*, May 1994.
18. T. Chen and S. Liu, *ATM Switching System*, Artech House, Inc., 685 Canton St., Norwood, Ma 02062, 1995.
19. ANSI, "AAL5 – A New High Speed Data Transfer AAL," in *ANSI T1S1.5 91-449*, November 1991.
20. A. Forum, *ATM User-Network Interface Specification*, Pretice Hall, Inc, Englewood Cliffs, New Jersey 07632, 1993.

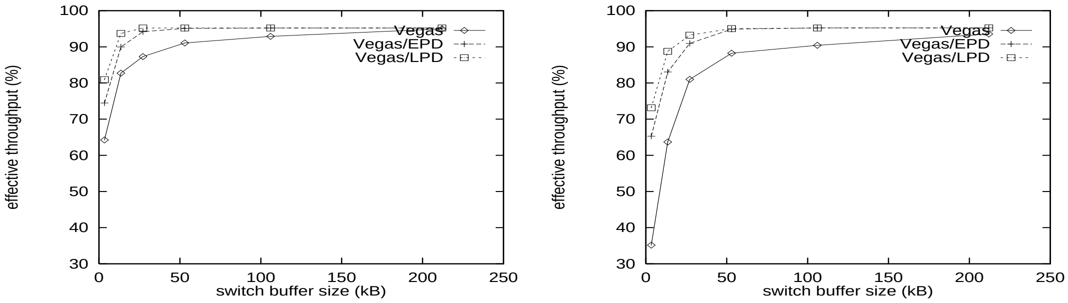

Figure 5. Comparison of the effective throughput of plain Vegas, Vegas/EPD, and Vegas/LPD, over ATM, for 1,500-byte packets (left) and 9,180-byte packets (right) and a 64 kB window size, as a function of switch buffer size

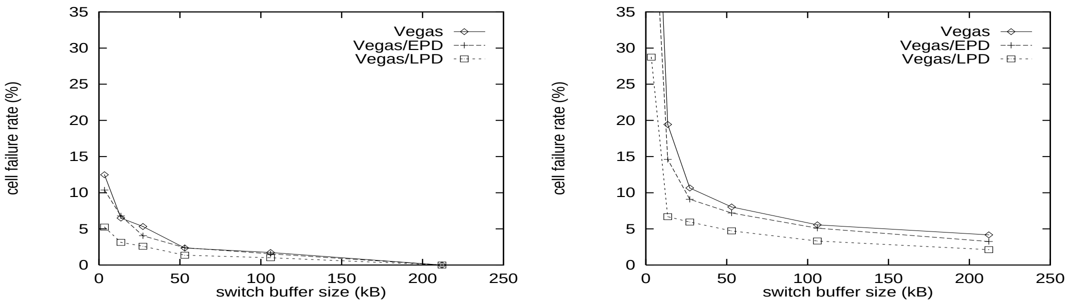

Figure 6. Comparison of cell failure rates for Vegas, Vegas/EPD, end Vegas/LPD, over ATM, for 1,500-byte packets (left) and 9,180-byte packets (right) and a 64 kB window size, as a function of switch buffer size

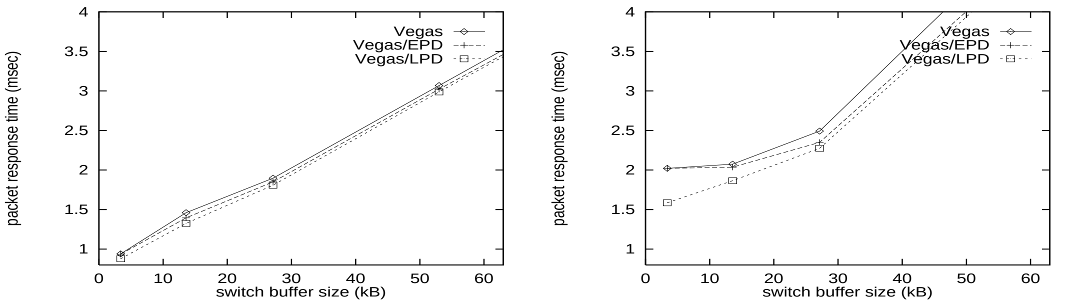

Figure 7. Comparison of packet response times for Vegas, Vegas/EPD, and Vegas/LPD, over ATM, for 1,500-byte packets (left) and 9,180-byte packets (right) and a 64 kB window size, as a function of switch buffer size