Striping within the Network Subsystem

C. Brendan S. Traw   Jonathan M. Smith

Abstract

Striping is a general architectural technique for the transparent aggregation of multiple, functionally similar, resources to obtain higher performance.

This paper establishes a precise terminology for discussing striping and striping techniques, differentiates striping from the more general architectural technique of multiplexing, and outlines the major uses and trade-offs associated with the technique within the network subsystem.

Opportunities for applying striping to high performance network subsystems are explored through a systematic case study. The case study uses a network subsystem consisting of TCP/IP operating over an ATM/SONET infrastructure. Four striping points are analyzed in detail: *TCP striping* by the application, *IP packet striping* by the TCP layer, *ATM cell striping* by the ATM adaptation layer, and *byte striping* by the ATM layer. Two of these points are further evaluated using a simulation driven by a trace of a *World Wide Web* session.

A principal result of the exploration is a set of engineering trade-offs among the striping points within this high performance network stack. These trade-offs are a significant aid in choosing the appropriate striping point for use within the network subsystem.

1 Introduction

Within high performance network subsystems, the optimization of a single datapath between an application running on a host to the network is typically the most straightforward approach to obtaining increased performance. Examples of this type of optimization include minimizing data copies and other effects of layering [2], and efficient utilization of I/O resources. At some point, further optimization of this single path may be either impossible or impractical. In either case, the aggregation of multiple, parallel datapaths can be a solution for obtaining further improvements in overall system performance. Striping is a technique developed for use within the disk subsystem and is a key aspect of Redundant Arrays of Inexpensive Disks (RAID) architectures [11]. It is also the solution for augmenting the performance of network subsystems through the aggregating of multiple datapaths which is explored in this paper.

This paper is structured as follows: The next section establishes a precise terminology for describing striping and its relationship to multiplexing. Section 3 motivates the use of striping within the network subsystem. Section 4 is a discussion of other work in the area. Section 5 presents a set of criteria for evaluating network striping and a case study of striping within a typical high performance network stack consisting of TCP/IP over ATM/SONET. Section 6 concludes the body of the paper and presents a set of "rules of thumb" for the inclusion of striping within the network subsystem. An appendix is included which describes the trace driven simulation used to support the case study.

2 What is Striping?

Striping is a term which has been widely used to describe instances where physical resources have been aggregated together to obtain higher performance. As the application of this term has spread from its origin within the disk subsystem to the networking subsystem and beyond, the range and details of resource aggregations which have been labeled as striping has greatly expanded the applicability of the term. This is partially a result of the lack of a clear definition for the term. The following discussion develops one. The definition is broad enough to include most instances of

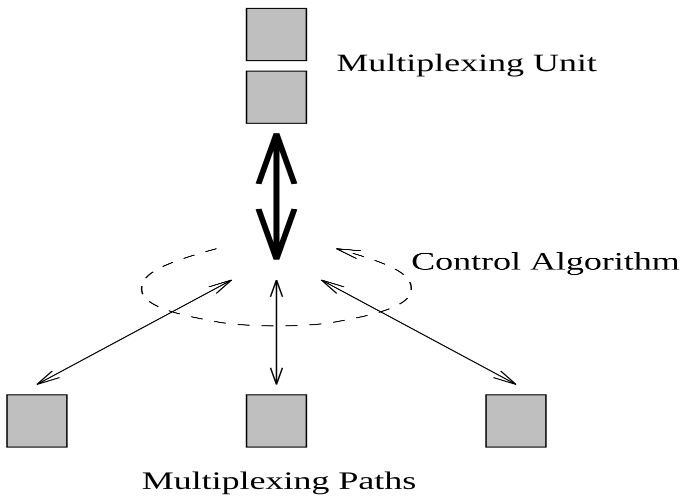


Figure 1: Characteristics of a (De)Multiplexing Point

resource aggregation which have already been labeled striping, but narrow enough to differentiate it from the more general term *multiplexing*. Striping can be distinguished from other architectural techniques which exploit parallelism, such as pipelining, because all of the parallel resources must be interchangeable.

2.1 Multiplexing

To understand the properties of striping, it is useful to first examine a more general architectural technique, multiplexing. The most general definition of multiplexing is as a means of combining multiple data inputs into a single output in such a manner that the data on the inputs can be recovered through the process of demultiplexing. *Demultiplexing* (or *inverse multiplexing*) is the inverse operation of multiplexing – taking a single input and spreading that input across multiple outputs. A system using both multiplexing and demultiplexing is a *multiplexed system*.

Multiplexing can be divided into two major categories, *physical* and *logical* [6]. Types of physical multiplexing include frequency/wavelength (FDM/WDM), time (TDM), and space (SDM). Logical multiplexing within the network subsystem involves the mapping of multiple layer n (where $n > 1$) streams into a single stream in layer $n-1$ which share the same physical resource [1].

Any logical or physical location in an architecture where multiplexing or demultiplexing takes places is called a multiplexing (or demultiplexing) point. A given (de)multiplexing point may perform both physical and logical (de)multiplexing. There are three primary characteristics of a (de)multiplexing point which are illustrated in Figure 1 and enumerated below.

1. Control Algorithm

   The control algorithm for a multiplexing point determines the path of a multiplexing unit through the multiplexing point.

2. Number of paths being multiplexed

   This parameter for WDM, SDM, and TDM is the number of wavelengths, instances, or time slots which are available to be multiplexed. Borrowing from the terminology developed in the memory subsystem, a multiplexing situation where there are n paths is called an *n-way* multiplexed system.

3. Multiplexing unit

   The multiplexing unit is the unit of data which the multiplexing algorithm operates upon.

[1]In this definition, layer n refers to a layer of the OSI model where Layer 1 is the physical layer.

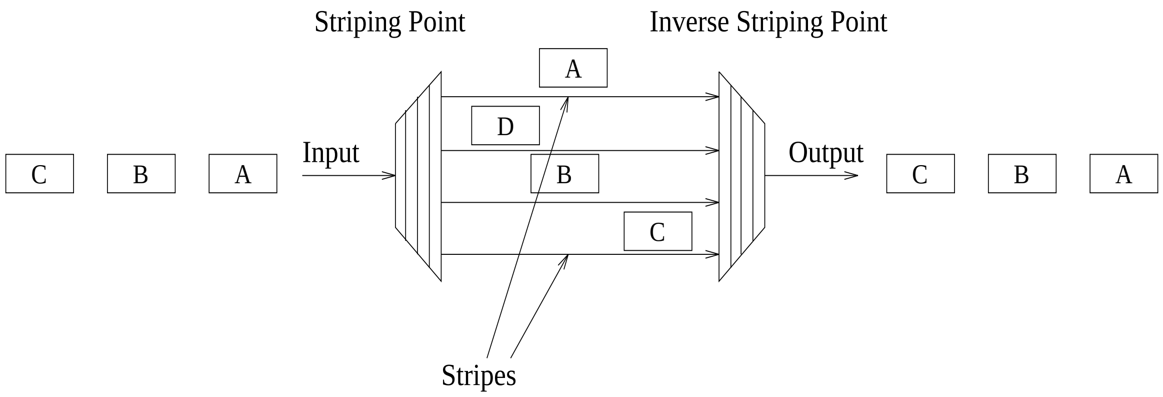


Figure 2: A Striped Communications System

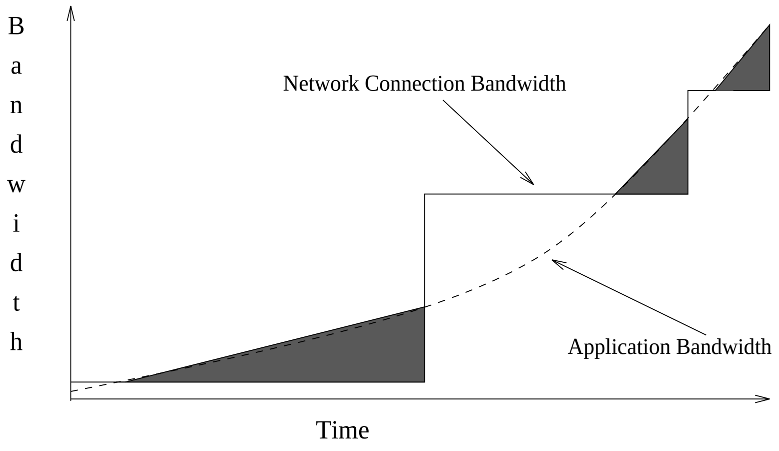


Figure 3: Network Standards and Application Bandwidths

2.2 Relationship between Striping and Multiplexing

Striping and multiplexing may appear to be synonyms. They are not.

Striping is a subset of the design space encompassed by the technique of multiplexing. Striping is physical multiplexing where the operation of the multiplexing algorithm is *transparent* to higher level functions except for the increased performance realized. Ensuring this transparency can be difficult if properties such as the preservation of order must be maintained across the stripes. The more general technique of multiplexing does not make any such transparency guarantees[2].

A *striping point* is at minimum a physical demultiplexing point which conforms to the restrictions imposed above. A striping point must include a physical demultiplexing point since striping requires that multiple *physical* resources be aggregated together. Logical (de)multiplexing may also be performed. An *inverse striping point* is a location where the inverse activities are performed to recover the data units which entered the striping point. A *stripe* is an instance of the resource which is connected to one of the outputs of a striping point and one of the inputs of an inverse striping point. Figure 2 shows an abstract striped communications system where order must be preserved to maintain transparency.

When a striped system that must preserve ordering is operating correctly, it is said to be *synchronized.* Upon startup, or following the loss of data, such a striped system is said to be *unsynchronized.* To regain synchronization, the striping and inverse striping points must have a procedure to restore the ordered property of data being carried by the striped system.

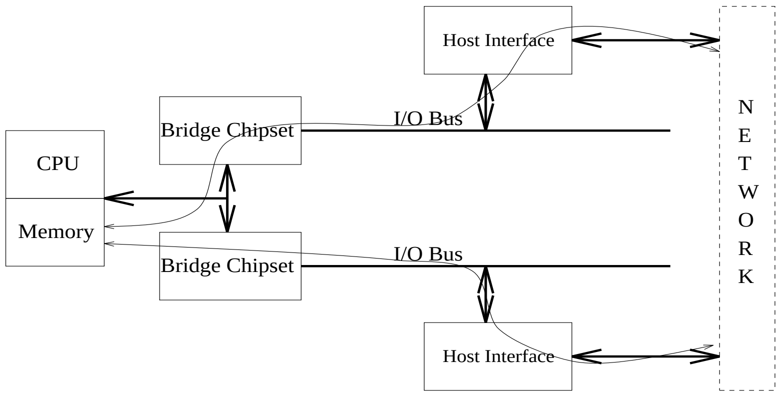


Figure 4: Striping for Bottleneck Avoidance

3 Applications of Striping in the Network Subsystem

Striping, a general purpose technique for aggregating resources, can be applied to a variety of situations within the network subsystem.

3.1 Network/Application Bandwidth Mismatch

The maximum bandwidth which can be supported by a single physical network connection is driven by standards and the infrastructure which has been deployed. As such, it tends to improve in a stair-step manner. The collection of applications running on network endpoints, due to their number and diversity, do not increase their need for bandwidth in such a rigid manner. Network striping provides the flexibility necessary to match network bandwidth to that which is needed by the applications running over the network. Figure 3 illustrates this pattern. The shaded areas are domains where the use of network striping could be beneficial.

3.2 Bottleneck avoidance

Striping can also be used to circumvent architectural and implementational bottlenecks. For instance, in a network I/O subsystem, if a single network interface, I/O bus, or any other hardware component in the system is incapable of supporting the desired bandwidth, then multiple instances of that component might, in aggregate, support it. Figure 4 shows how striping could be applied to resolve a bottleneck in a processor/memory bus to I/O bus bridge chipset such as that reported in [15] when multiple copies of the bottleneck resource are available.

3.3 Driving Function for New Network Technologies

A final observation is that striping is an effective approach for motivating technological improvements in commercially available network services. A step up in deployed technology takes place when the technology is sufficiently mature and when it can be shown that there are applications which will be able to use it – thereby generating income for the provider. It is often difficult to demonstrate that applications can take advantage of a new service rate before it is actually available because the applications have not been developed since there is nothing to run them on.

Network striping can break this cycle by permitting applications to obtain the network services they require by aggregating the capabilities of multiple, currently existing links, thus demonstrating the need to deploy the next service increment.

[2]The authors appreciate the input of Bruce Davie, David Feldmeier, and David Sincoskie in the formulation of this definition.

Effort	Link Rate	Number	Multiplexing Unit	Algorithm
Bellcore (Nectar)	155 Mbps	16	HIPPI Packet	FCFS
LANL	155 Mbps	7 + parity	Byte	RR
IBM	155 Mbps	4	PTM Packet	FCFS
Bellcore (Aurora)	155 Mbps	4	ATM Cell	Trunk Group

Table 1: Other Network Striping Work

4 Other Network Striping Work

Disk striping [12] and network striping have the same goal of aggregating multiple resources to improve performance, but unfortunately much of the experience developed within the disk subsystem is not directly applicable for networks. This is primarily due to the difficulty in ensuring that the transparency requirement is met as there is the potential for extreme variability in quality of service between stripes in a network environment.

There has been some work in the area of network striping reported in the literature. In each case, striping at only a single layer in the a protocol stack has been examined. No systematic evaluation of the design space has been performed. Table 1 shows the primary attributes of four efforts which have resulted in experimental evaluations of gigabit per second striping.

Bellcore has developed, for the Nectar Gigabit Testbed [3], the HIPPI-ATM-SONET (HAS) adapter [10]. The HAS stripes HIPPI packets across the 16, 155 Mbps STS-3c connections constituting a SONET OC-48 using a first come, first served (FCFS) control algorithm.

Los Alamos National Labs (LANL) has explored a different striping point to provide a similar service [13] within the CASA Gigabit Testbed [3]. The major difference between their work and that performed at Bellcore is that they are striping HIPPI packets across seven STS-3cs (six for data and one for error correction) in a round robin (RR) manner at the byte level. In this case the SONET frame is used to provide alignment between stripes to ensure that byte ordering is maintained.

IBM has explored another point in the network striping design space. They are striping PTM packets across four STS-3cs using a control algorithm similar to the HAS [14].

The final instance of high performance network striping is another Bellcore effort for the Aurora Gigabit Testbed [3]. To obtain the equivalent bandwidth of an STS-12c, the bandwidth of four STS-3cs is aggregated together into a *trunk group* [8]. ATM cells are striped across the link in an order determined by the trunk group control algorithm. This algorithm, by the placement of idle and active cells, allows the receiver to determine the order in which the cells were placed into the trunk group. Unfortunately, this mechanism requires that there be less than one cell time of skew among the STS-3cs. In the course of evaluating the OSIRIS network interface [4] it was determined that within the Aurora testbed, the commercial NEC and Northern Telecom OC-12 to OC-48 multiplexors introduced more skew than was acceptable for the correct operation of the trunk grouping mechanism. The trunk grouping algorithm has since been replaced by other algorithms which explore the use of sequence numbering and higher level framing to compensate for the skew between the stripes which can result in a misordering of ATM cells [5].

One effort toward standardizing a particular application of network striping is proposed by Fredette [7]. He calls network striping *Inverse Multiplexing* when it is performed at the physical layer. The goal of the BONDING standard for narrow band ISDN (N-ISDN) is the aggregation of multiple N-ISDN channels to provide applications with optimum bandwidth at the lowest cost. To overcome skew among the aggregated channels, “training signals” are used to determine the skew among the various channels at the destination. Once this skew is calculated, delays are added at the source to align the data such that it will be received in the correct order at the destination. This scheme assumes that the skew does not change for the duration of the connection.

5 Evaluation of Striping within the Network Subsystem

This section provides an initial, systematic exploration of the network striping options which are possible in a typical broadband protocol stack. These options are evaluated in a comparative manner to expose the characteristics of striping at several layers. These characteristics provide an understanding of the engineering tradeoffs that exist within the network subsystem when selecting a protocol layer in which to embed a striping point.

5.1 Evaluation Criteria

For network traffic, bandwidth and latency are the primary characteristics of a striping point. Other characteristics which are important from a system point of view include the scalability and complexity of implementing the control algorithm, buffering requirements, and ability to tolerate network induced skew between the stripes.

5.1.1 Latency and Buffering

Buffering and latency are separate, but typically closely related characteristics of a striping point. If traffic arriving at a striping point is bursty and the aggregate bandwidth of the striping point is not equal to the peak bandwidth of the burst, then buffering is required to prevent the loss of data. Latency is a measurement of the time required for data to transit the striping point from the input to the output stripes. There are two components to latency. The first is the time spent waiting in the input queue; the second is the time required to actually transmit the data. The second component is dependent on the striping technique used as well as the bandwidth supported by the stripes.

5.1.2 Tolerance of Skew

For the striped systems where ordering must be maintained to preserve transparency, skew among the stripes can be a complication. There are two types of skew. *Static skew* is skew between the stripes that is fixed for the duration of the operation of the striping point. *Dynamic skew* is the component of skew between the stripes which changes over time. A possible cause of this type of skew is congestion or rerouting of a stripe.

5.1.3 Scalability and Complexity

Scalability in the striping context is a characteristic which describes the extent to which it is possible to extend the number of stripes used in the striped system to an arbitrary degree. Some striping algorithms may scale well while others may only work well for relatively small numbers of stripes. For striping systems implemented in hardware, interconnection density, timing, and buffering may limit scalability while in software the scalability of the system may be limited by the computational requirements of the striping algorithm.

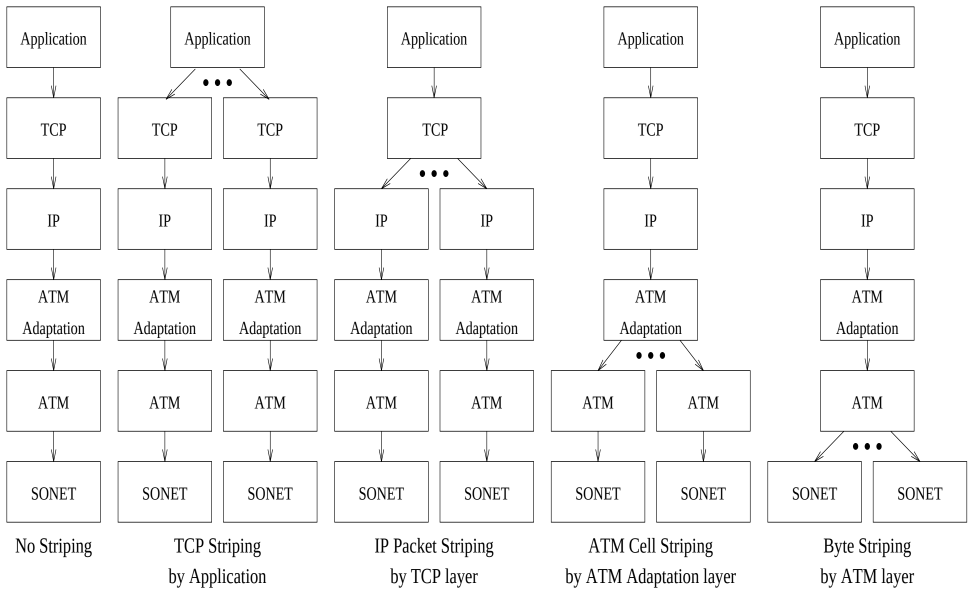
The complexity of implementation is a measurement of how difficult it is to implement the striping algorithm and data paths. This factor is typically closely related to the scalability since more complex algorithms frequently do not scale as well as less complex ones.

5.1.4 Bandwidth

Finally, the maximum bandwidth which the striping system can support is dependent on two factors, the bandwidth of each stripe and the number of stripes available. The maximum *aggregate bandwidth* which can be supported by a striped system is defined to be sum of the bandwidth of all of the stripes.

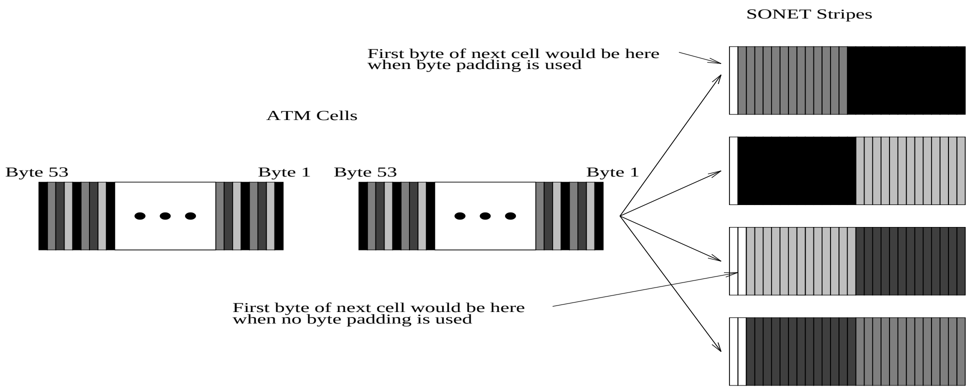
5.2 Case Study

A typical broadband networking protocol stack consisting of TCP/IP over ATM/SONET is used for the case study. The striping points (Figure 5) which will be studied are *TCP striping* by the



Application
TCP
IP
ATM Adaptation
ATM
SONET
No Striping
TCP Striping by Application
IP Packet Striping by TCP layer
ATM Cell Striping by ATM Adaptation layer
Byte Striping by ATM layer

Figure 5: Multiplexing Points to be Evaluated for Striping



SONET Stripes
First byte of next cell would be here when byte padding is used
ATM Cells
Byte 53
Byte 1
Byte 53
Byte 1
First byte of next cell would be here when no byte padding is used

Figure 6: Byte Striping at the ATM Layer

application, *IP packet striping* by the TCP layer, *ATM cell striping* by the ATM adaptation layer, and *byte striping* by the ATM layer.

5.2.1 Byte Striping at the ATM Layer

ATM layer striping involves the striping of ATM cells across multiple SONET physical layer [1] instances on a byte by byte basis. Byte striping must preserve byte ordering.

An initial issue which must be resolved is how to map ATM cells across multiple SONET payloads. Unless the number of stripes is a multiple of 53 (the number of bytes in an ATM cell) the first byte of each cell will rotate across the available stripes as cells are transmitted. This situation is illustrated in Figure 6.

This rotation can increase the complexity of the striping and inverse striping point since they must keep track of which stripe the first byte of the next cell will arrive on. Two techniques can be used to assist in identifying the first byte of an ATM cell:

- Alter the physical layer framing to support striping.

  The SONET (or other physical layer) frame structure could be modified to provides a pointer to the beginning of the first complete ATM cell carried on a particular physical stripe. Since

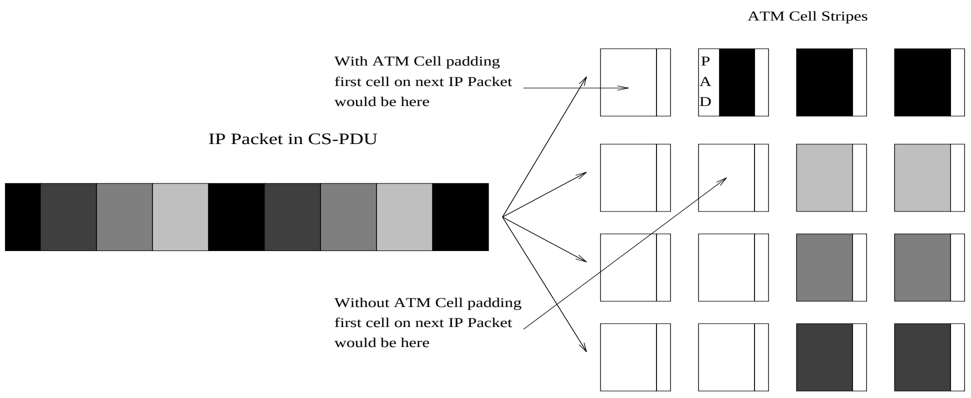

ATM Cell Stripes
With ATM Cell padding first cell on next IP Packet would be here
P A D
IP Packet in CS-PDU
Without ATM Cell padding first cell on next IP Packet would be here

Figure 7: ATM Cell Striping at the AAL

the first byte of an ATM cell could be located on any of the stripes, the striping algorithm must be able to insert pointer values into and monitor the physical layer overhead on each stripe.

- Padding can be added to the end of each cell to ensure that the beginning of the next cell will always be on the same stripe.

  This technique is also illustrated in Figure 6.

The second technique reduces the complexity of the striping algorithm and can be used with unmodified physical layers. Its disadvantage, however, is that it requires additional overhead in the form of padding bytes. Neither of these techniques address the issue of establishing the initial striping system synchronization or resynchronizing the system in the event of data loss or physical layer frame corruption. Two possible ways of handling this function are:

- Synchronize on physical layer framing [13].

  In the case of SONET, as long as the maximum skew between stripes is less than the length of a SONET frame, frame structure can be used to align the stripes.

- Synchronize based on a parity stripe.

  If the maximum skew between the stripes is small, then reserving one of the stripes to contain parity information may provide another synchronization mechanism. To establish synchronization, the alignment of the stripes could be adjusted within this the range of potential skew until there is parity match.

The parity stripe is useful even if the SONET frame is used for synchronization. It would allow for the prompt detection of improper stripe synchronization and provide a mechanism to detect when synchronization has been restored.

Ultimately, particularly in environments where there is a large amount of dynamic skew, striping at the ATM layer may be impossible. Sequence numbering, a technique which is typically used to ensure ordering, is impractical on objects as small as bytes.

Although byte striping has some limitations, it can be easy to implement in hardware and has the potential to scale quite well. Gbps byte striping implementations are not currently possible in software due to the extremely high rate at which the striping algorithm would have to operate.

5.2.2 ATM Cell Striping at the ATM Adaptation Layer

ATM cell striping at the ATM adaptation Layer takes AAL CS-PDUs as input to the striping point and then distributes whole cells across the stripes (Figure 7). Because the data unit being transferred across the stripes is the same as would be sent in a non-striped system, no modifications

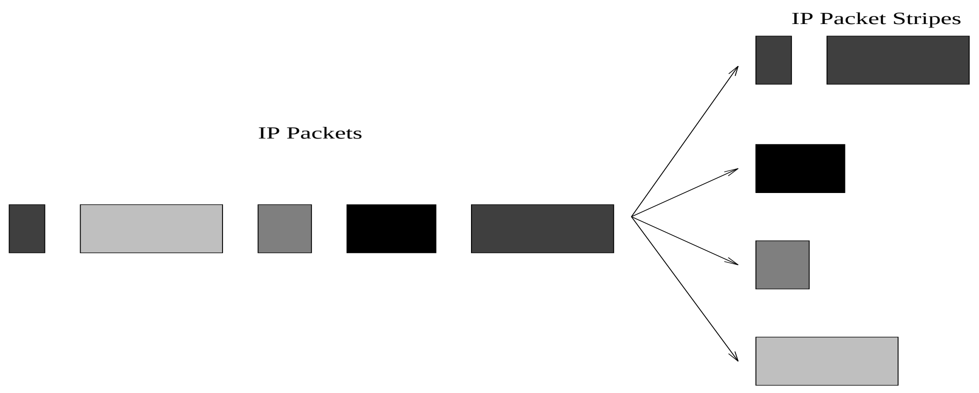


Figure 8: IP Packet Striping at the TCP Layer

to the physical layer framing are required. ATM cell ordering must be preserved across the striped system to ensure that the striping is transparent to higher layer function.

ATM Cell Striping is the lowest form of striping in the protocol stack being studied which can robustly handle the full range of problems associated with skew and loss. The techniques presented in the previous section for handling synchronization and padding at the byte level can also be applied to ATM cells. However, if the skew between stripes is greater than a SONET frame, sequence numbering can be applied. ATM cells are the smallest data entity in this stack for which sequence numbering is practical. Sequence numbers must be of sufficiently large to permit the reordering of cells following static or dynamic skew between the stripes and the detection of cell loss. The number of bits of sequence number must be greater than:

$$\lceil log_2 \frac{Skew_{Max} * \#_{Stripes}}{Time_{Cell}} \rceil$$

Unfortunately, adding sufficiently large sequence numbers to the ATM cells will require the use of a non-standard header format or the creation of a customized adaptation layer to support striping.

Another approach which does not rely on sequence numbering is to embed additional ATM cells at regular intervals on separate virtual channel into the cell streams of each of the stripes. At the inverse striping point these cells can be used to synchronize the time relationship among the stripes which permits the recovery of the original ordering of the ATM cells [9]. This technique is particularly useful when data streams with different destinations go through a single striping point.

Skew among stripes has a negative impact on the latency and buffering required by the system. The overall latency of the system can be increased by up to the maximum skew between the stripes because of the ordering constraint imposed by ATM cell striping. ATM cells arriving on the unskewed or less skewed stripes must be buffered until the cell on the most skewed stripe arrives, thus the quantity of buffering required is proportional to the number of stripes and the range of skews between the stripes.

ATM cell striping is relatively simple to implement in hardware or software provided that no reordering is necessary. Processor based reordering at the inverse striping point is possible for moderate physical layer rates and small numbers of stripes. The fixed quantity of instruction cycles will limit the speed and scalability of these solutions. Hardware implementations of the striping and inverse striping algorithms will provide better performance. The main limitations on the scalability of hardware solutions are the I/O requirements of the elements performing the data stream multiplexing and demultiplexing as well as the range of possible misordering to be corrected.

5.2.3 IP Packet Striping at the TCP Layer

Striping at the TCP layer involves striping the IP packets generated by the TCP layer across multiple lower level stacks. Thus, each IP packet will traverse a single stripe (Figure 8). Skew between the stripes can only introduce a misordering of IP packets. Due to the multipath properties of IP networks, there is no need to ensure ordering across the striped system.

IP striping is ideal for software implementation since the data units are relatively large, particularly in comparison to bytes or ATM cells. Host software is typically the only means of implementation since the protocol stack from the AAL layer up is almost always implemented in host software.

One problem present in lower level striping which IP striping can improve upon is head-of-line blocking. Head-of-line blocking occurs when the data units being striped are of variable size and small data units are forced to wait while a larger one is being transmitted. If smaller packets can be moved forward in the striping queue, the delay experienced by these smaller packets is greatly decreased while the delay experienced by the larger traffic is only slightly increased. Striping at the TCP layer provides such a means. TCP layer striping provides multiple IP packet stripes each capable of serving a separate IP packet. Because there are multiple stripes, service of a packet can begin immediately unless all of the stripes are already busy. Increasing the number of stripes reduces the effects of head-of-line blocking.

The major disadvantage of IP packet striping is that the aggregate bandwidth of the striping point is not available to each IP packet being striped. If the average transmission time for a PDU is shorter than the average inter-arrival time, then the system is effectively no longer striped. This results in greater striping latency since the packets cannot take advantage of the larger aggregate bandwidth of the striping point to reduce transmission latency. This is called reduced apparent bandwidth.

The simulation study discussed in the Appendix illustrates both the head-of-line blocking as well as the reduced apparent bandwidth issues in ATM cell striped and IP packet striped systems.

5.2.4 TCP striping at the Application Layer

Striping can also be performed at the application layer across multiple TCP stripes. Because TCP is running over each stripe, the stripes provides a reliable transport mechanism for application layer PDUs. Any loss or corruption experienced by the data during transit will be corrected before the data reaches the inverse striping point. The primary impact of loss or corruption will be an increased skew between the stripes since TCP corrects these conditions through retransmission. Thus, in systems striped at this layer, the dynamic components of skew, resulting from loss or corruption induced retransmission, will almost certainly be the dominant source of skew between the stripes.

Since stripe QOS may vary greatly, a simple RR style striping algorithm may result in very poor performance as the overall striped system performance may be seriously degraded by a single poorly performing stripe. More advanced striping algorithms may be able to take into account the differing qualities of service between the stripes to optimize overall performance. Striping at the application layer suffers from the same stripe utilization problems which were described in the preceding section.

Since the layer has the most knowledge about the characteristics of the data which are being transferred across a network, application layer striping may be able to exploit this knowledge within the striping algorithm. It may be possible to partition the stripes and classes of data being generated by the application into groups and then provide an appropriate striping algorithms for each group.

6 Conclusion

This paper has established a framework for the comparison of striping at a range of layers in a typical high performance protocol stack. A precise set of terminology has been presented for describing striping and distinguishing it from other related architectural techniques. The evaluation of striping provides a collection of insights for determining the most appropriate layer to perform network striping for a given environment and workload.

- Striping at lower layers typically leads to greater striping point utilization since there is a higher apparent bandwidth, as shown in the Appendix.

- Striping at higher layers typically leads to less head-of-line blocking, also shown in the Appendix.
- At Gbps rates with current technology, low level striping can only be implemented in hardware since the granularity of the striping units is too fine for software to control.
- Higher level striping is typically implemented in software since it must be embedded within an already existing software system.

In environments where ordering must be preserved to maintain transparency, the following observations apply.

- Static skew between stripes can be solved by transmission timing at the striping point or by buffering at the inverse striping point.
- Dynamic skew can only be solved by sequence numbers in the general case.
- Sequence numbering is not a practical mechanism for ensuring ordering of byte striped systems.

Appendix: ATM Cell vs IP Packet Striping – A Simulation Study

To illustrate the tradeoffs between striping at the ATM cell and IP packet level, a trace driven simulation study has been performed [16]. This appendix is not intended to be an exhaustive study of striping; rather, it is intended to illustrate the different properties of ATM cell and IP packet striping.

Simulator

The simulator used for this study was written in the C programming language specifically to conduct an analysis of various striping configurations. The simulation is capable of reading a trace file containing a header which is used to configure the simulator for a particular striping scenario. The following parameters are read from the trace file header:[3]

Cell Time $= T \ \frac{Seconds}{ATMCell}$
Number of IP packet stripes $= P$
Number of ATM Cell stripes $= C$

The simulator then proceeds to read the trace file and perform the requested simulation at a resolution of one ATM cell time. The following time stamps are recorded for each packet processed: time queued, time striping service begins, and finally, the time striping service is completed. Other statistics collected include: stripe utilization and average/peak buffer utilizations. Figure 9 illustrates the striping configuration. It is assumed that unlimited buffer space is available.

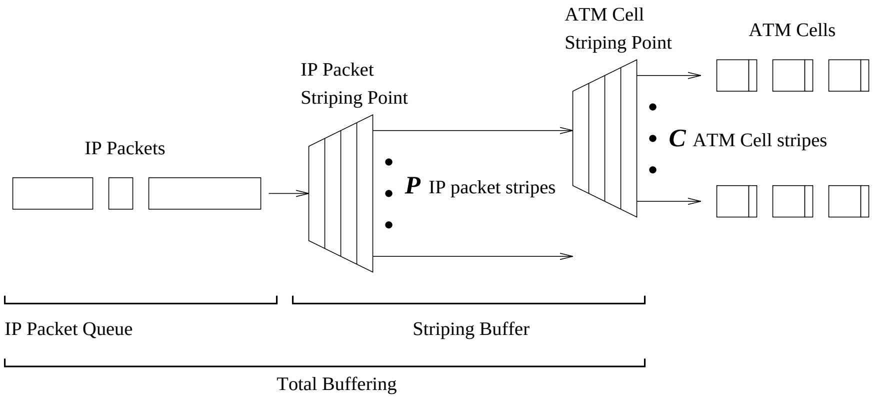


Figure 9: Simulated Striping Configuration

The following parameters can be derived from the values read from the trace file header:

ATM stripe bandwidth $= B = 8 * 53 * \frac{1}{T}$ bps
Aggregate bandwidth of striping point $= A = P * C * B$ bps

Four striping configurations of equal aggregate bandwidth are examined during each simulation. These configurations are referred to by the value of P, the number of IP packet stripes. For all four, $P * C = 8$. Pure IP packet striping is performed when $P = 8$ while pure ATM cell striping is performed when $P = 1$. When $P = 2$ or $P = 4$ a hybrid of ATM cell and IP packet striping is performed.

[3]For the purpose of this simulation, it is assumed that all IP packets stripes are of equal bandwidth and that all ATM Cell stripes are of equal bandwidth.

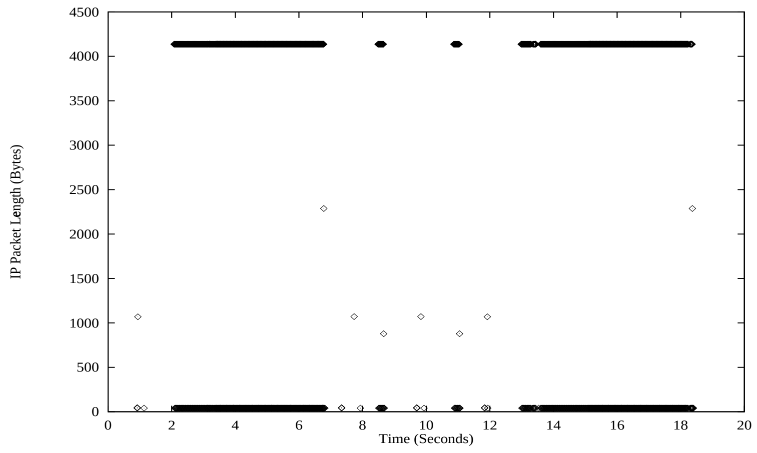


Figure 10: Traffic Trace

The effects of ATM cell padding are also examined. ATM cell padding is useful when performing ATM cell striping since it ensures that the first cell of any data unit being striped will always be located on the same stripe. This simplification can reduce the complexity of an ATM cell striping mechanism since the stripe which will contain the first cell of a data unit does not have to be calculated or tracked.

Packet Trace

This study used a trace of the network traffic generated by a typical World Wide Web (WWW) session driven by the Mosaic WWW viewing application. Such a session involves the retrieval of numerous large data, audio, video, and still images from remote, networked servers. This application is characteristic of a class of emerging interactive applications which will have extremely large numbers of simultaneous users who will each require access to substantial amounts of network resources.

The traffic trace used for this study has been collected from a loop back interface located at the base of the AIX Version 3.2.5 TCP/IP protocol stack running on IBM RS/6000 workstations. For the generation of these traces, an IBM RS/6000 Model 580 has been used. The traces consist of a sequence of $\{nanosecond\ time\ stamp,\ packet\ length\}$ pairs which are acquired as the packets are passed to the loop back interface by the transmit side of the protocol stack.

For this trace, the average latency experienced by each IP packet and the maximum buffer requirements of the striping point were recorded over a range of striping point utilizations. The maximum buffer space required is the metric selected for buffer utilization since it is the amount of buffering needed to ensure loss free operation. Striping point utilization is defined to be the following:

Average bandwidth utilization over trace $= a$ bps
Striping point utilization $= U = \frac{a}{A}$ %

Inverse striping was not examined in this study since a meaningful examination requires an method of modeling the congestion, loss, and skew present in real networks.

Figure 10 shows the size distribution of packets versus time for the trace. This trace consists of about 18 seconds of interactive use of the Mosaic application. During this period, 5 file transfers totaling about 9 MB are performed. The bandwidth utilized averaged 4.3 Mbps. This trace clearly shows the bimodal distribution of IP packet size which is generated by TCP. Approximately 2/3 of the packets are at the MTU limit of 4K while most of the remaining packets are acknowledgments.

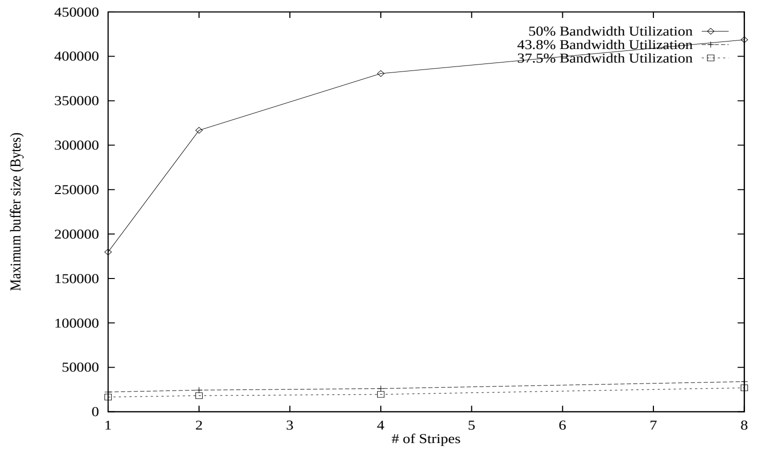


Figure 11: Buffering Requirements (No Padding)

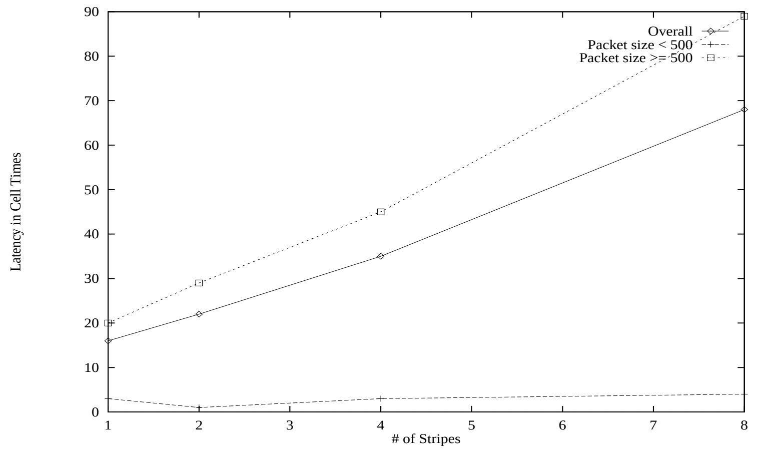


Figure 12: Latency 43.8% Utilization (No Padding)

No ATM Cell Padding

Figure 11 shows the impact of increasing the striping point utilization on the peak buffer size required to ensure that no data is dropped. A 6% increase in the striping point utilization resulted in a 7 to 15 fold increase in the amount of buffer space required. Note that the higher apparent bandwidth of the ATM cell striped configuration ($P = 1$) results in a significantly lower buffer utilization.

At 43.8% utilization, where input queuing is minimal, as the number of IP packet stripes increases, the total latency experienced by small[4] packets remains consistently low. Unfortunately, the decreasing stripe bandwidth greatly increases the transmission latency experienced by larger packets. This effect is shown in Figure 12.

Figure 13 shows the striping point total latency when the bandwidth utilization is set at 50%. Recall from Figure 11 that at 50% utilization, a significant amount of buffering was needed to queue IP packets at the striping point input. The effect of this queue can clearly be seen in this graph by the large latency experienced by both large and small IP packets. The reduction of head-of-line blocking due to the increased number of IP packet stripes is apparent in this case, as the total latency experienced by small packets does not grow as dramatically as that of large packets.

[4] Small packets are defined to be those whose size is less than 500 bytes.

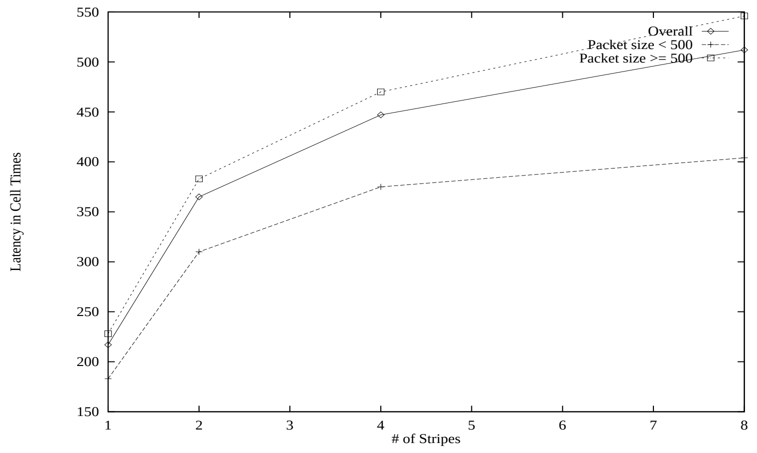


Figure 13: Latency 50% Utilization (No Padding)

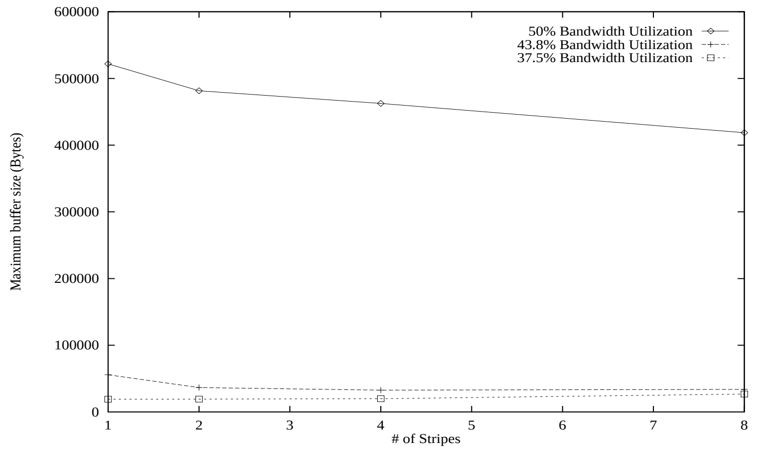


Figure 14: Buffering Requirements (Cell Padding)

ATM Cell Padding

The graph in Figure 14 demonstrates the additional bandwidth overhead contributed by ATM cell padding. As the number of IP packet stripes increases, the amount of ATM cell padding overhead decreases, resulting in the need for less buffer space.

When the striping point utilization is 43.8%, the effect of ATM cell padding is noticeable (Figure 15). For ATM cell striping, the total latency of both small and large packets is quite high due to the queue of IP packets created by the ATM cell padding. As the number of stripes increases, the total latency experienced by small traffic decreases continually since the multiple IP packet stripes reduce the effect of head-of-line blocking. The total latency experienced by large packets initially decreases as the effects of ATM cell padding are reduced, but then increases as the increased transmission latency caused by the reduced stripe bandwidth dominates.

The final graph, shown in Figure 16, shows the queuing caused by the relatively high stripe utilization of 50% and the overhead of ATM cell padding which dominates the total latency for both small and large packets.

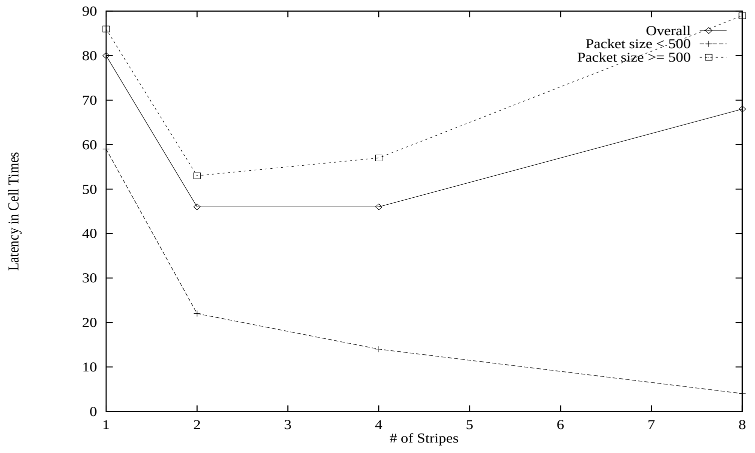


Figure 15: Latency 43.8% Utilization (Cell Padding)

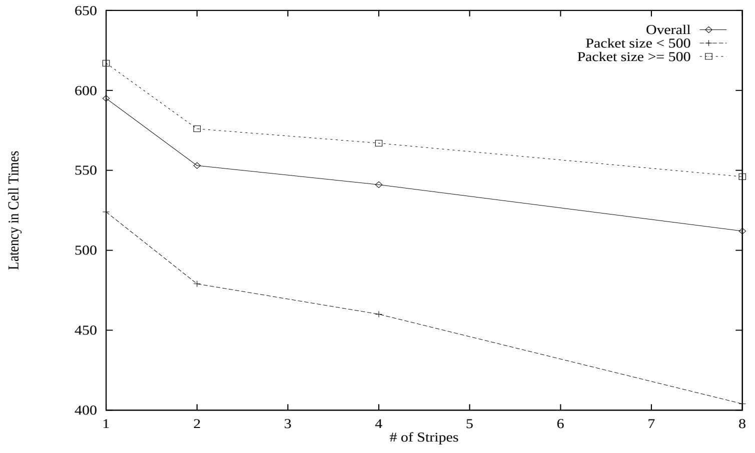


Figure 16: Latency 50% Utilization (Cell Padding)

Summary of Simulation Results

This simulation of ATM cell and IP packet striping has been able to illustrate some of the tradeoffs of striping at these levels in a typical broadband network striping stack. These tradeoffs are the following:

- Buffering requirements and queue induced latency can vary greatly with the level of striping point utilization.
- Cell padding can significantly impact both buffer requirements and latency when there are a large number of ATM cell stripes.
- For low utilization IP packet striping, the reduced bandwidth of each IP packet stripe significantly increase latency.
- Packet striping can be an effective mechanism for reducing the latency experienced by small packets.

References

[1] Bellcore, “Synchronous Optical Network (SONET) Transport Systems: Common Generic Criteria,” TR-NWT-000253, December, 1991, Issue 2.

[2] D. D. Clark and D. L. Tennenhouse, “Architectural Considerations for a New Generation of Protocols,” in *Proc. ACM SIGCOMM '90*, Philadelphia, PA, September, 1990.

[3] Computer Staff, “Gigabit Network Testbeds,” *IEEE Computer,* pp. 77-80, September, 1990.

[4] Bruce S. Davie, “The Architecture and Implementation of a High-Speed Host Interface,” *IEEE Journal on Selected Areas in Communications*, February, 1993.

[5] Peter Druschel, Larry Peterson, and Bruce Davie, “Experiences with a High-Speed Network Adaptor: A Software Prospective,” in *Proceedings, SIGCOMM '94,* London, England.

[6] David C. Feldmeier, “Multiplexing Issues in Communication System Design,” in *Proceedings, SIGCOMM '90,* Philadelphia, PA.

[7] Paul H. Fredette, “The Past, Present, and Future of Inverse Multiplexing,” *IEEE Communications Magazine,* pp. 42-46, April, 1994.

[8] J. Giacopelli, J. Hickey, W. Marcus, W. D. Sincoskie, and M. Littlewood, “Sunshine: A High-Performance Self-Routing Broadband Packet Switch Architecture,” *IEEE Journal on Selected Areas in Communications,* pp. 1289-1298, October, 1991.

[9] Mike Ismert, LCS, MIT, Personal Communications, 1994.

[10] Cesar Johnston, Presentation at CNRI Gigabit Testbed Workshop, June, 1993.

[11] Randy H. Katz, Garth A. Gibson, and David A. Patterson, “Disk System Architectures for High Performance Computing,” in *Proceedings of the IEEE*, Vol 77, No.12, December, 1989.

[12] Kenneth Salem and Hector Garcia-Molina, “Disk Striping,” *Proceedings, IEEE Data Engineering Conference,* Los Angles, CA, February, 1986.

[13] Wally St. John and David DuBois, “HIPPI-SONET Gateway,” CASA Gigabit Testbed Annual Report, pp. 47-52, 1993.

[14] Vasilios Theoharakis and Roch Guerin, “SONET OC-12 Interface for Variable Length Packets” in *Proceedings, Second International Conference on Computer Communications and Networks,* San Diego, CA, June 28-30, 1993.

[15] C. Brendan S. Traw and Jonathan M. Smith, "Hardware/Software Organization of a High-Performance ATM Host Interface," *IEEE Journal on Selected Areas in Communications,* February, 1993.

[16] C. Brendan S. Traw, "Applying Architectural Parallelism in High Performance Network Subsystems," Doctoral Dissertation, Computer and Information Science Department, University of Pennsylvania, 1995.