Appears in *Proceedings of Infocom*, March 1998, Copyright held by IEEE.

Dynamic Flow Switching
A New Communication Service for ATM Networks

Qiyong Bian, Kohei Shiomoto, Jonathan Turner
{bian,shiomoto,jst}@cs.wustl.edu
Washington University

ABSTRACT

This paper presents a new communication service for ATM networks that provides one-way, adjustable rate, on-demand communication channels. The proposed dynamic flow service is designed to operate within a *multi-service cell switched network* that supports both conventional switched virtual circuits and IP packet routing and is designed to complement those services. It is particularly well suited to applications that transmit substantial amounts of data (a few tens of kilobytes or more) in fairly short time periods (up to a few seconds). Much of the current world-wide web traffic falls within this domain. Like IP packet networks, the new service permits the transmission of data without prior end-to-end connection establishment. It uses a variant of ATM resource management cells to carry dynamic flow setup information along with the data, including an IP destination address and burst transmission rate. Switches along the path select next an outgoing link dynamically using the burst rate to guide the routing decision. The dynamic flow setup protocol is based on soft-state control and is designed to facilitate the use of buffers that are shared across all ports of a switch, minimizing per port memory costs.

1. Introduction

The last decade has seen a rapid explosion in the growth of the World Wide Web and related applications. These services have converted the Internet from a tool for technical sophisticates to a mass market phenomenon. One consequence of the growth of the web has been to create a rapid increase in the number of interactive communication sessions that last for very short periods of time, less than a second in many cases. This is a direct consequence of the linked hypertext structure of information on the web, which allows a document to reference documents on other computers, down the hall or on the other side of the world. Often these remote references result in a single page being transferred, leading to a high ratio of control overhead to data transferred. ATM networks are not currently well engineered to handle large numbers of short-lived sessions. The overhead associated with establishing a virtual circuit in ATM networks is high enough that a traffic mix which includes a large proportion of such short lived sessions will be unavoidably expensive, in both bandwidth usage and control load. TCP/IP is also less than ideal for this class of traffic. The TCP setup overhead is excessive for short sessions, and IP routers must perform a lot of redundant work to forward the many packets that may be present in a single transfer.

Another problem with the use of current methods for handling World Wide Web traffic is that resource reservations are difficult to make on a session basis. While it may be reasonable to select a maximum data transfer rate for a web session, there is little basis for selecting an average rate for such a session. Even if one had good statistical data for web traffic, the average data rate over a population of sessions is not that helpful in predicting what will happen during the course of an individual session. The variations are simply too great, and attempting to force individual users to conform to population averages, as ATM usage parameter controls (and similar mechanisms being introduced into IP) seek to do, is counter-productive. The use of rate-based congestion control mechanisms [7] can be helpful in this regard, but these mechanisms are only effective for data bursts that have a duration that is many times the round-trip delay associated with a given session. For web traffic, this condition frequently doesn’t hold. This is an even greater problem for the adaptive windowing methods used in TCP that have even longer reaction times.

This paper proposes a new communication service that can efficiently handle short-lived sessions with minimal control overhead and little or no wasted bandwidth. This service, called *dynamic flow establishment* or *dynaflow* for short, allows data transfer without advance virtual circuit establishment. Dynamic flows provide a best effort sort of service, similar in philosophy to IP packet switching, but suitable for efficiently transferring larger volumes of data and enabling more effective allocation of network bandwidth than is possible with datagrams.

To provide effective support of short-lived sessions, the dynaflow service allows users to send data without advance end-to-end virtual circuit establishment. This type of service is sometimes referred to as “send-and-pray” since there can be no guarantee that the resources needed for a given data transfer will be available to it. We feel that such guarantees are inherently in conflict with the nature of short-lived sessions and hence must be sacrificed if we are to serve such sessions more effectively. However, while we can’t provide guarantees, we can improve

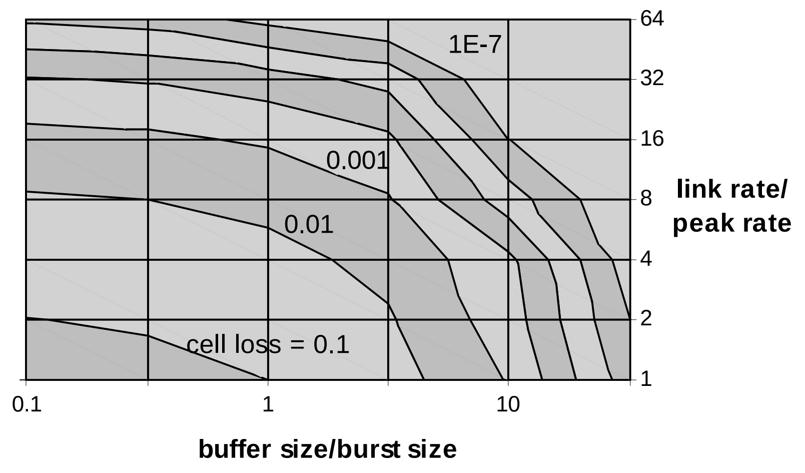

Figure 1. Contour Plot for Cell Loss Rate
(offered load =0.6, peak-to-average ratio=10)

the probability of successful transmission, converting send-and-pray to “send-while-fixing-the-odds.”

The dynaflow concept is designed to take advantage of the fundamental behavior of queuing systems subjected to bursty traffic, as typified by the contour plot showed in Figure 1. This plot shows cell loss rate for a simulated queue with on-off bursty traffic having a peak-to-average ratio of 10:1, exponentially distributed burst lengths and imposing an offered load of 60% of the link’s capacity. The two axes give the ratio of the link rate to the peak rate of the individual virtual circuits and the ratio of the buffer size to the burst length. (For queues that use an effective frame-level discarding technique, such as early packet discard, the packet loss rate equals the cell loss rate.) These two parameters are the primary tools available for improving queuing performance. Note that while increasing either ratio by itself has a strong impact on the cell loss rate, in combination they are extremely powerful and can drive cell loss rates to infinitesimally small values. Similar results are observed for heavy-tailed distributions, such as the Pareto distribution. The chief difference is that the contours intersect the horizontal axis further to the right. There is no significant change to the intersections of the contours with the vertical axis.

There are two ways to maintain a high ratio of link rates to virtual circuit peak rates. First, use the highest speed links feasible in the network, and second keep the virtual circuit peak rate as low as possible, while still giving satisfactory performance. For interactive data applications, the primary criterion for satisfactory performance is providing fast response to user requests. The typical performance target is about one second and there is usually little value in reducing response time to much under one second. Knowing the amount of data to be transferred, the sender (e.g. a web server) can easily calculate the data rate at which data must be transferred to achieve a transfer time of say 0.5 seconds, leaving another 0.5 seconds for other components of the response time. By sending data at this rate, the sender minimizes the probability of congestion-induced data loss and the subsequent need for retransmission. In a network with backbone links operating at 2.4 Gb/s, a link-to-peak ratio of 64:1 still allows data transfers of 37.5 Mb/s, fast enough to transfer an uncoded full-screen 24 bit image in under one second. The vast majority of web transfers are much smaller than this and can be sent in a fraction of a second at data rates of .1 to 10 Mb/s.

However, we can’t always rely exclusively on a high ratio of link rate to virtual circuit peak rate. Some applications do require high rates, not all networks links operate at gigabit rates and links may experience high average loads for extended periods of time. Thus, it’s also important to have large amounts of storage available for buffering bursty data. Unfortunately, data storage can be costly in high performance networks and since memory provided to accommodate bursty data is not in use most of the time, there are powerful motivations to minimize the amount of storage that must be built into switches. Fortunately, this conflict can be resolved effectively by sharing the memory required among all the links in a large switch. Using this approach, it becomes possible to have ample storage available to all links in a system, while amortizing the cost of that storage across the links.

Dynamic flow switching can be viewed as a logical evolution of a class of fast reservation protocols for ATM [1,6,8,10]. These papers discuss burst-level bandwidth reservation on previously established virtual circuits. In some cases, the protocols require end-to-end reservation, while others operate in a contention mode with reservations made on-the-fly. Because the bandwidth reservation is made over established virtual circuits, there is no opportunity to dynamically select the route on a burst basis, as advocated here. Suzuki and Tobagi [9] show that this lack of routing flexibility can limit statistical multiplexing efficiency when the peak transmission rate is a substantial fraction of the link rate. They propose setting up

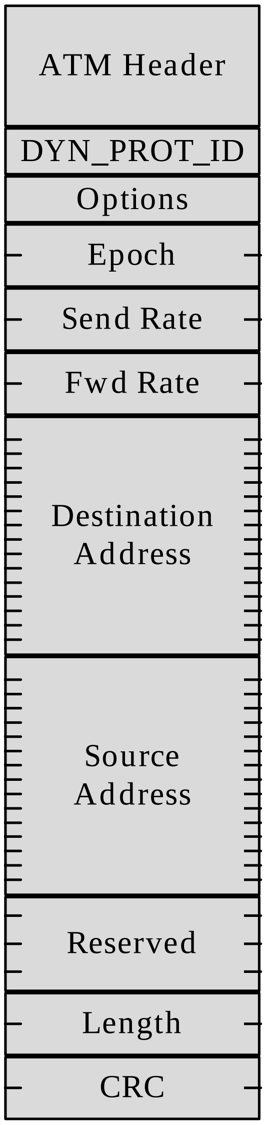

Figure 2. Dynaflow Setup Cell Format

multiple virtual circuits and reserving bandwidth prior to data transmission by either sequentially or in parallel, sending reservation requests along the different paths. Cidon, Rom and Shavitt [4] also make the case for routing flexibility and in [3] describe methods for obtaining such routing flexibility in one-way reservation algorithms, using multiple established virtual circuits. The dynaflow service described here, eliminates the requirement for prior establishment of a fixed number of paths, eliminating the associated control overhead and providing more flexible route selection.

2. Dynaflow Communication Service

The dynaflow communication service provides dynamic establishment of one-way, adjustable rate channels. To use the dynaflow service, a sending host selects an unused virtual circuit identifier on its access link and begins transmitting its data, preceded by a *dynaflow setup cell*, encoded with an ATM payload type of 110_2. (This is the payload type used by ATM resource management cells. It also can serve as a general mechanism for defining new communication services.) The setup cell specifies the destination host and the rate at which data is being transferred. Switches in the network use this information to select an outgoing link, allocate bandwidth to the flow and if necessary, to react to congestion. During data transmission, setup cells are sent periodically to guard against loss of the original setup cell, and at the end of the transmission, an end cell is sent, releasing resources previously reserved. Release of resources can also be triggered by expiration of a time-out, so a lost end cell cannot leave resources allocated indefinitely.

The format of the dynaflow setup cell is shown in Figure 2. The first byte of the payload is the *dynaflow protocol identifier* which distinguishes dynaflow cells from other cells with a payload type of 110_2. The options field is used to specify what is to be done, if when a burst arrives, none of the outgoing links that could be used to reach the destination have sufficient bandwidth available to handle the new flow. The options include the following

- *Discard burst* discards the entire transmission.
- *Discard until next setup* discards arriving cells until the next setup cell is received and then retries.
- *Buffer until required bandwidth available* directs a congested switch to buffer the data until there is an appropriate outgoing link with sufficient unused bandwidth.
- *Best effort forwarding* directs a congested switch to forward the data at whatever rate it can, buffering excess data as need be.
- *Re-routing allowed* permits a switch to route a burst to a different outgoing link in the middle of the burst if re-routing would allow data to be sent at the requested rate.

The sending host increments the *Epoch* field in the setup cell whenever the value of one or more fields in the cell has changed from previous transmissions. This facilitates hardware filtering of setup cells in switches so that the control processors need only handle setup cells that reflect new information. The *Sending Rate* field specifies the rate at which the sending host is transmitting data. If the two bytes of the field are *x* and *y*, the rate in bits per second is $x \cdot 2^y$. The *Forwarding Rate* field specifies the rate at which an intermediate switch is forwarding a given data stream and may differ from the original sending rate as a result of congestion. The source and destination addresses are specified using the IPv6 address format. The *Reserved* field is reserved for the use of switches along the path for internal control purposes, but has no significance outside a switch. The length field gives the length of a burst in bytes using the two byte floating point format mentioned above, with 0 used to indicate that the burst has indeterminate length.

3. Implementation of the Dynaflow Service

Figure 3 shows an ATM switch that implements the dynaflow service. It is based on the Washington University Gigabit Switch [2,12], which uses a highly scaleable architecture and supports link speeds of up to 2.4 Gb/s. The *Input Port Processors* (IPP) for any input link supporting the dynaflow service must be modified to recognize dynaflow setups cell and handle them appropriately. In particular, when a setup cell is received on a previously unused VCI, the IPP forwards the setup cell to one of a number of dynaflow control processors and then buffers data arriving on that virtual circuit in a per VC buffer until the control processor makes its switching decision. When a dynaflow control processor receives a setup cell, it performs a routing operation to determine which links can be used to reach the destination, and selects a link from this set that has sufficient unused bandwidth to accommodate the new flow. It then forwards the setup cell on the selected link using a previously idle virtual circuit and sends a control cell to the IPP, modifying

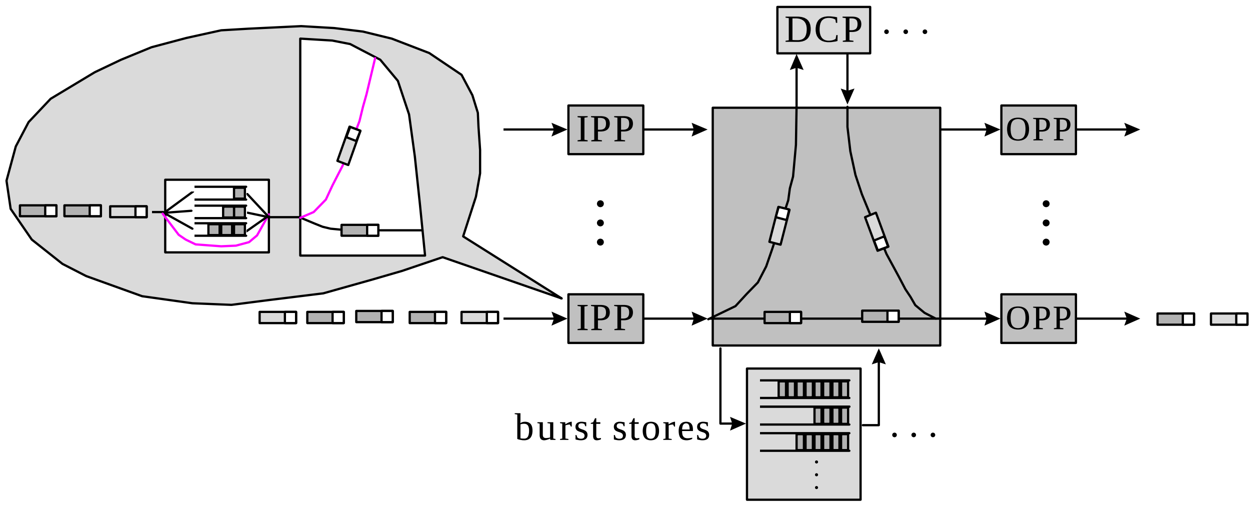

Figure 3. Dynaflow Switching Architecture

its virtual circuit table entry so that it forwards cells to the proper output link and virtual circuit. If no outgoing links to the destination have sufficient unused bandwidth to accommodate the new flow, the control processor handles the flow according to the options specified in the setup cell. If buffering is called for, the burst is forwarded to one of a set of *burst stores*, which maintain separate queues for each data flow passing through them and can forward cells from those queues at rates determined by the control processor. The diversion of a flow into the burst store is implemented through control cells sent by the control processor through the switching network to the IPP and to the selected burst store itself.

Note that the resource allocation decision that must be made by the control processors implementing the dynaflow service is straightforward; they simply keep track of the rate of all flows using a given output link and allow a new flow so long as the sum of its rate and the rates of the current flows is no more than the outgoing link bandwidth (or portion of that bandwidth allocated to the dynaflow service, if the link is shared with non-dynaflow traffic). This allows high link utilization to be achieved with no congestion at switch output ports. Similarly, if a flow must be diverted to a buffer, the rate of data flow into the buffer can be tracked to ensure that the bandwidth of the interface between the switching network and the buffer is not exceeded. The simplicity of dynaflow bandwidth allocation is due to the fact that the sending host specifies only the rate at which data is being sent *right now* rather than specifying some statistical prediction of its transmission rate in the future.

To enable large numbers of dynaflow-capable links in a switch, multiple dynaflow control processors may be needed. The next section explores this and other performance issues in more detail. Based on this analysis, we estimate that a system with 256 external links of 2.4 Gb/s can be implemented using a control complex comprising two shared memory multiprocessors with between 8 and 16 processors each and network interfaces capable of sustained data transfer rates of 1 Gb/s or more. An example design for the control complex is shown in Figure 4. This design uses a two port ATM network interface called the APIC (ATM Port Interconnect Chip) capable of gigabit data transfer rates [5]. Here, the second port is used to link the two multiprocessors together, providing a low latency path for the exchange of status information.

There is a variety of possible ways to distribute the dynaflow processing among the different processors. The key issue here is deciding how to allocate the control over the outgoing link resources (both bandwidth and virtual circuit identifiers). Here we focus on an approach which is particularly suitable in the case of two multiprocessors. First, we define a *link group* as a set of links that share the same pair of endpoints. When forwarding a dynaflow to a particular next hop, any link in the link group can be used (of course, a link group may consist of just a single link). The outgoing virtual circuit identifiers in a link group are ordered over all the links in the group and the two multiprocessors allocate virtual circuit identifiers from opposite ends of this range. A moveable boundary is defined within the range to prevent conflicting allocation of any single virtual circuit. A similar approach is taken with respect to the bandwidth of the link group, with one multiprocessor controlling the “lower half” of the bandwidth range and the other controlling the “upper half.” Of course this approach does allow fragmentation of resources with respect to both bandwidth and VCIs, but has the advantage of requiring little or no communication between the multiprocessors during the processing of setup cells. Responsibility for input links and virtual circuits is also divided between the two multiprocessors, but the division need not be the same as for the outgoing links. Within each multiprocessor, all state information is accessible to the processors in shared data structures. The processor handling a given setup cell uses the shared data structures to allocate an outgoing virtual circuit identifier and bandwidth, updating the data structures appropriately and also sending control cells to the Input Port Processor to initiate forwarding of data.

Each *burst store* implements a collection of queues (one per dynaflow using the burst store) using a large common memory. Each of the queues implemented by a burst store has its own output link, VCI and forwarding rate. The queue controller

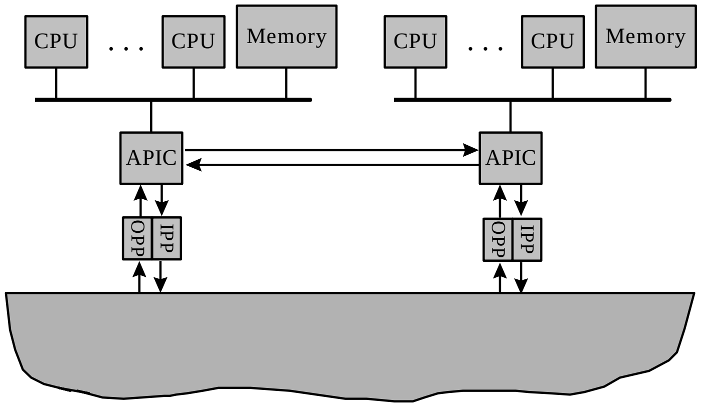

Figure 4. Dynaflow Control Processors

includes a cell scheduler to pace cell transmissions individually for each dynaflow. When a burst is first diverted to a burst store, the control processor either disables forwarding or selects a forwarding rate that is less than the arriving rate. If the link that a given dynaflow is being forwarded to later becomes uncongested, the forwarding rate from the burst store can be increased. In particular it may be increased to some rate higher than the rate at which the original sender is transmitting, causing the queue in the burst store to drain. In such a case, when the queue in the burst store is empty, the flow is re-switched to bypass the burst store and proceed directly to the output link. This can be accomplished using the *transitional time-stamping* technique described in [12] for re-switching multicast virtual circuits. Using this mechanism, cells sent from an input port directly to an output port, immediately following a re-switching operation are delayed at the output port allowing cells following the path through the burst store to catch up and be correctly sequenced at the output. The rate at which a burst is forwarded during the period that the burst store queue is draining is constrained by the bandwidth available on the outgoing link and by the need to avoid excessively increasing the burstiness that downstream switches are subjected to. To match the bandwidth of the switch in [2], each burst store requires a memory bandwidth of about 1 gigabyte per second. This can be achieved using eight SRAM chips with 16 data pins and 16 ns cycle times. Assuming 4 Mb chips, this yields 4 MB per burst store. Of course, the buffer dimension will also depend on queuing considerations, which are examined in the next section

4. Performance Considerations

In this section, we explore the performance issues associated with dynaflow switching. In particular, we focus on the performance of the burst stores, and the dynaflow control processors. In order to assess the feasibility of large-scale use of the dynaflow service, we consider the performance of a large ATM switch with 256 external links of 2.4 Gb/s each, in which dynaflow traffic is the only traffic type. While in practice, we would expect traffic to be divided among several services (including conventional IP datagrams and ATM VCs), the dynaflow-only assumption allows assessment of the system's ability to support the dynaflow service under the most extreme conditions.

4.1. Burst Store Performance

In this section, we evaluate the potential for cell loss in the burst stores. There are two cell loss mechanisms to consider. First, cells may be lost because at the time a dynaflow becomes active, there is no burst store with sufficient unused input bandwidth to handle the additional traffic. Second, cells may be lost because a burst store runs out of room to store arriving cells. In our analysis, we assume that a particular dynaflow can only use a single burst store, meaning that cells can be lost even when there is storage available in other burst stores. While it is certainly possible to improve cell loss performance by allowing cells from a given dynaflow to use more than one burst store, we don't consider that possibility here. Our analysis also does not explicitly model the positive effect of routing flexibility on the queuing performance. That is, we assume that each arriving flow has only one outgoing link that it can use. This is a worst-case situation. Certainly, in a network backbone environment, we would expect both parallel links between switches and the availability of multiple alternate routes, which would lead to better performance.

4.1.1. Analysis of cell loss due to insufficient input bandwidth at burst stores

Let x_r denote the random variable for the number of excess flows associated with output r. Equivalently, x_r is the number of flows addressed to output r that are using a burst store. To evaluate cell loss due to burst store input bandwidth limits, we first determine the distribution of the sum of the x_r s. This will be conservatively approximated by taking the convolution of the distributions of the x_r s. For uniform traffic the x_r s are identically distributed and we can calculate the distribution using a Markov chain analysis. Each output link is assumed to carry on-off bursty traffic from m end-to-end application sessions, each with a peak-to-average ratio of β and a peak data rate equal to $1/k$ times the link rate where k is an integer.

Assuming exponentially distributed burst lengths and inter-burst times, we can model this as a two-dimensional continuous

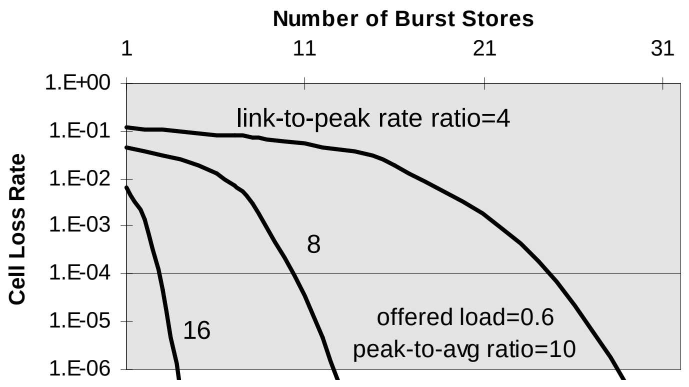

Figure 5. Cell Loss Due to Burst Store Input Bandwidth Limits

time Markov chain with states indexed by two integers *i* and *j*, where *i* denotes the number of active sessions sending data directly to the output (so $0 \le i \le k$) and *j* denotes the number of active sessions sending data to a burst store (so $0 \le j \le m - k$). When *i*>0, there is a transition from state (*i*,*j*) to state (*i*–1,*j*) with rate *i*/*kB* where *B* is the average burst length. Similarly, when *j*>0, there is a transition from state (*i*,*j*) to state (*i*,*j*–1) with rate *j*/*kB*. When *i*<*k*, there is a transition from state (*i*,*j*) to state (*i*+1,*j*) with rate $(m-(i+j))/(b-1)kB$. Finally, when *i*=*k* and *j*<*m*–*k* there is a transition from state (*i*,*j*) to state (*i*,*j*+1) with rate $(m-(k+j))/(b-1)kE$. We can find the steady state probabilities $\pi_{i,j}$ of the chain numerically and then $\Pr\{x_r = j\} = \sum_i p_{i,j}$.

Results from this analysis are shown in Figure 5. For flows whose peak rate is at most one eighth the link rate (300 Mb/s with 2.4 Gb/s links), acceptable loss rates can be achieved with fewer than 16 burst stores. For peak rates of one fourth the link rate, the number of burst stores must be increased to close to 30.

4.1.2. Cell loss due to burst store overflow

To determine the cell loss due to burst store overflow, we performed a series of simulations, again for on-off bursty traffic. In these simulations, new dynamic flows arriving for congested links were sent to the burst store with the lowest input traffic rate. Data stored in burst stores was forwarded to the target output link using the bandwidth left unused by flows passing straight through the switch to the output, with this residual bandwidth shared equally among all bursts waiting for that output. Selected results from the simulations are shown in Figure 6. To achieve small loss rates (at the given offered load of 60%) with small ratios of link rate to peak rate requires buffer sizes of one or more bursts per output link. For larger ratios of link rate to peak rate, the amount of buffering needed can be a fraction of a burst per output link. For the case of a link-to-peak ratio of eight, we need about .6 bursts worth of buffering per output. Since this is distributed over 16 burst stores and there are 256 output links, each individual burst store requires enough buffering for about 10 bursts in this case. So a burst store of 4 Mbytes could handle average burst sizes up to about 400 Kbytes and a burst store of 16 Mbytes could handle average burst sizes of about 1.6 Mbytes.

Burst store performance can be affected by the policy used to select the burst store to receive a given burst. We briefly discuss the issues here and summarize results from a preliminary comparison of the different approaches. Several burst store selection policies are listed below.

- *Least-loaded input*. One of the simplest policies to implement (and the one assumed in the simulations reported above) is to select the burst store with the lightest input load. That is, the one with the smallest rate of arriving traffic.
- *Slowest buffer growth rate*. In this policy, we select a burst store from among those with sufficient input bandwidth to handle the new burst and which has the smallest buffer growth rate from among these burst stores. The buffer growth rate is simply the difference between the arriving cell rate and the total outgoing traffic rate.
- *Longest time to overflow*. In this policy, we use both the buffer-growth rate and the current buffer level to estimate at what time in the future each burst store will overflow and route the new burst to the burst store whose overflow time is furthest in the future. If there is one or more burst stores whose buffers are currently draining, we route the new burst to the burst store that is projected to become empty earliest.

The first two policies are easy to implement, since they require only information that the DCP has at its immediate disposal. The third policy also requires the current buffer level which the DCP would either have to retrieve from the burst stores or estimate, making its implementation somewhat problematical. If burst lengths are known in advance, one can actually project the longest time to overflow precisely, rather than simply estimating it, based on rates and current buffer level. This optimal policy is impractical to implement, but provides a useful benchmark for evaluating the simpler policies. We have found that at heavy loads, the least-loaded input policy can lead to up to ten times the cell loss experienced with the optimal policy. The slowest buffer growth rate policy did slightly better and the longest time to overflow policy was better yet. However, the differences among the three practical policies were all fairly small (at most a factor of two in cell loss rate).

There are two other aspects of the burst store operation that can affect cell loss performance. The first one is the *burst diversion policy*, which determines which bursts get diverted to the burst store when an output link becomes overloaded. The simplest approach (and the one used in the simulations above), is a first-come-first-served policy which diverts those bursts that arrive when the output link they need is congested. Another policy is to divert to a burst store those bursts that are sending at the highest rates. This improves the service received by low rate bursts and tends to penalize those bursts that are contributing most to the congestion condition. It also has the additional benefit that fewer high speed bursts must be diverted to reduce traffic to the output link, thereby reducing the amount of work that must be done by the BCP to divert bursts to burst stores.

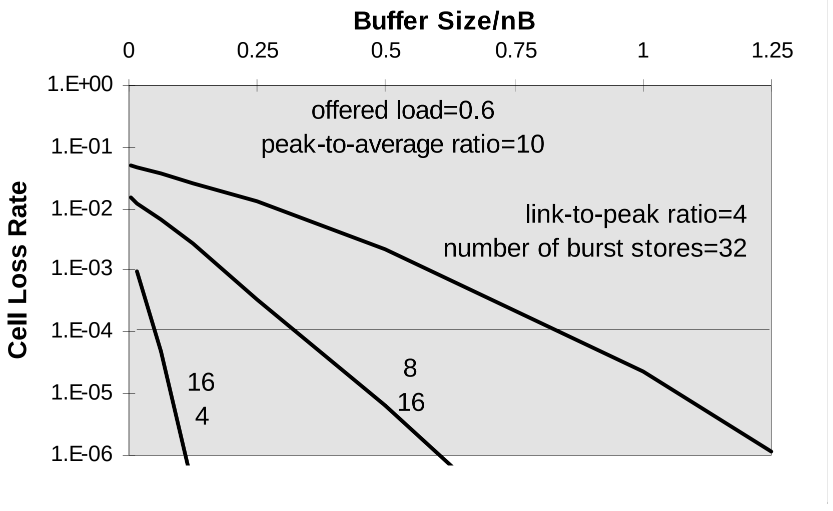

Figure 6. Cell Loss Due to Burst Store Overflow

Cell loss performance can also be affected by the *forwarding policy* that determines the rate at which data is forwarded for bursts that have been diverted through burst stores. The policy used in the simulations reported above, simply shares the bandwidth left over from those bursts not diverted to a burst store, equally among all bursts that are diverted. When the link becomes uncongested, this can lead to high forwarding rates from the bursts still passing through burst stores, increasing the burstiness of traffic forwarded to downstream switches. This effect can be avoided by limiting the forwarding rate to the original sending rate, or something slightly larger (say 10% more). Our preliminary studies of this show that limiting the forwarding rate has very little effect on the cell loss rate experienced by the burst stores.

4.1.3. Other Issues

While our analysis does not explicitly model the effect of routing, one can infer something about the special case in which links are grouped, with all links in a given group going to the same next hop. If there are *p* links in each group, then we can treat the group as a single link with a high bandwidth and hence a higher link-to-peak rate ratio. By this reasoning, Figure 5 can be re-interpreted for a system with 256 link groups. To perform this re-interpretation, replace the “link-to-peak rate ratio” label with “link group-to-peak rate ratio” and multiply the horizontal axis labels by *p*. For Figure 6, we can perform a similar adjustment by replacing “link-to-peak rate ratio” with “link group-to-peak rate ratio”, multiplying the number of burst stores by *p* and interpreting *n* as the number of distinct link groups.

4.2. Dynaflow Control Processor Performance

Perhaps the most crucial performance issue for the dynaflow service concerns the dynaflow setup latency, since input port processors must buffer arriving data cells while the control processors handle the setup cell. To limit the IPP buffer requirements, we would like to keep the setup cell latency low and highly predictable. The setup cell latency will depend on how the workload from arriving setup cells is handled by the different processors within each multiprocessor. In the simplest case, each processor has its own input queue and arriving setup cells are simply distributed randomly to one the queues. This can be modeled by an M/D/1 queue. Figure 7 shows the response time distribution for an M/D/1 queue. For an average load of 50% on the control processor, the probability of the response exceeding 12 times the amount of time needed to actually process the setup cell, is about 10^{-6}. If the time to process a setup cell is 10 µs and the time for a cell to go from an input port processor to the control processor is also at most 10 µs, then with high probability the input port processor will need to buffer cells for a newly active flow for no more than about 140 µs. At 2.4 Gb/s, this corresponds to 42 Kbytes of storage. If an arriving setup cell finds the control processor idle, the amount of data that can accumulate while the setup cell is being processed drops to about 9 Kbytes in the example scenario. We can make this best case behavior common, if instead of randomly distributing arriving setup cells to processors, we place them in a common queue which can be serviced by any idle processor.

To determine the number of processors needed to handle a given system, we must determine the time required to process a burst setup cell. We assume a model in which the APIC transfers cells directly into a circular list in the BCP’s memory and the multiple CPUs poll this list to retrieve incoming setup cells and process them directly. To minimize overhead for gaining access to shared data structures, all BCP processing is done by kernel-resident programs using hardware test-and-set instructions for mutual exclusion. With these assumptions, we can outline the steps involved in processing a setup.

1. Scan APIC buffer list to find a new setup cell that is not already being processed by another processor. Exclusive access to buffers is implemented using a per buffer lock.
2. Lock *input link table* for link on which setup cell was first received. Transfer information from setup cell to entry in input link table for the virtual circuit specified in setup cell. Mark virtual circuit entry to indicate that a setup is in

progress and unlock input link table (this allows other CPUs to access other virtual circuit entries in the input link table).

3. Perform an IP routing lookup using the destination address in the setup cell. This results in a link group, or possibly a list of link groups which can be used to reach the destination.
4. Scan the links in the link group (or groups) for one with sufficient available bandwidth. If a suitable link is found, lock the *output link table* for that link and update the field that specifies the amount of bandwidth still available. Select an unused outgoing VCI and enter the information identifying the input link and VCI that are assigned to that outgoing VCI in its table entry. Unlock output link table.
5. Enter the selected output link and VCI in the input link table entry associated with the input VCI.
6. Queue a copy of the burst setup cell for transmission by the APIC and forwarding through the switch and out on the selected output link and VCI.
7. Format a control cell and queue it for transmission by the APIC for delivery through the switch to the IPP. This control cell will cause the output link and VCI information to be written into the IPP's routing table and will enable the flow of cells from the IPP to the outgoing link. The IPP returns the control cell as an acknowledgement.
8. Verify that the control cell has been acknowledged.

Locking at the granularity of individual input or output link tables ensures that lock contention is rare. To avoid a long delay in step 8, processors should not wait for the control cell acknowledgement, but continue processing new setup cells and check back between successive setup cells to verify reception of expected acknowledgements. Using this approach, we estimate that a processor can fully process one burst setup cell in no more than 1,000 instructions.

Given a target processor utilization and an estimated instruction count for setup cell processing, we can estimate the number of control processors needed in the sample system. Assume that each processor has an effective instruction processing rate of 100 million instructions per second and that one out of every 1,000 cells on each of the 256 input links is a setup cell that must be processed by a control processor. This leads to one setup cell on each 2.4 Gb/s input link every 177 μs or rough 1.5 setup cells per microsecond overall. At a target processor utilization of 50%, each processor can handle an average of one setup cell every 20 microseconds, so about 30 processors are needed overall to handle this load. This is about one for every eight external links. Obviously, the number of processors required scales linearly with all of the various parameters. For fully loaded external links, the offered load of one setup cell in every 1,000 cells corresponds to burst sizes of about 50 Kbytes. This is an indication that while the dynaflow service can be very efficient for burst sizes on the order of tens of kilobytes, it may be less attractive for smaller burst lengths.

6. Summary

This paper has introduced a new communication service for ATM networks that provides greater flexibility and efficiency for bursty data traffic, particularly when session durations are short, as is often the case for web browsing and similar applications. The dynaflow service achieves these advantages by allocating resources only at burst transmission time, which for many data applications is the earliest time at which the resource requirements are known. The dynaflow service is designed to operate efficiently in an open-loop mode initially, but end-to-end congestion control mechanisms at either the ATM level or the transport protocol layer can also be used in combination with the dynaflow service for data transmissions that last for long enough to make these mechanisms effective. The service operates most effectively when there is a high ratio of link (or link group) bandwidth to dynaflow virtual circuit bandwidth and/or the switches have buffers that are substantially larger than the average burst size. In situations where there is ample routing diversity, higher data rates can be accommodated. Given, the increasingly large data bursts associated with interactive services, the use of a shared buffer pool can be important for keeping system costs low. The dynaflow service makes it possible to implement shared queuing using independent burst stores whose behavior are under the control of the dynaflow control processors. The performance evaluation given here is preliminary. Its purpose has been to provide an initial assessment of the viability of a dynaflow service and identify the various issues that need more detailed examination. We believe these studies have shown that a dynaflow service has some promise. More detailed studies are needed to determine how best to engineer various components and to determine the effective limits of operation. The issue of how dynaflow's routing flexibility affects its performance is a particularly interesting one to explore.

We view the dynaflow service as providing one service within a multi-service cell-switched network. Such a network would also provide conventional ATM virtual circuits, which are well-suited to applications with session durations of more than a few tens of seconds, and IP packet switching, which is ideal for transaction-type traffic, which typically involves simple query and response pairs, with relatively little data exchanged (typically, a few hundred bytes). The same hardware (and much of the software) infrastructure that supports the dynaflow service can also be used to support IP forwarding and for ATM virtual circuit establishment

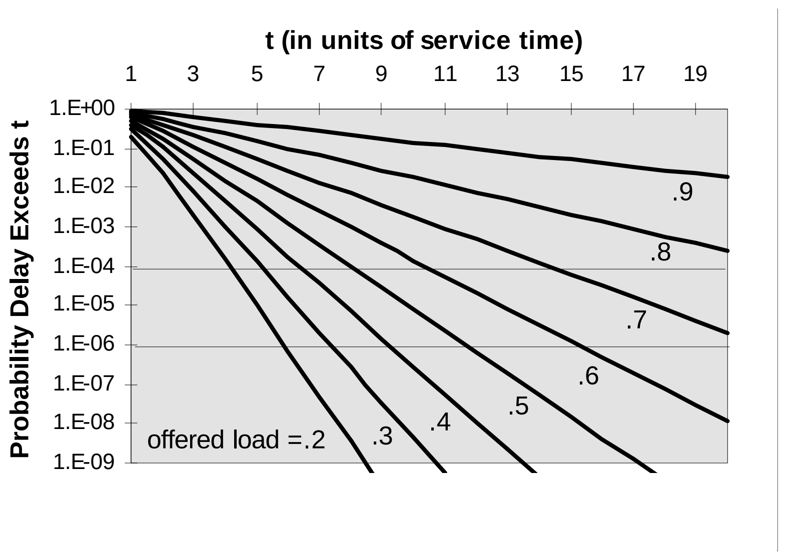

Figure 7. Control Processor Delay Distribution

References

1. Boyer, Pierre E. and Didier P. Tranchier. "A Reservation Principle with Applications to ATM Traffic Control," *Computer Networks and ISDN Systems*, 1992, 321-334.
2. Chaney, Tom, J. Andrew Fingerhut, Margaret Flucke, Jonathan Turner. "Design of a Gigabit ATM Switch," *Proceedings of Infocom*, 4/97.
3. Cidon, Israel, Raphael Rom and Yuval Shavitt. "A Fast Bypass Algorithm for High Speed Networks," *Proceedings of Infocom*, 1995, 1214-1221.
4. Cidon, Israel, Raphael Rom and Yuval Shavitt. "Analysis of One-Way Reservation Algorithms," *Proceedings of Infocom*, 1995, 1256-1263.
5. Dittia, Zubin, Jerome R. Cox, Jr. and Guru Parulkar. "A High Performance ATM Host-Network Interface Chip," *Proceedings of Infocom*, 1995.
6. Hui, J. Y. "Resource Allocation for Broadband Networks," *IEEE Journal on Selected Areas in Communications*, 1988, 1528-1608.
7. Jain, Raj. "Congestion Control and Traffic Management in ATM Networks: Recent Advances and a Survey," *Computer Networks and ISDN Systems*, 10/96.
8. Ohnishi, H., T. Okada and K. Noguchi. "Flow Control Schemes and Delay Tradeoffs in ATM Networks," *IEEE Journal on Selected Areas in Communications*, 1988, 1528-1608.
9. Suzuki, Hiroshi and Fouad A. Tobagi. "Fast Bandwidth Reservation Scheme with Multi-Link and Multi-Path Routing in ATM Networks," *Proceedings of Infocom*, 1992, 2233-2240.
10. Jonathan Turner. "Managing Bandwidth in ATM Networks with Bursty Traffic," *IEEE Network*, vol. 6, no. 5, September 1992, 50-58.
11. Jonathan Turner. "An Optimal Nonblocking Multicast Virtual Circuit Switch," *Proceedings of Infocom*, June 1994, pp. 298-305.