Performance of Application-Specific Buffering Schemes for Active Networks

Samrat Bhattacharjee[†] bobby@cc.gatech.edu College of Computing

Martin W. McKinnon b.mckinnon@ieee.org Georgia Tech Research Institute

Georgia Institute of Technology
Atlanta, GA 30332
[†]Corresponding author

Abstract

As the cost of computing power decreases and network traffic patterns become more complex, it becomes worthwhile to consider the benefits of allowing users to specify policies for managing their traffic within the network. Active networking is a new design paradigm in which the network is architected not merely to forward packets, but also to be dynamically programmed in order to support per-user services. Active networks export uniform "meta-level" interfaces that expose network-level resources, policies and mechanisms to its users. In this work, we present a new scheme for managing network buffer space, *aggregate application level buffering*, in which a group of flows of similar types are managed by an active node as a single entity. Thus, resources within the network are provisioned on a "per application" basis rather than a "per user" basis. We justify this scheme by comparing it to previously accepted buffering schemes such as a single shared buffer, per flow queueing, and an extension to per flow queueing which accommodates application specific packet dependencies. Within this work, we justify the benefits of this scheme in terms of required computational processing, information maintained by a node within a network, goodput achieved per application type, and mean delay. Based on the work herein, we show that the impact of an aggregate buffer management scheme can be significant depending on the interdependencies which may exist within an application's traffic stream.

Keywords: Active networking, congestion control, buffer management, active discard

1 Introduction

Packet switched networks are designed and built for efficient forwarding of user data, and traditionally have included only the minimal processing necessary for correct forwarding of packets. However, fueled in large part by the success of the Internet, packet networks continually transport more and different traffic types, many of which have requirements beyond just best-effort forwarding. Additionally, the price-performance ratio of computing power continues to decrease rapidly. As services required of the network increase in complexity and relative processing cost decreases, the recent trend has been to introduce more user-specific[1] processing *inside* the network. In other words, as the cost of computing power decreases and network traffic patterns become more complex, it becomes worthwhile to consider the benefits of allowing users to specify policies for managing their traffic within the network.

Consider traditional congestion control schemes. Generally, network losses are *magnified* at the higher layers as the network discards data without regard to application data units. For instance, policies such as "Early Packet Discard" and "Partial Packet Discard" are currently imposed on all flows traversing a particular node or interface in order to restrict congestion. While these two policies may be sufficient to improve goodput of data-oriented connections, they are not enough to necessarily maintain ideal performance of, for example, video connections. Video connections require a much more complex packet discard scheme in order to maintain acceptable performance, one which accounts for the different types of frames in a video connections and their relationships between one another. Therefore, the use of a congestion control scheme which is *tailored* to the video application would have the advantage of maintaining a more acceptable goodput level (than EPD or PPD) while still restricting congestion. Similarly, recent advances in understanding the self-similar nature of traffic [1, 2, 3] indicate that effective congestion control is likely to require methods which account for longer time periods (e.g., source adaptation and admission control, as opposed to adding buffer space).

One means of implementing this type of behavior which has been significantly addressed in the current literature is *active networking* (e.g., [4, 5, 6]). *Active networking* is a new design paradigm in which the network is architected not merely to forward packets, but also to be *dynamically programmed* in order to support per-user services. Active networks export uniform "meta-level" interfaces that expose network-level resources, policies and mechanisms to its users. Using these uniform interfaces, users can dynamically modify the network's behaviour and "on-the-fly" intro-

[1] In the ensuing discussion, user refers to any entity that requests network layer services and includes (but is not restricted to) traditional transport protocols, and higher-level abstractions such as flows and processes.

duce new capabilities inside the network. Therefore, within active networking's paradigm, the network is no longer viewed as a passive mover of bits, but rather as a more general computation engine: information injected into the network may be modified, stored, or redirected as it is being transported. While there are many architectures currently being investigated, given the basic premise above, it would be possible to adapt an active network architecture to perform functions such as those suggested. In fact, prototype studies have already pushed into caching and congestion control as applications for active networks [7, 8].

We have shown the benefits of application specific congestion control in our previous work [8, 9]. Per application processing within the network provides maximum goodput to end users while providing maximum utilization of network resources. However, the use of per application processing and state within the network may be prohibitively expensive to provision and maintain. In this paper, we extend our previous work to include *aggregate application level buffering*, in which a group of flows of similar types are managed by an active node as a single entity. Thus, resources within the network are provisioned on a "per application" basis rather than a "per user" basis. We justify our schemes by comparing them to previously accepted buffering schemes such as a single shared buffer and per flow queueing. We present simulations and analyses of the four different methods for managing a set of connections traversing a network node. Within this work, we justify the benefits of the schemes in terms of the following measures:

1. Computational processing required for each scheme's implementation
2. State information maintained by the node;
3. Goodput achieved per application type; and,
4. Mean delay for traversing flows.

We begin the paper by describing the models which are used throughout this paper. We then present, in Sections 3 through 5, the arguments and analyses justifying the benefits and trade-offs of each of the traffic management schemes studied within the paper. Finally, we conclude the paper in Section 6.

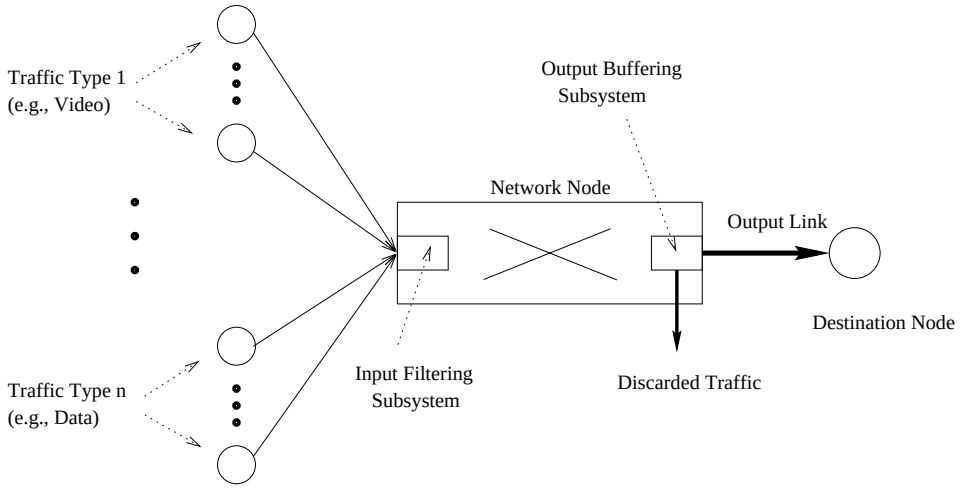

Figure 1: Experimental Topology

2 Model Description

2.1 Topology

Within our models, a significant number of sources operate concurrently and transmit data which is routed through a network node. While the sources' data may be destined for several different outputs, all of data in these experiments is routed over the same output port of an network node, as shown in Figure 1; therefore, for the purposes of these experiments the traffic is considered as having the same single destination. Traffic throughout this model is assumed to use a packet-oriented protocol such as IP.

2.2 Source Models

In these experiments, the traffic sources are classified as either "Video" or "Data" sources[2]. Our model for Video traffic will be based on the MPEG compressed video framing and transmission scheme. Our Video model uses a variable rate video encoding scheme. It uses three types of frames (I, P, and B frames) to encode data using relational and temporal compression. I frames provide periodic updates to the complete frame. P and B frames are compressed representations to changes in a frame; the compression scheme uses both forward and reverse compression. In general, B frames are dependent upon the most immediately preceding I or P frames and P frames

[2] When referring to the specific models of sources used in these experiments, the terms "Video" and "Data" will be capitalized; the generic versions of these terms will be denoted in lower case.

Table 1: Frame Sizes per Session of Video Traffic

Frame Type	Minimum Value	Distribution Type	Mean Value	Maximum Value
I frame	15,000 bytes	Uniform	16,000 bytes	17,000 bytes
P frame	512 bytes	Exponential (truncated)	1,900 bytes	17,000 bytes
B frame	128 bytes	Exponential (truncated)	500 bytes	17,000 bytes

are dependent upon the most immediately preceding I frame [10].

Our Video model provides for the transmission of these frames according to a fixed, repeating structure (termed a *group of pictures* or "GOP"); for these experiments we will use the following frame ordering: I-B-B-P-B-B-P-B-B. The distributions and key values relating to the frame sizes are shown in Table 1; notice that the mean amount of traffic offered by one session of the Video application is approximately 605 kilobytes per second (kBps)[3].

The significance of the structure is that a *dependency map* can be easily formulated to define the interdependencies of the frames upon one another. Using such a map, it is seen that, if the data upon which a frame is dependent is lost, the frame cannot be decoded and loses its value to the executing application. Therefore, a significant goal of this process is to *selectively discard* data such that data which has little value (i.e., upon which a minimum of other data is dependent, e.g., B frames) is discarded, if necessary, followed by data of gradually increasing value.

In order to determine the behavior of Video's highly correlated data and extremely rigid structure in a realistic environment, we also introduce a generic "Data" application. Data packets are all of a fixed (maximum) size, 576 bytes in these experiments, and have no *known* dependency structure. Packets are generated with an exponential interarrival time distribution such that they generate the same amount of offered traffic as the previously described Video application (605 kBps).

2.3 Network Node Models

The network node in Figure 1 may perform buffering and discarding of arriving traffic[4] in any of a number of different ways. The manner in which buffering and discarding of arriving traffic is

[3] The amount of offered traffic per connection is determined based on the NTSC broadcast standard of 30 frames per second (fps). The sizes of the frames were determined based on observations of MPEG encoded video at Georgia Tech and corresponds to data presented in [11], [12], and [13].

[4] The terms *flow*, *connection*, and *traffic* are all used interchangeably within this work.

managed will collectively be referred to as the network node's *buffering paradigm*. The buffering paradigms which are studied within this paper are as follows:

1. **Shared Buffer (SHB)**: All data from all applications is aggregated into a single buffer. If space is not available for an incoming packet, the incoming packet *alone* is discarded.

2. **Per Flow Buffering (PFB)**: Each flow is allocated a dedicated buffer. If space is not available for an incoming packet in its respective buffer, the incoming packet *alone* is discarded.

3. **Per Flow Buffering with Active Discard (PFB/AD)**: Each flow is allocated a dedicated buffer. If space is not available for an incoming packet in its respective buffer, the incoming packet *and all packets dependent upon the incoming packet* are discarded.

4. **Active Buffering and Discard (ABD)**: Each *application type* is allocated a dedicated buffer. If space is not available for an incoming packet in its respective buffer, the incoming packet is discarded along with (potentially) other packets, depending upon the discard scheme implemented.

In the case of Data traffic, there are no dependencies acknowledged; therefore, any discard of data from these flows is always simply individual packets. The dependencies of Video traffic, however, make for a more intricate situation. The basic dependency structure of a single Video flow (the traffic which traverses the single connection) was outlined in Section 2.2. Based on this structure, in the case of using "Per Flow Buffering with Active Discard", if an I frame is discarded, all data from that single flow until the next I frame is discarded (i.e., the entire single GOP); similarly, if a P frame is discarded, all ensuing B frames until the next I or P frame are discarded. However, in the case of using "Active Buffering and Discard", if an I frame is discarded, all data from *all* Video flows which is received is discarded until another I frame is received; equivalently, if a P frame is discarded, all packets from *all* Video flows which are received are discarded until either an I or a P frame are received.

The thrust of this work is to justify and demonstrate that the performance of the different buffering paradigms varies significantly and that advantages can be yielded by using active networking mechanisms to manage buffering at an network node. The performance of several buffering schemes is evaluated in terms of a number of metrics in Table 2. It is seen that, while one buffering paradigm may be extremely effective at managing one aspect of performance, it may be accommodate other aspects poorly. For instance, while the traditional "Shared Buffer" scheme does make very good use of buffer space, it provides very poor quality of service for real-time applications under load. This is exactly the reason that "Per Flow Queueing" schemes were introduced; they,

Table 2: Characteristics of studied buffering paradigms

Paradigm	Computation	State	Goodput	Delay
SHB	Low	Low	Low	High
PFB	Medium	High	Medium	Medium
PFB/AD	High	High	High	Low
ABD	Variable	Low	High	Low

however, suffer from as a much less effective buffer management scheme. It is noted in Table 2 that the computation requirement for the ABD buffering paradigm depends upon several factors; the details of these factors are highlighted out in Section 3.

3 Computational Processing Requirements

Within this section, we address the issues relating to the amount of data required to operate a node using one of the previously described buffering paradigms.

In the node in Figure 1, we consider three functions which may significantly consume the processing resources of a routing node as shown in our topology: packet sorting at the entrance to the output port, packet discard at the entrance to the output port, and output buffer scheduling. For the purposes of this study, we will only consider the case in which a packet entering the node is only intended for a single destination.

In the SHB case, the computational requirements are minimal; since there is only one buffer managed at the output port with none of the features of the other schemes, the computational process should be able to be performed in constant time. In the PFB paradigm, the incoming packet need only determine the output buffer (based on its flow identifier or equivalent mechanism) to which it should be assigned. Since this determination may be either linear or sub-linear in complexity, the time for performing this operation is listed in Table 3 as $t_{\text{pf}-\text{sort}}$, where this value is given in (1).[5]

$$t_{\text{pf}-\text{sort}} = O(f^{(0)}(\#\text{ flows})) \tag{1}$$

The PFB/AD case adds an additional level of complexity to the PFB paradigm. The input

[5] The family of functions denoted by $f^{(\cdot)}(\cdot)$ are arbitrary and unspecified functions which are dependent on the discard algorithm and other operations implemented for the application type and instance.

Table 3: Computational characteristics of studied buffering paradigms

Paradigm	Computational Requirements
SHB	$O(1)$
PFB	$t_{\text{pf-sort}} + t_{\text{pf-sched}}$
PFB/AD	$t_{\text{pf-sort}} + t_{\text{app-discard}} + t_{\text{pf-sched}}$
ABD	$t_{\text{app-sort}} + t_{\text{app-discard}} + t_{\text{pf-sched}}$

packet filtering operation is equivalent to that in the PFB case; however, the packet discard process will require additional processing, as shown in (2). This expression denotes a complexity which is a function of the discard algorithm for the application of every defined buffer and the buffer itself. Notice that the term "application(buffer)" is intended to return the application corresponding to the traffic which traverses the given buffer.

$$t_{\text{app-discard}} = O(\sum_{\forall \text{buffers}} f^{(1)}_{\text{discard}}(\text{application}(\text{buffer}), \text{buffer})) \tag{2}$$

For the ABD paradigm, packet sorting is performed on a "per application" basis. Therefore, the complexity of this operation is reflected in (3).

$$t_{\text{app-sort}} = O(f^{(2)}(\#\text{ applications})) \tag{3}$$

Finally, for nodes which support a buffer scheduling process, scheduling will require an amount of processing corresponding to $t_{\text{pf-sched}}$, as shown in (4). This type of process will vary from a "weighted round robin" scheme which actually performs a scheduling process to an arbitrary function which may be provided by the user.

$$t_{\text{pf-sched}} = O(f^{(3)}(\#\text{ buffers})) \tag{4}$$

This reasoning leads to the information given in Table 3.

4 State Information Requirements

Within this section, we address the issues relating to the amount of data required to operate a node using one of the previously described buffering paradigms. First, we argue that there exists a

Table 4: State characteristics of studied buffering paradigms

Paradigm	# Buffers	State Requirements
SHB	1	$O(1)$
PFB	# flows	s_{buffer}
PFB/AD	# flows	$s_{\text{buffer}} + s_{\text{discard}}$
ABD	# applications ($\ll$ # flows)	$s_{\text{buffer}} + s_{\text{discard}}$

minimal amount of data required for the operation of the node; therefore, this amount of memory is $O(1)$ and cannot be significantly affected regardless of the buffering paradigm implemented. This data would include information such as the number of separate buffers which are managed, the number of packets which are enqueued, the data which would be included as part of a standard MIB (or similar statistic database), etc.

Second, we argue that there is a separate set of information which is maintained on a "per buffer" basis. This information would include the number of packets enqueued to the buffer, any sorting or selection criteria for the buffer, and other similar information. We expect the same information to be stored on a "per buffer" basis regardless of the buffering paradigm; therefore, the cost of maintaining this information within memory is equal to s_{buffer} as defined in (5).

$$s_{\text{buffer}} = O(\#\text{ buffers}) \tag{5}$$

Finally, consider, for example, the currently discussed Video application type. In processing the active discard algorithm for Video traffic, state information related to what types of packets are being accepted/discarded must be maintained (in our implementation, 2 bits were necessary to indicate if the most recent I or P frame had been discarded). Extrapolating this notion to other applications, we contend that the state information required for this functionality is encountered on a "per buffer" basis and may be expressed as s_{discard}, as defined in (6)[6].

$$s_{\text{discard}} = \sum_{\forall \text{buffers}} g_{\text{discard}}(\text{application}(\text{buffer}), \text{buffer}) \tag{6}$$

Based on this analysis, the information given in Table 4 is justified.

[6] $g_{\text{discard}}(\cdot)$ is an arbitrary function with the same constraints as those previously mentioned applying to $f^{(\cdot)}(\cdot)$.

5 Goodput, Buffer Space, and Delay Requirements

Given the complex structure of Video traffic, it would be extremely difficult to provide an intuitive explanation for the flow's buffer space requirements or delay. Therefore, a simulation was used to determine and evaluate these requirements. In order to assure some sense of comparability between situations, we used application-level quality of service as a criteria for determining when two scenarios provided "equivalent" performance. For every case, 20 replications of each scenario were run and aggregated in order to ensure that the results are representative; each replication had a duration of 35 seconds with the first 5 seconds being considered a transient period (during which statistics were not accumulated). Also, during each of the replications, the exact time at which the Video connections commenced was randomly varied between the starting instant of the simulation (at time "0.0") and exactly one-third of a second later. This "staggering" of the sources was performed to prevent the regular structures of the Video streams from causing unrealistic buffer sizing (as a result of, e.g., several I frames arriving to the node sequentially); the implications of not considering this point are shown later in this work.

The buffer service policy generally used within these experiments is a "weighted round robin" scheme; the amount of service that a buffer receives is weighted according to the percentage of traffic offered to the node which is subsequently routed to the given buffer. The results which are presented in this work without annotation use this "weighted round robin" service policy. However, during the course of our experiments, some interesting behavior was noticed when the service weights were set equal to one another. As warranted, we present the results of implementing this "equal weights" policy and denoted the results by suffixing the relevant buffering paradigm by "WEQ".

5.1 Comparing Aggregate Goodput to a Given Level

Goodput is a term which has evolved in recent years to capture the notion of the fraction of data that contributes to the quality of service of a transmission. Within these experiments, we define "goodput" for a single connection as the percentage of traffic sent to a destination which can be used during an application level decoding process (i.e., for which all data upon which a received portion of data is dependent are also received). We then average and determine the confidence interval for several connections' measures; these values are used in the following manner to establish if a goodput level has been met.

The first criteria for meeting a specific desired goodput level is that the lower end of the confidence interval found in the previous set is at least a given level; in our experiments this level

varied from 70% to 90% (i.e., fair to very good quality or, equivalently, 21 to 27 fps). The second criteria for determining convergence is that the minimum goodput *of any single connection* is at least 90% of the level used in the previous test (i.e., 63% to 81%); this criteria was imposed to prevent, for example, one connection achieving near-ideal performance at the expense of another connection's inordinately poor performance. Only if both criteria were met, the aggregate goodput level was deemed to have been met.

5.2 Determining Buffer Space Requirements Given Fixed Goodput, Utilization, and Link Size

The first set of experiments were used to determine the minimal buffer space required to satisfy a number of connections while still meeting a "minimal" required quality of service for the end-users. The buffer size required to accommodate the given goodput is found only when both of the criteria given in the previous subsection are met.

A 20 Mbps link was used in these experiments. The range of values for the total number of connections used for an experiment was 26, 30, and 32; this range was chosen in order to determine the impact of loading on the effectiveness of the buffering schemes. Notice that 26, 30, and 32 offered connections correspond approximately to a utilities of 80%, 90%, and 95% of the 20 Mbps link. The connections were a mixture of Video and Data; the number of Video connections were varied from 2 to the maximum number of connections used in a particular experiment (with the remaining connections being used as Data connections).

We present in Figures 2 and 4 the resulting buffer space required per Video connection operating over the 20 Mbps link with 90% utilization; the figures corresponding to 80% and 95% are not presented for the purpose of brevity and since their results were similar. In these figures, in the cases of the SHB and ABD buffering paradigms, the buffer space per connection is computed based on the size of the entire buffer. Since our simulation converged based on actual buffer sizes, confidence intervals on the buffer sizes for these buffering paradigms are not presented in these figures. Instead, the (extremely tight) confidence intervals for only these cases are shown in the plots of total buffer space allocated to Video in Figures 3 and 5.

Intuitively, one would expect that, as the offered traffic (and, correspondingly, the output link's utilization) is increased *or* as the required goodput is increased, the required *total* buffer space would increase. Further, as the percentage of offered traffic which is Video traffic (with its intricate interdependencies) increases, we see that the buffer space required to achieve the given goodput level increases. This behavior is exactly what is seen in Figures 2 and 3. The PFB

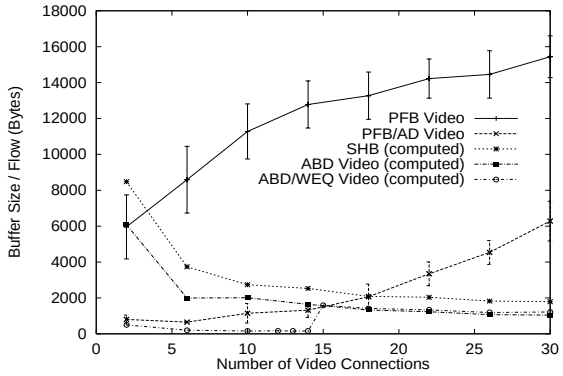

Figure 2: Per Video Connection Buffer Sizes for 90% Utilization and 90% Goodput (Total Connections = 30)

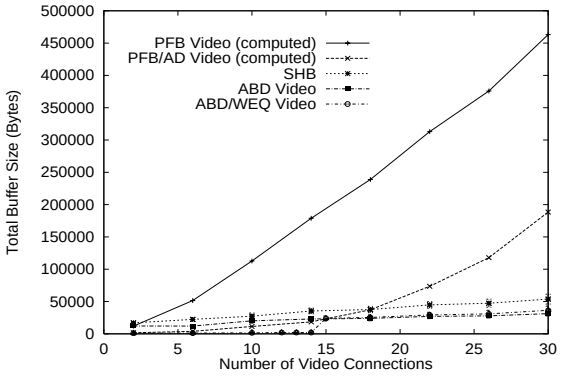

Figure 3: Total Video Connection Buffer Space for 90% Utilization and 90% Goodput (Total Connections = 30)

and PFB/AD buffering paradigms increase (approximately) linearly as they is not able to benefit from any statistical multiplexing. In Figure 2 a dramatic jump is seen when the number of Video connections is increased from 14 to 15 (when the percentage of traffic offered by Video connections is 50% of capacity) under the ABD/WEQ buffering paradigm; this phenomena is due to the equal weightings of the two (Video and Data) buffers in the buffer servicing scheme. It should be noted that the SHB paradigm performed well when the majority of incoming traffic was Data (without any dependencies imposed by the traffic stream), but its requirements increased dramatically with increasing amounts of Video. The remaining paradigms were able to take increasing advantage of the multiplexing with several other connections. Similar behavior is seen in Figure 4.

The only aberration to these explanations is seen in Figure 4, when 26 Video connections are simulated. At this point, the buffer size required per Video connection actually decreases from its value at 22 Video connections. However, based on the shown confidence interval, it appears that this value is a combination of the high utilization of the output link given these connection as well as statistical variation.

The most significant point to note in these figures, though, is that the ABD buffering paradigm is able to provide the benefits of statistically multiplexing a significant number of connections within a relatively small buffer space while controlling each connection's goodput levels.

The mean packet delay for the previous experiments are shown in Figures 6 and 7. Delay may not be a significant factor in the Data connections used within these experiments, but it definitely is significant for Video connections; frames must be received by a destination within 33 milliseconds

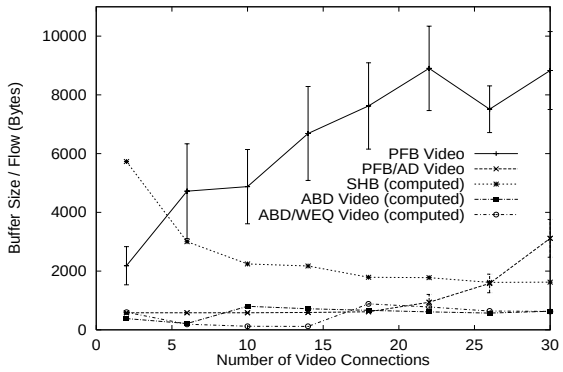

Figure 4: Per Video Connection Buffer Space for 90% Utilization and 70% Goodput (Total Connections = 30)

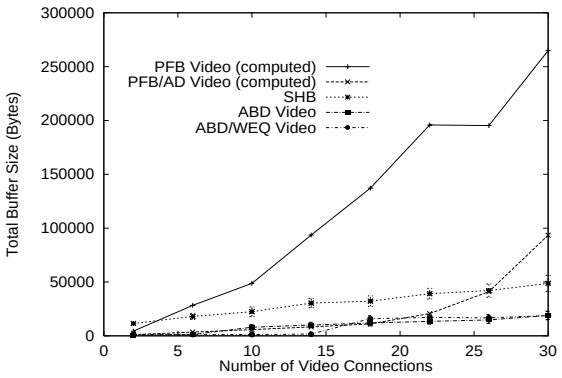

Figure 5: Total Video Connection Buffer Sizes for 90% Utilization and 70% Goodput (Total Connections = 30)

(mS). In these plots, the mean delays do not reach unacceptable levels in any of our experiments. Also, we generally see consistantly increasing mean delays as the number of Video connections increase. In the case of the SHB bufering paradigm, this phenomena is due to the fact that, as the number of Video connections increases, more of the traffic is developing a deterministic pattern and less of the traffic is arriving with exponential interarrival times. In the case of the PFB and PFB/AD cases, the sets of buffers are essentially polled and service irrespective of the arrival times of the packets; in comparing these two curves, however, the impact of the Active Discard scheme should be noted.

Most dramatically, in both Figures 6 and 7, the ABD/WEQ buffering paradigm[7] shows a dramatic increase in packet delay about the point at which 15 out of 30 connections are Video traffic. This point is due to the observation that, in this paradigm, each of the two queues will receive either (a) enough service to keep the buffer's occupancy relatively low (when the buffer does not receive enough traffic to consume 50% of the service time), or (b) the 50% to which the buffer is entitled *plus* any unused service time made available by its counterpart. In the first case, a packet's delay will be relatively small (i.e., on the order of a single packet transmission time since the buffer's occupancy will be kept small and the assignment of service periods is performed on a "per packet" basis). In the latter case, the input buffer will grow due to the fact that the server will periodically not be available to service the heavily loaded buffer (in order to service the more lightly loaded buffer).

[7] Recall that, in the ABD/WEQ buffering paradigm, the buffers corresponding to Video and Data traffic each have priority over exactly 50% of the service time

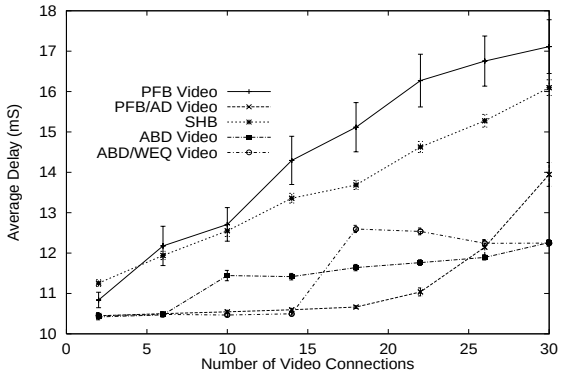

Figure 6: Mean Packet Delay for 90% Utilization and 70% Goodput (Total Connections = 30)

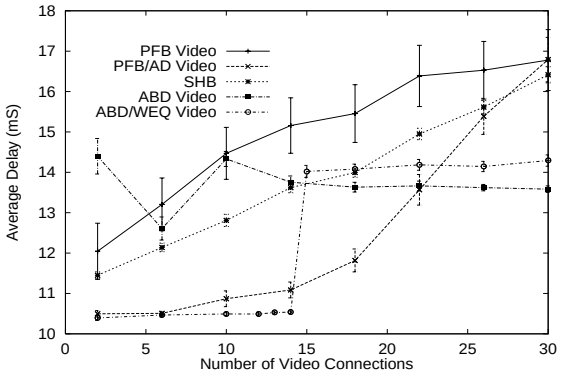

Figure 7: Mean Packet Delay for 90% Utilization and 90% Goodput (Total Connections = 30)

Finally, though, it is seen that, while using the ABD buffering paradigm, the mean packet delay experienced a certain amount of volatility at lower numbers of connections, but, as the overall traffic became dominated by Video, the mean delay stabilized in both figures. This stabilization can be attributed to the combined effects of the statistical multiplexing of the Video connections and the Active Discard scheme. As a result, the ABD scheme was able to not only control the goodput of the Video connections (thereby maintaining the quality of the connections), but also controlled the delay of the connections.

5.3 Determining Buffer Space Requirements Given Fixed Goodput and Utilization

In order to determine the impact of the link's capacity being fixed and fairly large in comparison to a connection's size, we next fixed the required goodput level and the utilization of a *variably sized* output link. The link's size was varied with the number of connections offered to the node. Then, as in the previous experiments, the minimal buffer size per connection was determined. Since the PFB and PFB/AD buffering paradigms required significantly more buffer space to implement in the previous experiments, and the traffic traversing these paradigms incurred substantially larger mean delays, these paradigms have been excluded from this experiment.

These experiments were performed for 80%, 90%, and 95% utility of links of sizes varying from approximately T1 to OC3 rates. Goodput was similarly varied from 70% to 90%. Presented in Figures 8 and 9 are representative results obtained at 90% utilization and 70% and 90% goodput,

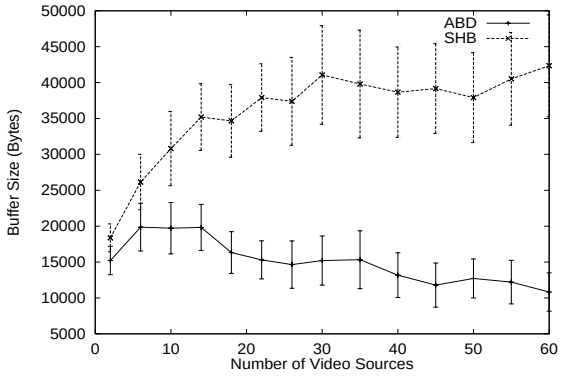

Figure 8: 90% Link Utilization and 70% Goodput

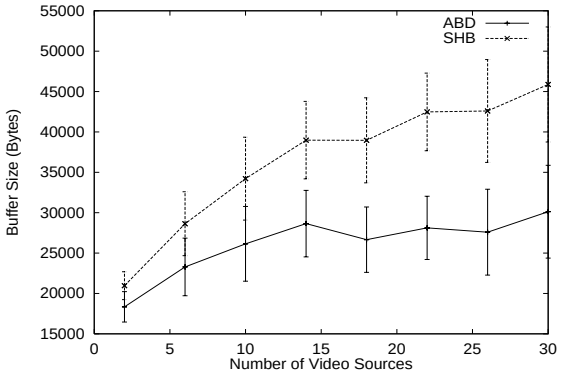

Figure 9: 90% Link Utilization and 90% Goodput

respectively.

As one would expect, the ABD buffering paradigm performed significantly better than the SHB paradigm. In fact, the ABD buffering paradigm is able to manage its buffer space significantly more efficiently than the SHB buffer paradigm as the link rates and number of sources are varied. It is seen in both figures that the SHB paradigm's buffer requirements essentially increased consistantly as the offered load increased. While it appears that the ABD paradigm's buffer requirement is essentially stagnant at the higher values of offered load in Figure 9, the same paradigm requires less space at higher values in Figure 8. This point would seem to be due to a combination of the effects of multiplexing the increasing number of Video connections and the Active Discard algorithm's ability to optimally take advantage of the space available while still meeting the required goodput level.

5.4 Determining Goodput Given Buffer Space

In the next set of experiments, the number of connections offered to the network node was fixed at 10 Data and 20 Video connections and the output link size was fixed at 20 Mbps. The buffer size per connection was gradually increased and the aggregate goodput for each application type were recorded in Figures 10 and 11. Recall that the goodput for Data connections is simply the fraction of traffic which is not discarded in transit.

By comparing Figures 10 and 11, it is seen that, as expected, Data flows overall had higher goodput levels than Video flows given equal buffer space. Similarly, both the PFB and PFB/AD

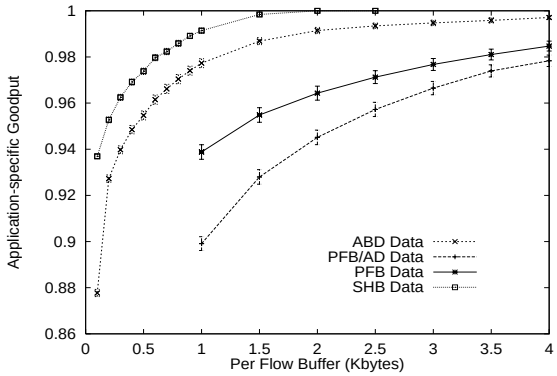

Figure 10: Data Application Specific Goodput for 20 Video & 10 Data Connections Traversing a 20 Mbps Link

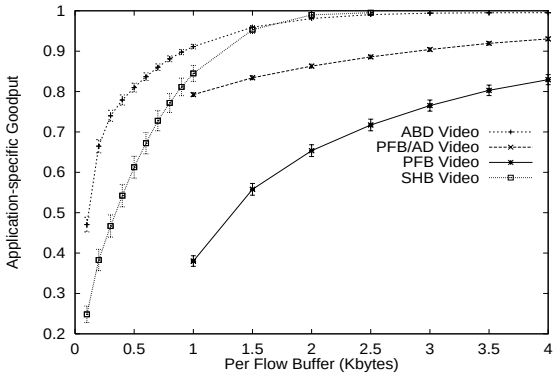

Figure 11: Video Application Specific Goodput for 20 Video & 10 Data Connections Traversing a 20 Mbps Link

paradigms performed significantly poorer than SHB and ABD paradigms in cases of both traffic types. ABD was obviously more beneficial to the Video connections than the Data connections.

However, by comparing Figures 12 and 13, the benefits of the different buffering paradigms can be seen under changing offered loads. The goodput of both types of traffic degenerated under load with all of the paradigms. It should be noted that the degeneration was fairly uniform for the Data traffic, with it benefitting the least from the PFB/AD paradigm. This point is reasonable since, as the number of maintained connections increases under PFB/AD, more time would elapse while serving individual buffers between service attempts for a given flow; therefore, the queues would be most likely to build within a given period.

However, to see why the Data traffic benefitted the most from the ABD paradigm, even more than the SHB paradigm, we must observe the degeneration of the Video traffic curves in Figure 13. Video was much more sensitive to the increase in offered load due to the interdependencies between its frames. Recall that, if an I frame is lost, all ensuing frames preceding another I frame lose their value; similarly if a P frame is lost, all ensuing frames preceding another P or I frame lose their value. The ABD and PFB/AD paradigms was able to discard frames which were no longer valuable (due to a preceding frame already being discarded) and maintain a higher goodput level under load. The performance of the PFB and SHB paradigms were significantly worse.

In the previous sections, it was shown that the ABD buffering paradigm is able to effectively multiplex connections within an allocated buffer space. This set of experiments illustrates that, while the ABD paradigm is able to utilize the buffers in an effective manner, it also provides higher

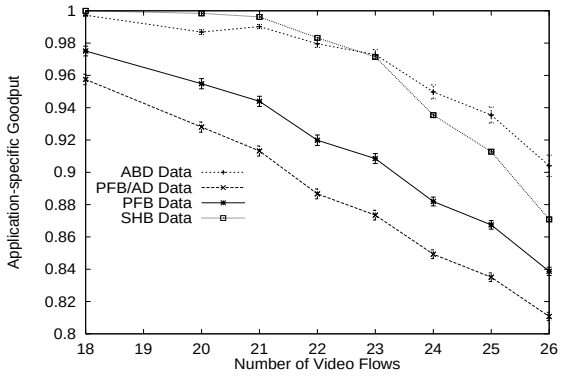

Figure 12: Data Application Specific Goodput for Traffic Traversing a 20 Mbps Link (Number of Data Connections = $\lfloor \frac{\text{Number of Video Connections}}{2} \rfloor$; 1.5kB Buffer Space per Connection Available)

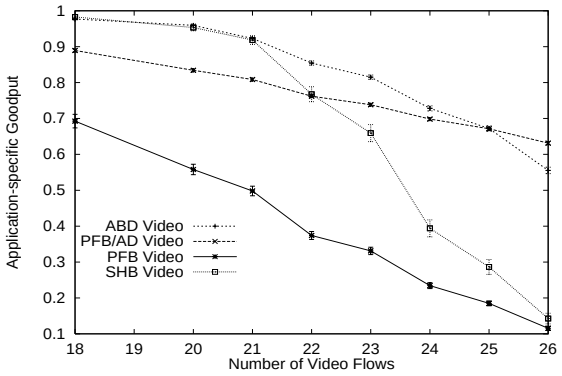

Figure 13: Video Application Specific Goodput for Traffic Traversing a 20 Mbps Link (Number of Data Connections = $\lfloor \frac{\text{Number of Video Connections}}{2} \rfloor$; 1.5kB Buffer Space per Connection Available)

goodput while controlling the buffered packets' mean delay.

5.5 The Impact of Non-Staggered Traffic Sources

As previously mentioned, the repeating nature of Video traffic can have a significant impact on buffer sizing (see [14]). In order to determine the impact of the source staggering scheme, we compared the previously described situation to one in which the 30 video connections simultaneously at 0.0. The buffer size required per connection in order to yield 90% goodput are shown.

Both Figures 14 and 15 show approximately linear increase with the number of Video sources offered. This point is due to the fact that the arrival of the I frames, the largest frames in the Video streams, between all sources are synchronized. Therefore, the buffer space will be constrained by the sizes of the I frames (which cannot be discarded and still yield a non-zero goodput level) which are buffered.

The point of this experiment is that the exact synchronization of Video sources could significantly impact the buffer requirement or quality of provided service. However, in an actual system, it is not expected that a *significant* number of sources (in comparison to the total number of connections supported) will be synchronized such that I frames arrive virtually concurrently.

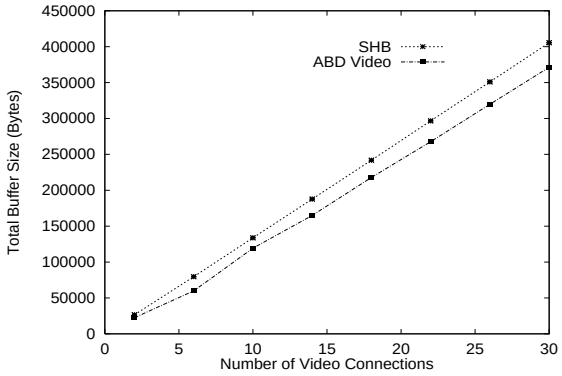

Figure 14: Total Buffer Size Required for Perfectly Synchronized Video Sources with 90% Goodput

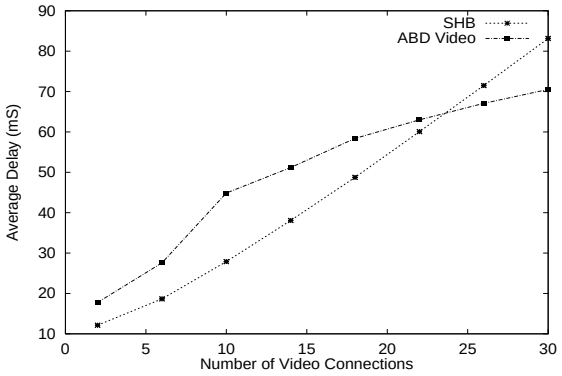

Figure 15: Mean Packet Delay Incurred by Perfectly Synchronized Video Sources with 90% Goodput

6 Conclusions and Future Work

This paper has shown the value of providing *aggregate application level buffering*. This scheme is made possible by using active networking concepts in order to allow a node within the interior of a network to manage groups of similar flows as single entities. This type of scheme has two significant aspects. One is the benefits obtained by statistically multiplexing similar flows within a buffer, the size of which may be dynamically managed as new connections are accepted. The second is the advantage realized in the aggregate flow's goodput which is achieved by expending a minor amount of computational resources to determine the value of data before buffering it.

We expect this type of scheme to be a significant advantage to active networking. Additionally, the reduced use of network resources should make for a more scalable implementation of active nodes. One significant area of future work in this area would be determining criteria for gauging the "similarity" of application types (e.g., determining if 24 fps video is close enough to 30 fps video to be managed according to the same buffering paradigm). We expect that area, as well as that of designing effective discard algorithms, will be realized within the near future.

References

[1] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of Ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, pages 1–15, February

1994.

[2] N. Likhanov, B. Tsybakov, and N. Georganas. Analysis of an ATM buffer with self-similar (fractal) input traffic. In *IEEE Infocom '95*, 1995.

[3] 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 *ACM Sigcomm '95*, pages 100–113, 1995.

[4] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1), 1997.

[5] D. L. Tennenhouse and D. J. Wetherall. Towards an active network architecture. *Computer Communication Review*, 26(2):5–18, April 1996.

[6] D. Wetherall, J. Guttag, and D. L. Tennenhouse. ANTS: A toolkit for building and dynamically deploying network protocols. In *IEEE OPENARCH'98*, San Francisco, CA, April 1998.

[7] Samrat Bhattacharjee, Kenneth L. Calvert, and Ellen W. Zegura. Self-organizing wide-area network caches. In *IEEE Infocom'98*, 1998.

[8] Samrat Bhattacharjee, Kenneth. L. Calvert, and Ellen W. Zegura. An Architecture for Active Networking. In *Proceedings of High Performance Networking 97*, 1997.

[9] Samrat Bhattacharjee, Ken Calvert, and Ellen Zegura. Active networking and the end-to-end argument. In *Proceedings of ICNP'97*, 1997.

[10] MPEG-2 Systems Committee. MPEG-2 Systems Working Draft. ISO/IEC/JTC1/SC29/WG-11-N0501, July 1993.

[11] Rahul Garg. Characterization of video traffic. Technical Report TR-95-007, International Computer Science Institute, Berkeley, CA, 1995. available at ftp://ftp.icsi.Berkeley.edu/pub/techreports/1995/tr-95-007.ps.gz.

[12] Steven L. Blake, Sarah A. Rajala, and Fengmin Gong. Efficient techniques for two-layer coding of video sequences. Technical report, North Carolina State University, 1993.

[13] J. Enssle. Modelling and statistical multiplexing of VBR MPEG compressed video in ATM networks. In *Proceedings of the 4th Open Workshop on High Speed Networks, IND, Universitt Stuttgart*, page 10, Brest, France, 1994.

[14] Maurizio Casoni and Jonathan S. Turner. On the performance of early packet discard. *IEEE Journal on Selected Areas of Communication*, 15(5), June 1997.