# ServerNet and ATM Interconnects: Comparison for Compressed Video Transmission

Ashfaq Hossain, Sung-Mo Kang, and Robert Horst

*Abstract:* **We have developed fully functional Video Server and Client applications which can transmit, receive, decompress and display compressed video over various networks. Our video transport allows dynamic rate control feedback, loss detection, and repair requests from Clients to the Server. Our experiments show how feedback-before-degradation scheme for rate adaptation maintains good display frame-rate for video playback. We show how the playback degradation (reduction in display frame-rate) occurs and what happens if corrective measures are not taken to improve the situation. The degradation is attributed to the increased internal kernel buffering which consumes scarce CPU resources. We demonstrate with our experimental results that ServerNet, with improved hardware delivery guarantees, can significantly reduce host CPU resource consumption while serving video streams. We present the maximum number of streams which can be served for each of ATM and ServerNet interconnects. The appropriate user-level packet sizes for the video server are also determined for each case.**

## I. INTRODUCTION

We have investigated distribution of pre-compressed digital video over ATM (Asynchronous Transfer Mode) and ServerNet (from Tandem Computers Inc.). We present experimental results for video throughput and the corresponding CPU utilization for different end-host platforms which implement these networks. We identify the strengths and weaknesses of such networks by observing the end-host behaviors. All the experimental runs reported in this work are based on fully functional Video Server (VS) and Video Client (VC) applications developed in the iPOINT (Illinois Pulsar-based Optical Interconnect) testbed with a fully functional 800Mbps ATM switch [1], [2]. We have reported the operation of our VS and VC applications in more detail in an earlier paper [3].

Our Video Server (VS) is a multimedia storage server. The transmission rate from the VS can be varied in response to adaptation feedback signals from the Video Clients (VC). VS also maintains a finite sequence of previously transmitted video frames to service repair requests from the VCs arising from packet loss due to network congestion. Our VS maintains unicast (point-to-point) and multicast (point-to-multipoint) connections with the VCs and can serve Motion-JPEG (MJPEG) and MPEG-1 video.

The VC performs front-end operations such as reception, decompression, display and control of the multimedia data. The control operation is done by sending rate adjustment feedback and repair-request messages to the server after a video packet loss has been detected. For multicast sessions, our VC implements a *selective* repair-request algorithm. That algorithm uses Round Trip Time (RTT) to the VS, client's consumption rate of video packets and fullness of its receiving buffer to selectively send repair requests to the VS for lost packets.

This paper presents our research in two major veins: first, the implementation of our VS and VC applications along with rate-control feedbacks and network loss handling mechanisms; second, the performance study for our video service using two major machine interconnects (ATM and ServerNet) using 60MHz SparcStation20s (SS20) and 100 MHz Pentium (P5) systems as video servers. Our study compares CPU utilization and corresponding video throughput of these platforms, running Solaris 2.5 and WindowsNT 3.51 as operating systems, respectively. We show how these network interfaces affect the behavior of these hosts working as video servers and thereby the quality of served video.

## II. INTERWORKING OF iPOINT VIDEO SERVER AND CLIENT APPLICATIONS

The iPOINT VS and VC implementations [3] draw on UDP/IP socket interface and Fore System's ATM APIs (Application Programmer's Interface). The socket interface is used for IP-over-ATM experiments, both for Solaris and Windows NT platforms. For native-ATM transmission, only ATM API is used.

The VS listens to a well-known port or VPI/VCI for incoming client requests (Fig. 1). On receiving a request, the server creates a Compressed Image Sequence (CIS) for a specific type of stream (MJPEG or MPEG-1), `mmaps` the video file to the user memory, then gradually reads an integral number of frames into the CIS. These frames are then transferred as IP packets or AAL5 frames. Any rate-adaptation (e.g. `SLOWDOWN`, `SPEEDUP`) or repair-request (`RETRANSMIT`) feedback messages from the VCs are handled by the VS with high priority. The packet sizes are limited to 9180 *bytes* for AAL5 frames (maximum MTU) or to the size set by the `setsocketopt()` call for IP transmissions (maximum 64KBytes). Several other methods for resilient transmission of compressed video frames have been reported[4]. Although such modifications can be built

Manuscript received July 29, 1998; approved for publication by John Limb, Division III Editor, May 7, 1999.

A. Hossain is with Bell Laboratories, Lucent Technologies, 600 Mountain Avenue, Murray Hill, NJ 07922, USA, e-mail: ashfaq@lucent.com.

S.-M. Kang is with Dept. of Electrical & Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, e-mail: s-kang@uiuc.edu.

R. Horst is with Tandem Computers Inc., Cupertino, California, USA, e-mail: bob.horst@tandem.com.

1229-2370/99/$10.00 

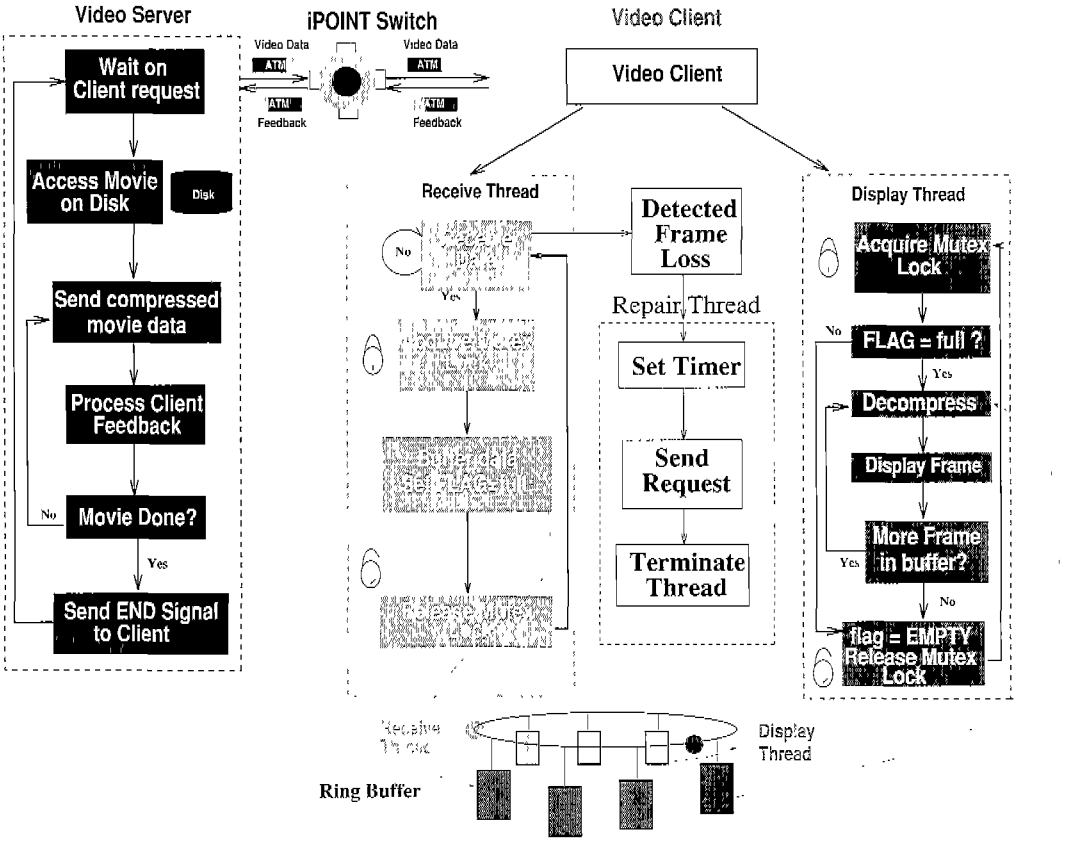



Fig. 1. Schematic representation of video server and client applications.

into our VS and VC applications, currently the frames transmitted from our server are compressed streams generated by standard MJPEG or MPEG encoders. This reduces extra processing on the VC's end and any network loss is handled by retransmission requests.

The VC implements a ring-buffer on its end to receive the compressed frames (Fig. 1) – which can be sent by the VS as an entire video frame in a single packet or can be broken up into smaller parts (at the application layer). The ring-buffer consists of $n$ elements.

Each element can hold up to 2 compressed MJPEG frames (approximately, 10KBytes/frame for a frame size of 320 × 240 *pixels*) and a single I-P-B frame combination for an MPEG-1 stream[1]. For the current implementation, the total memory for the ring buffer is about $1MByte$. The `Receive()` and `Decompress()` threads chase each other in the ring. The application buffer space is full when the `Receive()` thread is just behind the `Decompress()` thread. The VC is responsible for detecting lost portions of a transmitted video frame. It identifies any delayed, duplicated and retransmitted datagrams and *stitches* the video data in appropriate locations of the ring buffer. The VC considers a video frame to be incomplete if it starts receiving fragments of video frame $i+1$ without receiving all the fragments of frame $i$. When such a situation is detected, the client dynamically creates a *RepairRequest()* thread which is responsible for sending a bit array to the server (`RETRANSMIT` message). This bit array indicates which portions of the entire video frame has not been received (if frames are being transmitted in parts). When the retransmitted packets are received from the VS in response to `RETRANSMIT` messages, they are identified by the packet header field and placed in appropriate positions in the ring-buffer. For our experiments over the ATM environment, we have not considered out-of-order arrival scenarios.

Our VC application is near-real-time in the sense that the ring buffer is initially allowed to store a few frames before the `Decompress()` thread starts consuming them. This is done to allow for a `RETRANSMIT` message to be sent to the VS (for a lost packet) and to receive the repair before the `Decompress()` thread is ready to consume it. Therefore, our VC is expected to have some estimate of the round-trip-time between itself and the VS to correspond to the number of frames to be initially buffered in the ring.

[1]Our MPEG-1 stream is compressed in IPBIPBIPB... format; other encoding formats have not been considered for this work.

## III. PERFORMANCE RESULTS ON THE ATM TESTBED

In this section, we report the performance of our VS and VC using the iPOINT testbed, which consists of an 800Mbps, 5-port ATM switch [1]. We have used a SparcStation20 (60MHz, 1 CPU) and a SparcStation10 (25MHz, 1 CPU) running Solaris 2.5, as our VSs. SS20s have been exclusively used for VCs. For our VC application, decompression of the received video is software-based, and the SS20 is the obvious choice over SS10

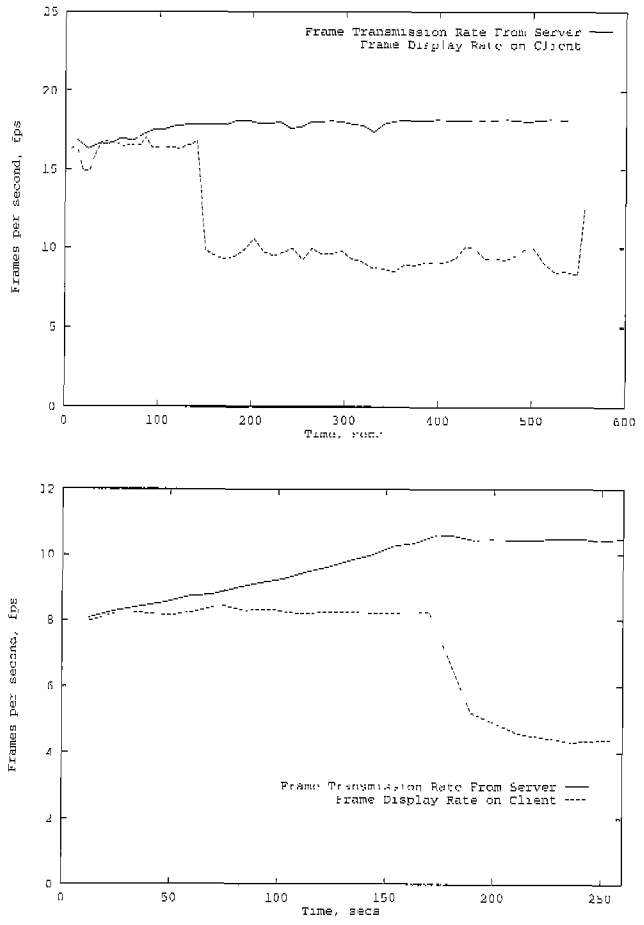



Fig. 2. Dynamic operation of VS and VC applications for IP-over-ATM on iPOINT testbed (MJPEG - top, MPEG-1 - bottom) – *No Feedback.*

for a higher display frame rate. Our host machines (SS20, SS10) have been equipped with 100Mbps TAXI ATM cards (SBA-200E) from Fore Systems. Each of these host machines have been equipped with 32MB Random Access Memory.

### *A. Dynamic Operation*

Fig. 2 shows the transmission rates of the VS and the display frame rates ($fps$) on the VC for experimental runs in the IP-over-ATM (unicast) mode. For this experiment, the VC does not send any rate-control feedback to the VS – the mechanism of sending feedback from the VCs to the VS is described later in this section. We observe the following from Fig. 2: the transmission frame rates of the VS are 16-18fps for MJPEG and 8-10 fps for MPEG-1; VC maintains a decompression/display rate which follows VS transmission frame rates reasonably up to a certain point ( 130 *secs* for MJPEG and 160 *secs* for MPEG-1). After these points, performance of VC degrades sharply (from $17fps$ to $10fps$) and the degraded quality of playback does not improve. The CPU utilization is 100% for the VCs (due to software decompression) during these experiments.

This phenomenon can be explained as follows: on the VC end, the application space ring buffer gradually becomes full because the rate at which frames are decompressed and displayed is slightly lower than the rate at which compressed frames arrive. As the ring-buffer becomes full, the OS kernel starts to buffer arriving packets in the finite kernel memory. This results in increased OS activity on the CPU. As a result, the CPU availability decreases for the `decompress()` and `display()` threads of the VC application, resulting in deterioration of VC's decompression/display rate. After the kernel buffer is filled up, arriving packets are dropped by OS, resulting in missing frames and jerkiness in the playback. For MPEG-1, a loss of frame is more detrimental than MJPEG because a dropped I-frame causes the dependent P and B frames to be non-decompressable. When the video packets continue to arrive even after the VC's performance is degraded, the aforementioned (extra) OS activity never decreases and the VC application continues to suffer from insufficient CPU resource. The fact that the CPU load increases due to kernel buffering is manifested by an increase of the VC's display frame rate, when the server stops sending packets (a sharp increase in the VC's display rate is observed after 540secs in Fig. 2).

In order to avoid the situation which arises due to application space buffer overflow, the VCs send feedback messages to the VS to control the transmission rate dynamically. We consider two approaches to feedback: feedback *after* the ring buffer becomes full and performance degrades (kernel feedback) and *before* the ring buffer becomes full (when the buffer fullness reaches a certain high water mark). These results have been presented in Figs. 3 and 4, respectively. The types of *feedback* messages `SLOWDOWN`, `SPEEDUP`, `NORMAL`, `RETRANSMIT` which can be sent from VCs to VS are `SLOWDOWN`, `SPEEDUP`, `NORMAL`, `RETRANSMIT`. The VS reduces the rate of transmission on receiving a `SLOWDOWN` message by increasing the time delay between the network transmits. `SPEEDUP` message from the VC results in the VS increasing the rate of frame transmissions by decreasing the time delay between network transmits. Receiving a `NORMAL` message results in the VS resuming the initial rate of frame transmissions. A `RETRANSMIT` message is a repair request message which requires the VS to retransmit a lost frame. More details in implementing this feedback mechanism can be found in [3], [5].

We observe in Fig. 3 that the client's situation improves due to reduced kernel buffering as the server decreases the rate at which packets are sent. After the client recovers and sends a `NORMAL` message to the server, the VS resumes its previous sending rate. Again, this causes the client's application space ring buffer to become full causing kernel buffering, resulting in increased CPU load and decreased decompression/display frame rate. This cycle is repeated continuously. The resulting playback causes reasonable playout jitter due to uneven time intervals between displayed frames.

The large playback jitter which arises in Fig. 3 can be reduced significantly by sending a feedback from VC to VS before the ring buffer becomes completely full. This avoids kernel buffering and keeps the CPU available for maintaining a smooth playback rate. This is shown in Fig. 4.

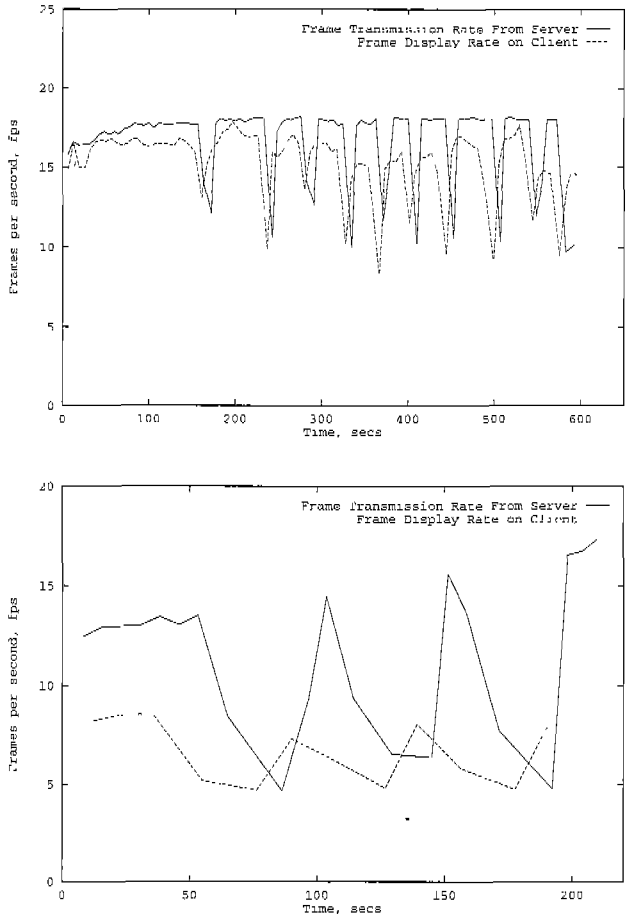



Fig. 3. Dynamic operation of VS and VC applications for IP-over-ATM on iPOINT testbed (MJPEG-top and MPEG-1 -bottom) – *With Feedback After Degradation.*

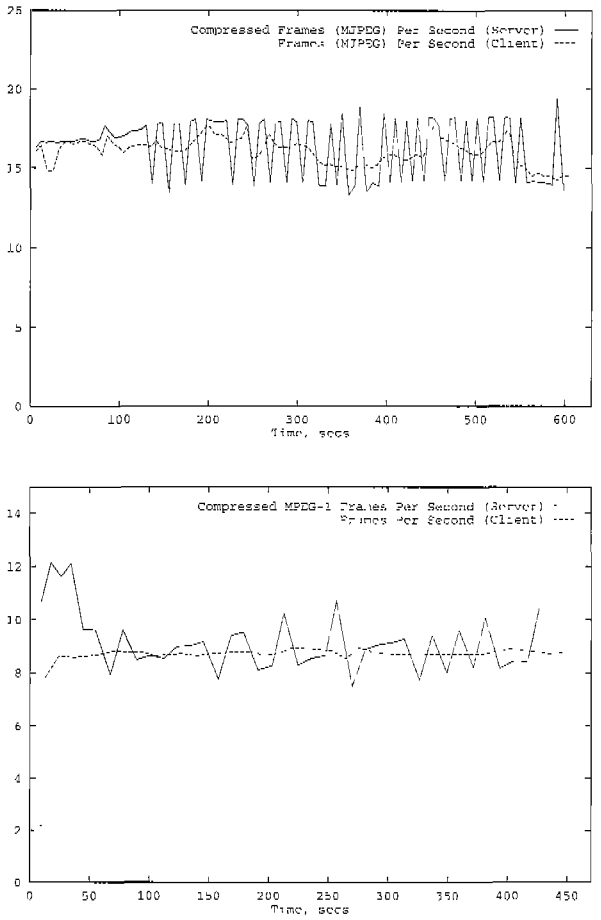



Fig. 4. Dynamic operation of VS and VC applications for IP-over-ATM on iPOINT testbed (MJPEG-top and MPEG-1 -bottom) – *With Feedback Before Degradation.*

### *B. Server Saturation For ATM Service*

We now determine how many such video streams can be served by VS application using the SS20 and SS10 as hardware platforms. Fig. 5-top gives the throughput of the video server with increasing number of MJPEG threads. It also shows the fraction of under-QoS transmission epochs, which are the time windows during which a particular thread is served with less than the desired $18fps$ rate. We choose this particular frame rate as the desired QoS rate because this is the rate our VC can maintain with software decompression (as shown in Fig. 4). Saturation occurs for about 20 threads on a SS10/30. For a SS20/60, the machine can serve 47 threads before it gets saturated. For these experiments, we consider a UDP packet size of *one* JPEG frame (10KBytes approximately for a 320 × 240 frame size) per send. For a $10KByte$ UDP packet size, the maximum throughput that can be achieved with SBA-200 in a SS20/60 is 80Mbps (approx), as seen in Fig. 8. With each video thread sending $18fps$ (also $18packets/sec$ per thread for this case), each packet being $10KBytes$ (approx), it is expected that an SS20/60 will be able to serve about 55 such MJPEG threads. Taking into consideration the overhead due to thread manipulation and the mutual exclusion locks we used for updating some global variables for our measurements, we find that saturation on a SS20 occurs at around 47 threads (instead of 55) for a total delivered throughput of $68Mbps$ (Fig. 5-top). Sending multiple video frames as a single UDP packet (2 MJPEG frames as a single packet is around $20KBytes$) will not increase the aggregate throughput (or number of QoS-satisfied threads) as the throughput saturation for a SS20 occurs at around 10KByte packets on IP-over-ATM. This is observed in Fig. 8. The numbers of streams for which saturation occurs in these experiments represent the best case as throughput loss due to retransmission and disk access penalty are not considered.

We will compare these video saturation points with those we obtain for ServerNet in the next section.

## IV. SERVERNET INTERCONNECT

Having reported the performance of our video service on our ATM testbed, we next compare it with the performance of Tandem Computer's *ServerNet* interconnect. ServerNet is a new System Area Network (SAN) that has been designed specifically as a reliable, high speed connection among processors and I/O devices in a cluster[6], [7]. ServerNet is a wormhole-routed multistage network formed by point-to-point connections to six-port router ASICs (Application Specific Inte-

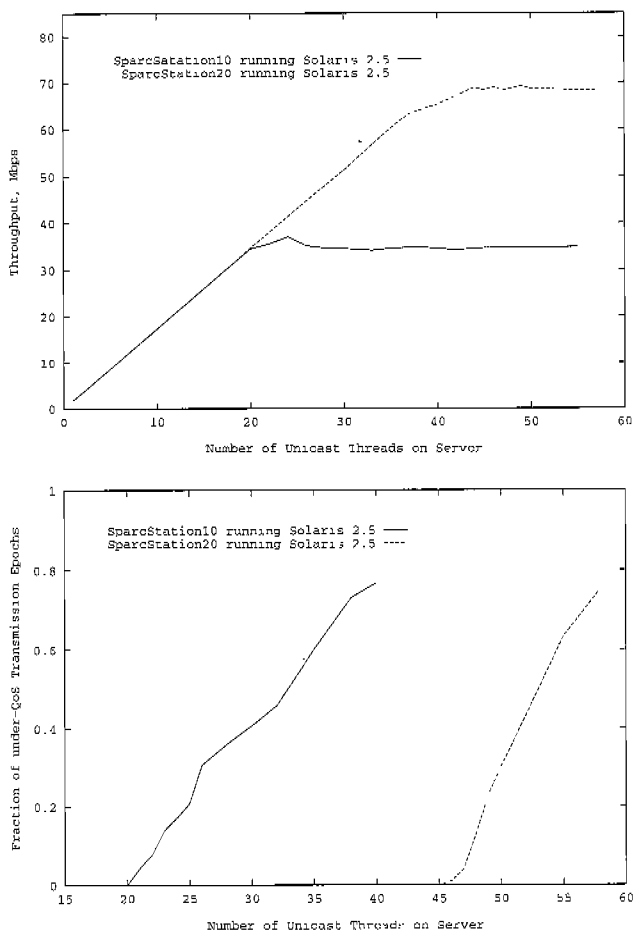



Fig. 5. Server saturation points for IP-over-ATM MJPEG video transfer.

grated Circuits). Reliability is assured by self-checking routers and network adapters, support for dual fabrics, and link-level flow control to *eliminate* packet loss on heavily loaded networks. This interconnect provides high machine-to-machine data-communication throughput while maintaining low CPU utilization on the end-hosts.

A ServerNet network consisting of multiple hosts (high-performance PCs and/or Workstations with ServerNet interface cards in the PCI slots) interconnected by ServerNet Routers is shown in Fig. 6. This figure specifically presents how video data can be transferred over such a network.

Our eventual goal is to compare and analyze the performance of ATM and ServerNet as media for video transport – such comparison is based on the behavior demonstrated by the end-hosts which implement these interconnects. We have reported some of the initial results of our experimentation in an earlier paper [8]. In this section, we report, in further detail, performance results of ServerNet interconnect using 100MHz P5 (Compaq's Proliant 1500) systems as end-hosts with 16MB Random Access Memory. We identify the attractive features of such an interconnect for compressed video distribution and compare its performance numbers with ATM.

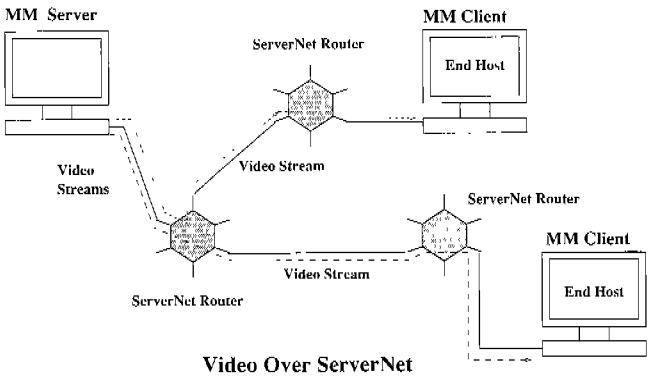



Fig. 6. ServerNet system area network for video transport.

### A. The Operation Of ServerNet

In order to transfer $N$ *bytes* of data from machine A to machine B, the *Interface Layer* of the ServerNet protocol writes a chain of BTE (Block Transfer Engine) descriptors in the main memory (MM) of machine A (Fig. 7). Each BTE descriptor is responsible for 4 $KBytes$ (max) of data (for $N$ *bytes* of total transfer, $\lceil \frac{N}{4K} \rceil$ BTE descriptors are created). Each descriptor contains the address of these $4KByte$ blocks which need to be transmitted, destination node ID, read/write access permission information, flow-control information and the address of the next BTE descriptor (thus establishing the chain). The ServerNet driver initializes the transfer by writing the MM address of the first BTE descriptor into a register on the ServerNet card. From this point onwards, the host CPU is relieved of any activity related to the transfer of $N$ *bytes* over ServerNet. Assuming that all $N$ *bytes* of data are in the MM, the ServerNet card reads in each $4KByte$ block (being pointed to by the current BTE) as $64Byte$ packets from the MM, across the PCI bus before it is transmitted over the link.

It is very important to realize that the $64byte$ packets are never lost in ServerNet due to network congestion – this is particularly interesting for video transmission. If congestion is detected, a BUSY command (similar to the SLOWDOWN feedback from the video client) is sent by the ServerNet hardware (either Router or end-host's interface card) to the source Network Interface Card (NIC) through the back channel. Congestion is indicated by the receiving FIFOs reaching a high water mark, which are located in the Routers and the NICs themselves. The sender reduces its sending rate after receiving this signal. Any packets in transit, or portions thereof, are accommodated in the remaining space of the FIFOs, thus eliminating loss due to congestion.

The only loss of $64Byte$ packets that can occur in ServerNet is due to link failure. If a positive acknowledgement is not received by the sender NIC within a time interval, the packet is considered to be lost. For tolerating this type of fault, two separate link fabrics are used (marked $X$ and $Y$ in Fig. 7), which provide alternate paths to the destination. The driver uses the second path to the destination to complete transfers that encounter errors on the other fabric.

*Flow Control* in ServerNet has a two-fold responsibility: (1) to maintain a TCP-like window (end-to-end control) for burst

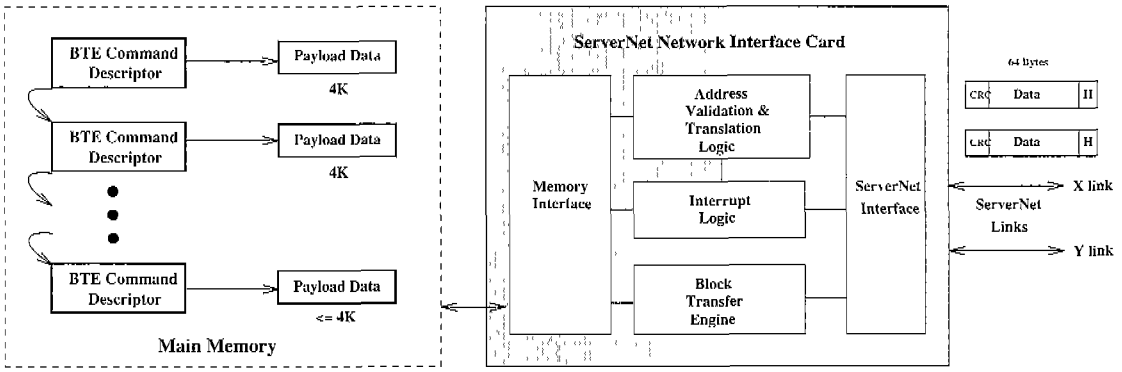



Fig. 7. Interaction between ServerNet PCI card, BTE descriptors and associated data in main memory.

transfers – the maximum number of outstanding transfers is limited by a value in the BTE descriptor field; if there is network congestion or a busy receiver, the sender will not generate a new transfer until the first acknowledgement is received, (2) to *guarantee* delivery of $64byte$ packets by eliminating losses due to network congestion and by re-transmitting unacknowledged packets (due to link failure) over the alternate link fabric.

The Address Validation and Translation (AVT) Logic (Fig. 7) is responsible for validating addresses of incoming packets and access permission to addressed pages. The Interrupt Logic generates an interrupt when an $N$ $byte$ transfer is completed or for other abnormal situations. More details can be found in [6], [7].

The underlying physical communication is currently established with copper cables, having a 9-bit channel in each direction. The driver clock frequency is 50MHz. Taking into consideration the header overhead for each 64 $byte$ packet, the expected maximum raw data throughput approaches 40 $MBytes/sec$.

### B. ServerNet As A Video Transport Interconnect

In addition to providing high throughput at low CPU utilization, ServerNet ensures guaranteed data delivery with protocols implemented in hardware as well as software. Reliable guaranteed delivery of VBR encoded video using software-only protocols have been reported [9]. The motivation for considering ServerNet as a medium for video transfer can be attributed to the following features:

- High throughput – More video streams can be served (fatter pipe), provided server CPU resources are available to push data out of the NIC.
- Low CPU utilization – More video service requests can be handled by the VS (as long as the communication bandwidth is available and PCI bus is not saturated).
- Hardware flow control – *Eliminates* packet loss in the intermediate routers; flow-control symbols stop transmission before receiving FIFO buffers overflow. This results in no retransmissions due to network congestion and consequently no extra load on the host CPU. Key video frames (I and P) can be sent with hardware acknowledgements and less important frames (B frames) can be sent in an unacknowledged mode.
- Fault tolerance – The ServerNet NIC can connect to two independent switching fabrics to provide alternate paths; the driver uses the second path to the destination to complete transfers (guaranteed) that encounter errors on the other fabric. I and P frames can be sent in this guaranteed mode of transfer.

### C. ServerNet Video Performance

In order to determine the maximum throughput and the number of video streams which can be served over ServerNet, we used 100MHz Pentium (P5) end-hosts running WindowsNT 3.51. We installed the ServerNet drivers written for NT and equipped each of the machines with prototype ServerNet NICs. The maximum throughput and CPU utilization for ServerNet are shown in Figs. 8 and 9. The corresponding IP-over-ATM results, as obtained from the SS20s are also shown in these figures. Even though the ServerNet measurement is performed on the higher clock rate of 100MHz P5, some benchmark tests put the SS20/60 machine at a better performance due to its dual issue (superscalar) instructions per cycle (Cycles/Instruction (CPI) can be less than 1). In regards to the SPEC numbers for these two processors, the P5 performance is between 0.9× and 2.0× that of the SS20[2]. We have also presented results from experimental runs for Fast Ethernet (3Com's 3C905-TX card) using NT's TCP/IP protocol stack on the 100MHz P5 machines. The Ethernet results are just for comparison purposes and we have not considered them for video transfer.

The maximum throughput for ServerNet is observed to be $20.3MBytes/sec$ for a packet size of $64KBytes$. The P5 CPU utilization corresponding to this maximum throughput is 9% (at 64KByte packet size). Further design optimizations on the ServerNet NIC are underway to achieve the expected $40MBytes/sec$ mark. A major design improvement has been to cache the BTE descriptor (each BTE descriptor is responsible for 4KByte data) in a register on the NIC itself, rather than fetching it across the PCI bus for every $64Byte$ packet transfer.

To determine the maximum number of video streams that can be served over ServerNet before the VS CPU saturates, we have performed similar experiments (as for ATM) on these P5 machine pairs (one P5 host acting as the VS and the other as a VC), by increasing the number of served video streams and by observing the corresponding aggregate throughput and frame rate of each stream delivered from the VS. The results are shown in Fig. 10 for MJPEG streams. For this experiment,

[2]These data were obtained from http://infopad.EECS.Berkeley.edu/CIC/

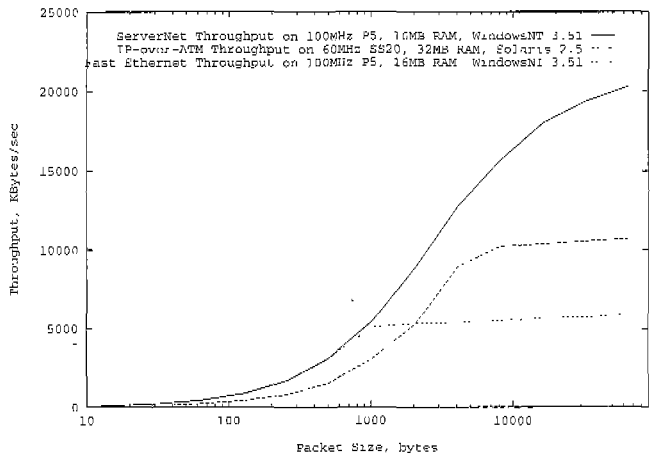



Fig. 8. Throughput comparison for ServerNet, IP-over-ATM and Fast Ethernet.

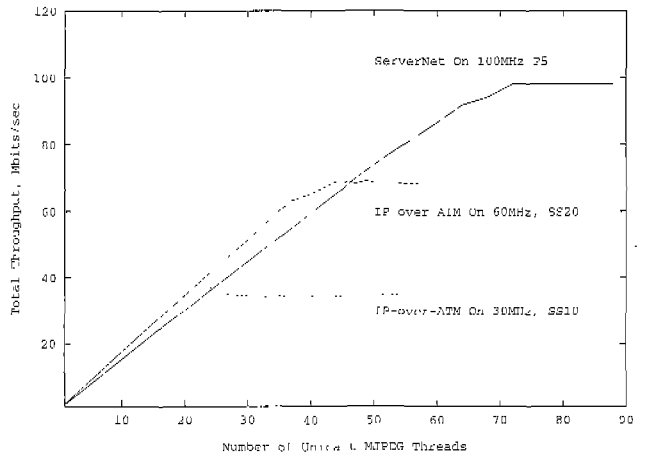



Fig. 10. Server saturation for MJPEG threads over ServerNet and IP-over-ATM.

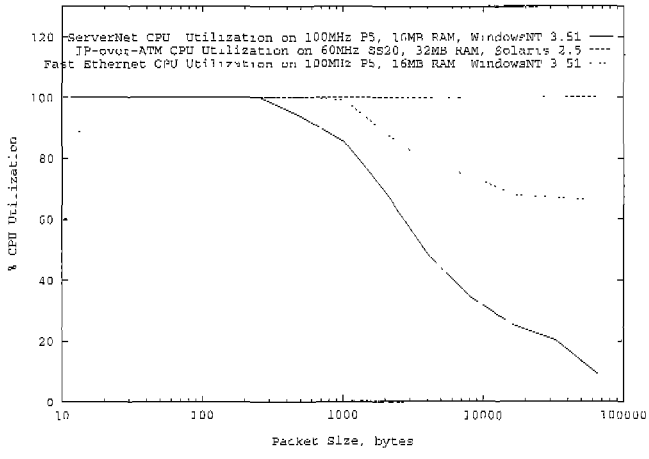



Fig. 9. CPU utilization comparison for ServerNet, IP-over-ATM and Fast Ethernet.

we send 2 compressed frames in a single send over ServerNet (20KBytes approximately). The motivation for sending multiple compressed video frames in a single send is to increase the sending packet size, thereby taking advantage of the higher throughput that a larger packet size provides in ServerNet. As shown in Fig. 8, the maximum attainable throughput for $20KByte$ packet transfers is about $17MBps$ or $136Mbps$. For the video experiment, we have achieved a maximum throughput of $100Mbps$ with 70 threads (Fig. 10) before the VS saturates and cannot serve more streams maintaining the desired frame-rate. Again, this difference can be attributed to overheads due to mutual exclusion lock handling and to the manipulation of a higher number of threads.

## V. SERVER AND ATM PERFORMANCE COMPARISON

We are now in a position to compare the performance for serving video streams as well as raw throughput and CPU utilization of these interconnects. First, we need to note the following:

- Our ATM experiments use the SS20 (1 superscalar CPU, 60MHz, dual instruction issue per cycle) and SS10 (1 CPU, 30MHz) platforms running Solaris 2.5,
- ServerNet results use Compaq's Proliant 1500 as end-hosts (1 P5 CPU/per-host, 100MHz) running WindowsNT 3.51,
- Our SS workstations use Fore System's SBA200 (SBus-based) NICs (100Mbps TAXI) – the drivers were also supplied by Fore,
- Both ServerNet NIC and the corresponding NT driver are early versions – future optimizations are expected.

Figs. 8 and 9 compare the IP-over-ATM, ServerNet and Fast Ethernet throughputs and corresponding CPU utilizations. The host platforms and operating systems used in the experiments are also indicated in these figures. Throughput of ServerNet is higher than that of ATM for all packet sizes that we have considered. Maximum throughputs are found to be $20.3MBps$, $10.8MBps$ and $5.9MBps$, for ServerNet, ATM and Fast Ethernet, respectively. In terms of CPU utilization (Fig. 9), ServerNet uses 9% of the CPU as it delivers maximum throughput. As the packet size increases, this utilization decreases to 65% for Fast Ethernet but remains at 100% for ATM (Fig. 9). The packet transfer overhead can be divided into two groups[10]-[12]: those that scale per byte (checksum, data copy), and those that are per packet (system overhead, protocol processing, interrupts). Even though the total overhead for larger packets is higher than for smaller packets, CPU utilization decreases for increasing packet sizes (as observed for ServerNet and Fast Ethernet NICs). This available CPU resource can be used for other related or non-related computing activity.

For ATM on the SS20, the CPU utilization is 100% for *all* packet sizes.

It is important to realize that at the saturated service points, the SS20 with ATM is entirely CPU bound – it has no CPU resource available to perform any other functionality. On the other hand, the P5 implementing ServerNet has about 90% of its CPU resource available – this can be used for additional data transfers across other NICs fitted in other slots provided the PCI bus saturation does not occur.

ServerNet has much lower CPU overhead than ATM (and also Fast EtherNet) because the hardware directly implements guar-

anteed, in-order delivery of packets even when the network is heavily loaded. There is no need for software to detect loss of packets, or to compute checksums. The ServerNet hardware does address translation to automatically perform scatter-gather operation if the physical pages are not consecutive. Once the BTE chain is initiated, the transfer is controlled completely by the hardware, and no more software execution cycles are consumed by either the client or the server until the transfer is completed. In effect, this means that the ServerNet hardware implements levels 1-4 of the OSI stack, while the ATM hardware provides only an unreliable datagram service and requires levels 3 and 4 to be performed by the software.

Fig. 10 compares the maximum number of MJPEG threads which can be served over ServerNet and ATM interconnects. We consider that each MJPEG thread delivers video at $18fps$ where each frame is $320 \times 240$ $pixels$). As we have explained before, the total throughput from all the video threads reasonably matches the maximum deliverable throughput from these NICs at the packet sizes used (except for thread overheads and global variable manipulation through exclusion locks). The number of MJPEG threads that can be served for the hosts indicated are 70, 47 and 20 over ServerNet and ATM (SS20, SS10), respectively. It should be noted that sending 2 MJPEG frames per send over ServerNet, results in higher throughput per thread (as well as overall delivered throughput), whereas, for IP-over-ATM, sending more than 1 frame per send does not result in increased throughput (saturation occurs around $10KByte$ packets for IP-over-ATM).

## VI. CONCLUSION

We have shown that throughput and QoS of video playback can be maintained by implementing a feedback-before-degradation scheme for rate adaptation. We have demonstrated that for our applications, how the playback degradation occurs and how corrective measures can improve the performance. Our experimental results indicate that degradation of video playback is mainly due to increased CPU activity for internal kernel buffering which occurs after the user-space ring buffer becomes full. This degradation in video playback can be prevented by sending appropriate feedback messages to the VS. If degradation has already occurred, the playback can only be improved by opening up space in the user-space ring.

We have determined that the maximum number of MJPEG video streams which can be served over our iPOINT testbed using ATM TAXI (100Mbps) cards on SS20s is 47 with a per stream frame rate of $18fps$. Each such frame is approximately 10KBytes. We have observed that increasing the sending packet size (at the user level) will not increase the number of served threads for IP-over-ATM transfers. We have also noted that for these ATM NICs SS20 CPU utilization remains at 100% for back-to-back sends, even when large packets are transferred.

We have compared the ATM performance of our video applications with that of ServerNet. We have observed from our comparative results that an interconnect (e.g. ServerNet) which provides improved hardware delivery guarantees, can reduce host CPU resource consumption while serving video streams. This available CPU resource, which is not present when we experiment with ATM NICs, can be used for serving a higher number of video streams provided the following conditions are true: (i) VS's PCI bus is not the bottleneck and (ii) communication bandwidth is available in the NIC to push more streams out to the VCs. ServerNet, implemented on 100MHz P5 systems, provides a maximum throughput of $20.3MBps$ (compared with $5.9MBps$ for Fast Ethernet on these machines) at a CPU utilization of only 9% (65% for Ethernet). The maximum number of MJPEG threads which can be served through ServerNet on these P5 machines is about 70 at approximately $20KByte$ packets per send. About 90% of the P5 CPU resource is free while the streams are served (compared with no CPU availability for ATM serving 47 threads on an SS20).

For video experiments over ServerNet, using a large packet size ($20KBytes$ vs $10KByte$ sends in ATM) is beneficial, as the throughput saturation in ServerNet occurs at about $64KByte$ packets. Since this interconnect provides delivery guarantees (using link-level congestion control, end-to-end flow control and redundant link fabric), we can also combine more video frames (say, 6 MJPEG frames) into a single packet of approximately $60KBytes$ – thereby achieving a higher number of total served streams. Such aspects can be investigated further.

## VII. ACKNOWLEDGMENTS

This research has been supported by the National Science Foundation Engineering Research Center grant ECD 89-43166 and the Advance Research Program Agency (ARPA) grant for Center for Optoelectronic Science and Technology (COST) grant MDA 972-94-1-0004. We would like to thank Prof. Klara Nahrstedt of Dept. of Computer Science of the University of Illinois at Urbana-Champaign and Charles Gasman of Bell Laboratories (Lucent Technologies), for their valuable comments on this paper. Thanks are also due to Gary Campbell and Krishnakumar Karoor of Tandem Computers Inc. for their inputs in ServerNet experimentations.

## REFERENCES

[1] J. Lockwood, H. Duan, J. Morikuni, S. M. Kang, S. Akkineni, and R. Campbell, "Scalable optoelectronics ATM networks: The iPOINT fully functional testbed," *IEEE J. Lightwave Technology*, vol. 13, no. 6, pp. 1093–1103, June 1995.

[2] J. Murakami, *Non-blocking packet switching with shift-register rings*, Ph.D. Dissertation, University of Illinois at Urbana-Champaign, 1991.

[3] K. Nahrstedt, A. Hossain, and S. M. Kang, "Probe-based algorithm for QoS specification and adaptation, quality of service – Description modeling and management," in *Proc. of the 4th International IFIP Workshop on Quality of Service (IWQoS96)*, Paris, Mar. 1996, pp. 89–100.

[4] E. J. Posnak, S. P. Gallindo, A. P. Stephens, and H. M. Vin, *Techniques for Resilient Transmission of JPEG Video Streams*, Dept of Computer Science, University of Texas at Austin, Web site at http://www.cs.utexas.edu/users/dmcl.

[5] A. Hossain, *Multicast Video Transport Over iPOINT ATM Testbed*, Ph.D. Thesis, Department of Computer Science. University of Illinois at Urbana-Champaign, May 1997.

[6] R. W. Horst, *TNET (ServerNet): A reliable system area network*, *IEEE Micro.*, vol. 15, no. 1, pp. 512–519, Feb. 1995.

[7] W. Baker, R. Horst, D. Sonnier, and W. Watson, *A flexible serverNet-based fault-tolerant architecture*, in *25th Intl Symposium on Fault-Tolerant Computing*, Pasadena, CA, June 27-30, 1995.

[8] A. Hossain, S. M. Kang, and R. Horst, *Performance comparison of video transport over ATM and ServerNet interconnects*, in *Proc. of IEEE Multimedia 97*, Ottawa, Canada.

[9] G. Pawan, H. M. Vin, C. Shen, and P. Shenoy, "A reliable, adaptive network protocol for video transport," in *Technical Report TR-95-18*, Dept of Computer Science, University of Texas at Austin, 1995.
[10] D. Clark, V. Jacobson, J. Romkey, and H. Salwen, "An analysis of TCP processing overhead," *IEEE Commun. Mag.*, pp. 23–29, June 1989.
[11] S. Heatley and D. Stokesberry, "Analysis of transport measurements over a local area network," *IEEE Commun. Mag.*, pp. 16–22, June 1989.
[12] K. Keeton, T. Anderson, and D. Patterson, "LogP quantified: The case for low-overhead local area networks," *Hot Interconnects III*, Stanford University, August 10-12, 1995.

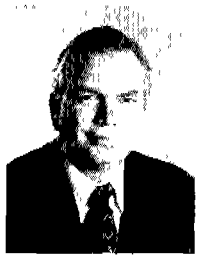
**Ashfaq Hossain** is a Member of Technical Staff at Bell Laboratories, Lucent Technologies, since July 1997. His professional interests are networking software development and high-speed Ethernet/ATM switching hardware design.

He holds a Ph.D. in Computer Science from University of Illinois at Urbana-Champaign (June 1997) where his research focused on developing software and hardware for an Asynchronous Transfer Mode (ATM) switching testbed. Dr. Hossain has applied for 2 U.S. patents.

**Sung-Mo (Steve) Kang** received his Ph.D. in Electrical Engineering from the University of California at Berkeley in 1975. Until 1985 he was with AT&T Bell Laboratories at Murray Hill and Holmdel.

In 1985, he joined the University of Illinois at Urbana-Champaign where he is Department head of Electrical and Computer Engineering and Research Professor of the Coordinated Science Laboratory and the Beckman Institute for Advanced Science and Technology.

Prof. Kang has co-authored many technical papers and five books: Design Automation For Timing-Driven Layout Synthesis(1992), *Hot-Carrier Reliability of MOS VLSI Circuits*(1993), *Physical Design for Multichip Modules* (1994), and Modeling of Electrical Overstress in Integrated Circuits (1994) from Kluwer Academic Publishers, and CMOS Digital Integrated Circuits: Analysis and Design by McGraw-Hill (1995), and Computer-Aided Design of Optoelectronic Integrated Circuits and Systems by Prentice-Hall(1997).



**Robert Horst** is Director of Research at 3ware, Inc. where he is developing high performance reliable storage systems. Previously he was a Technical Director in Compaq's Tandem Labs where he led the architecture team that developed the ServerNet System Area Network. At Tandem, he also contributed to the architecture and design of five generations of fault-tolerant parallel computer systems.

He holds a Ph.D. in Computer Science and MSEE from the University of Illinois and a BSEE from Bradley University. He is a senior member of IEEE, a member of ACM, and holds 37 US patents.