A Server Array Approach for Video-on-demand Service on Local Area Networks

Y.B. Lee and P.C. Wong
Advanced Network Systems Laboratory
Department of Information Engineering
The Chinese University of Hong Kong, Hong Kong
Tel:(852)-2609-8372 Fax:(852)-2603-5032
{yblee, pcwong}@ie.cuhk.hk

Abstract

Most video-on-demand (VOD) systems use a single, powerful server to deliver video streams to users. In this paper, we consider a novel server array approach for delivering video services on networks. Such an approach has the benefits of (1) more system capacity, as individual server has individual disk and network channel, (2) scalable, as more clients can be supported by adding more servers without data duplication, and (3) fault tolerant, as server-level fault-tolerant and fault-recovery schemes can be devised. We describe our experiences and results in the implementation of a server-array-based VOD system. Our system now has four P5-90 servers serving 40 PC-486 stations using a 10Mbps Ethernet switch. The system can deliver 40 simultaneous and independent 1.2Mbps, 30 fps, full TV size, MPEG-1 video streams. Our results demonstrate that with careful protocol designs, Ethernet can deliver continuous video and audio services.

1. Introduction

Video-on-demand (VOD) service has many exciting applications. Examples are movies-on-demand, music video (karaoke) on-demand, video magazines, video kiosks, computer-aided-training, and video library, etc. With the advent of network and video technologies, there has been tremendous interest in recent years on various kinds of video distribution services and technologies [1-6]. In this paper, we consider the delivery of 30 fps, TV size and quality, MPEG-1 video streams on a local area computer network (e.g., Ethernet).

Consider a VOD system which has only one video server connecting to a number of client stations through a network. The video server reads video blocks from the disk storage, processes blocks into packets, and transmits the packets to client stations through the network. The server capacity is limited by three factors: disk throughput, CPU processing and I/O capability, and network access bandwidth. To increase the system capacity, one can use (1) a disk array instead of a single disk, (2) a more powerful CPU, or (3) a high-speed network such as ATM. Yet, these approaches are expensive and possibly not cost-effective for small scale multimedia systems. Another way is to duplicate video storage on a number of servers. As video needs enormous storage (~1 Gbyte per movie for MPEG-1 streams), video duplication is undesirable and yet inevitable if we have to support a very large number of users. In this paper, we consider a *server array* approach where video blocks are interleaved across servers instead of being duplicated. Server array is a counter-part of interleaved memory banks and disk arrays [7], so is a natural extension of load sharing at the server level.

Figure 1 shows the architecture of a server array VOD system. A fast packet switch fabric is used to connect video servers and client stations. As individual server has its own disk subsystem and network segment, the available network and disk capacity is scaled up with the number of servers. In the following, we will discuss the design issues one by one.

First, we note an asymmetric traffic requirement between servers and clients. So the servers may be assigned with one or more high-speed links, whereas several clients may share a single low-speed link, as will be illustrated later in our implementation. Second, we see the need of a many-to-many communication protocol between clients and servers. It can reduce server processing overhead as compared with managing multiple point-to-point connections. It can also coordinate server transmissions so as to minimize congestion when multiple servers transmit to the same client at the same time. From our experience, the video transport protocol must exhibit the following characteristics. First, it must be able to deliver video packets in time to ensure playback continuity. Second, it

0743-166X/96 $5.00 © 1996 IEEE

must be able to recover packet loss to avoid video degradation, especially when error concealment capability is not available. Third, it must reduce the backward traffic from clients to servers. This can reduce the server processing overhead and collisions on a contention-based network such as Ethernet. Finally, the protocol must be fast and efficient.

2. System design

We will discuss three aspects in system design: the video transport protocol, the server, and the client.

2.1 Video transport protocol (VTP)

Our VTP consists of two parts: VTP Server and VTP Client. Figure 2 shows how a server and a client communicate. The client sends requests to the servers one by one, and the server will transmit the video packets with mechanisms for flow-control, and packet loss recovery. Video data packets are transmitted using a fast datagram protocol whereas control packets are transmitted using a reliable datagram protocol [8].

2.1.1 Flow control

The video transport protocol use a credit-based flow-control algorithm with prefetch-buffering. The client buffer consists of N_B video blocks, each of Q packets. The blocks are managed as a circular buffer. At connection setup time, the first (N_B - 1) blocks are prefetched from the servers before video playback is started. We do not prefetch all N_B blocks as the empty buffer will always be used for accommodating the next newly arrived video block. In this way, we avoid memory copy when we submit a video block for playback. The video block can still occupy the space in the circular buffer during playback.

The client then initializes the video decoder, and submits the first video block to the decoder for playback. At the same time, the client will send a request to one server for a new video block of Q packets. When the playback device finishes with a video block, the client will submit the next block to the device, and at the same time sends a request to the next server, and so on. In this way, the servers will never transmit more video data than a client can handle. At the client station, video packets are inserted into the receiver buffer directly using a packet pointer. So the sequence of packet arrivals is not important as long as they arrive before the time of playback. This *direct-buffer-insertion* technique is very useful as a client has to retrieve video blocks from several servers. Simply, video packets from different stations may arrive out of sequence.

As described earlier, we need to reduce the backward control traffic to a minimum. We see that the percentage of request traffic inside the network is $P_C / (P_C + Q P_V)$, where P_C and P_V are the sizes of control packets and video packets respectively. In our implementation, we have Q=15, P_C=48 bytes and P_V=4110 bytes, so the percentage of request traffic in the network is less than 0.1%. Such a small value can greatly reduce server processing overhead and can effectively avoid Ethernet collisions.

2.1.2 Packet loss recovery

In a local area network, there is a possibility of packet loss at a client station even if the network utilization is not high. This is because the client station may be busy in other tasks (e.g., submitting video blocks to the decoder) while video packets are arriving. Packet loss can affect the video quality, and can be detrimental if the lost packet contains control or time-stamp information. In particular, we need a time-sensitive retransmission scheme to recover the lost of video packets. In our scheme, a client will scan through the list of video packets whenever it sends a new video request. If there are lost packets which can be recovered before the playback time, retransmission requests will be transmitted to the server. The server will retransmit the lost packets accordingly. In this way, lost packets will not block future packet reception, which is important to ensure continual video reception. To protect unnecessary retransmissions due to random delay in packet arrivals, we set a minimum retransmission interval T_R so retransmission requests will not be sent for packets which have not exceeded T_R.

On the other hand, we need to limit the amount of retransmission traffic on the network as well. This is to protect the system from malfunctioning clients which might overload the network with retransmissions. We limit each control packet to request for at most N_R retransmission packets, so the server will retransmit at most N_R lost packets for every new block of Q video packets. The ratio of retransmission traffic over actual traffic is therefore $R=N_R/Q$. In our implementation, we set N_R=3, and Q=15, we have R=20%. Note that N_R cannot be too small, otherwise the retransmission process will be very slow. In reality, the packet loss percentage is less than 1%.

2.2 Server Design

While large video block gives better efficiency, a client may get congested if a server transmits to the client the entire block of packets in a short time. We therefore need traffic shaping algorithms to control the rate of

transmission. Many such algorithms proposed are designed for single traffic source, and are concerned with controlling the rate at the user-network interface. In the server array architecture, each server serves multiple video streams simultaneuosly. It is desirable to have a single scheduling algorithm controlling all video streams simultaneously.

We use a novel Batch-Round-Robin (BRR) algorithm with queue-sharing at each server to perform scheduling and traffic shaping. In a server, there is one request queue, one shared queue and M send queues. The incoming requests are stored into the request queue, and video blocks are read from disk storage into the shared queue. Whenever a send queue is empty, a block from the shared-queue is moved into the send queue for transmission. Each send queue therefore contains at most one block of packets at a time and each send queue can serve video blocks from any video streams.

Assume that M_A out of the M send queues have packets for transmission. These send queues are serviced in a round-robin manner, with g packets transmitted in each round. The parameter g controls the maximum number of consecutive packets (hence burst size) from each video stream. The instantaneous rate r_A of packet transmissions to a client will be C_S/M_A, where C_S is the per-segment network throughput in packets per second.

If the system is lightly loaded, most of the send queues would be empty and r_A may be too large for a client. To circumvent this we define a minimum interleaving interval T_I (in packet times). If the time for one round of transmission is less than T_I, i.e., $g(M_A - 1) < T_I$, the server will simply wait until T_I before starting the next round. Therefore, the maximum instantaneous transmission rate for any client will become $r_M = gC_S/(g + T_I)$. Similarly, the minimum transmission rate for a stream will be $r_m = C_S/M$, when all send queues have packets for transmission.

To ensure that servers are not overloaded, we use a simple admission control algorithm. A new stream is admitted only if the aggregate rate is below the server capacity, i.e., $\sum r_i \leq C_S$, where r_i is the average rate of individual streams. Note that $r_M \geq r_A \geq r_m$, and r_A have to be greater than r_i to ensure that the clients can receive video blocks at a rate faster than the blocks being consumed by the video decoder.

The proposed BRR algorithm has advantages over existing scheduling disciplines like FCFS and Round-Robin (RR). In fact, BRR degenerates into FCFS and RR schemes when M=1 and M equals to the number of active stations respectively. FCFS cannot control the transmission rate and burst size. RR has to manage a varying number of send queues, thus is more complicated if there is a large number of client stations. In BRR, we can have a guaranteed minimum transmission rate r_m, maximum transmission rate r_M, and maximum burst size g, which are controlled by the number of send queues M, the minimum interleaving interval T_I, and the service granularity g respectively. In addition, the memory requirement is fixed irrespective of the number of clients and servers, as long as the client-server ratio is fixed. This is important for the server array to be scalable for supporting more clients.

2.3 Client Design

At a client station, a circular buffer is used to absorb variable delays to ensure video playback continuity. The buffer also allows lost packets to arrive before playback. Below, we analyze the system delay and derive the client buffer requirement.

Figure 3 shows a typical statistics of the video block playback time of an MPEG decoder. As a client will send a request to the server array whenever a new video block is submitted to the decoder, the same statistics indicate the interarrival time for video requests to the servers. We see that the distribution has a mean T_m which is the reciprocal of video rate (1.2 Mbps in our case). We see a lower bound T_L and an upper bound of T_H. Figure 4 shows the statistics of system delay for a client to receive video packets after sending a video request to a server. The waiting time has a maximum value D_{MAX}. From Section 2.2, the transmission time required for the whole video block of packets will be Q/r_A, where $r_M \geq r_A \geq r_m$. The maximum time required T_T will be MQ/C_S. Consequently, the maximum time for receiving a video block from a server is

$$T_D = D_{MAX} + T_T \quad (1)$$

As long as one video block is consumed by the decoder, another block must be ready in the receive buffer for video playback continuity. As the client starts to play the $(k + N_B - 1)^{th}$ block, the request of this block must have been submitted $(N_B - 1)$ playback time earlier, i.e., while the client started to play the k^{th} video block. So we must have

$$(N_B - 1)\, T_L \geq T_D \quad (2)$$

or

$$N_B \geq 1 + (D_{MAX} + T_T)/T_L \quad (3)$$

in order that all video blocks will be received in time for playback.

In the above derivation, we have not considered packet loss and retransmissions, so is in many studies [9]. To incorporate delay in retransmission, we need to know the characteristics of packet loss. With our traffic shaping scheme implemented, the loss probability becomes very small, and packet loss are in the form of a burst of up to m packets. We observed that the loss occurs when the client is busy in other tasks. But we can safely assume that there will be no more than one outstanding loss burst at a time. As each retransmission request claims for N_R lost packets only, we need $s=\lceil m / N_R \rceil$ requests for recovering m lost packets. Our experiments show that almost all lost packets can be recovered in only one retransmission.

Let t_k be the time the client sends out the $(k+ N_B - 1)^{th}$ video request. At time $t_k + D_{MAX} + T_T$, the client will have received the block and find that the block has lost packets. Let $v=\lceil (D_{MAX} + T_T) / T_L \rceil$, then at the worst case the client will be sending retransmission requests at t_{k+v}, t_{k+v+1}, ..., $t_{k+v+s-1}$. Hopefully, the lost packets for the last retransmission request will arrive at time $t_{k+v+s-1} + D_{MAX} + MN_R/C_S$.

Note that each retransmission request claims for N_R packets only. To ensure video continuity, the last block of retransmitted packets should be ready by the time the client plays the $(N_B - 1)$ video block. So we must have

$$(N_B - v - s)\, T_L \geq D_{MAX} + MN_R/C_S \quad (4)$$

or

$$N_B \geq v + s + (D_{MAX} + MN_R/C_S)/T_L \quad (5)$$

Even if we continue with more retransmissions, the resulting N_B would not be much different. On the other hand, our results can be extended for networks with longer delay (e.g., wide area networks) by using a large value of D_{MAX}.

2.4 Interleave policy

In the server array, video duplication is avoided by interleaving video blocks across the servers. Cnsider a video file of length L packets. It can be divided into B blocks of Q packets, with the last block smaller or equal to Q. For simplicity, these blocks can be stored in the servers in a round-robin manner as shown in Figure 5. However, the storage and loading will not be evenly distributed among the servers. Our solution is to randomly assign one server as a *starting* server for any video stream, by placing the first video block in that server. The other video blocks are stored in subsequent servers in a round-robin manner. A database can be used to store the information of all video files and the starting server for each video stream.

System Parameters and Variables	Symbol	Empirical Value
Network throughput (per segment)	C_S	320pps
Min retransmission interval	T_R	200ms
Min interleaving interval	T_I	20ms
Max system delay for 10 active clients	D_{MAX}	300ms
Min block playback time	T_L	160ms
Interleave Granularity	g	1
Number of send queues	M	5
Max # of retx packets per requests	N_R	3
Video block size (packets)	Q	15
Maximum burst size of lost packets	m	6 (ISA) 3 (PCI)
Client buffer size	N_B	9 blocks

Table 1: System parameter values

2.5 Fault tolerance

Like disk array, server array can have reliability problem. Specifically, failure in one server will render the entire server array inoperable because there is no data duplication. Fortunately, schemes operate at the server level analogue to Redundant-Array-of-Inexpensive-Disks (RAID) can be devised for the server array. Through redundant servers and client data reconstruction, a server array can have fault tolerant capability. We are considering various architectures for Redundant-Array-of-Distributed-Servers (RADS), and will report our results in a later paper.

3. Implementation results

To demonstrate the feasibility of the server array architecture, we implemented a video-on-demand system on top of a 16x16 Ethernet Switch, each segment running at 10 Mbps. Our server array consists of 4 independent servers. Each is a P5-90 with two 10Mbps Ethernet links and 4GB harddisk for video storage. So there are altogether 16GB video storage (~1800 minutes of 1.2Mbps MPEG video). Eight network segments are used for connecting client stations. Each switch port is connected to a multiple-port repeater serving five client stations. So a total of 40 client stations can be supported. The client-server ratio is 10, and the

utilization of both server and client segments are around 66% (~6.6 Mbps with protocol overhead included). We are in the process of ordering a Fast Ethernet switch. Our preliminary evaluation shows that a P5-90 server can deliver over 40Mbps throughput using a 100Mbps Fast Ethernet adapter. This means that the client-server ratio can be greatly increased. When the Fast Ethernet switch arrives, we will be using Fast Ethernet adapters at the servers, and client stations will still be sharing a 10Mbps segment. Table 1 shows all the system parameters unless otherwise defined.

3.1 Protocol performance

An experiment was set up to compare VTP with three existing transport protocols (TCP, NetBIOS, SPX) on a shared Ethernet segment with five video client stations. The network utilization is obtained with a protocol analyzer, and the server CPU utilization is obtained from the Windows NT performance monitor. Figure 6 shows that the network utilization of VTP increases linearly with the number of clients. It also shows that the network overhead is minimal. The overhead includes protocol header, control traffic, and packet retransmissions. Using VTP, five clients can be supported with full frame rate and continuity. For the other three protocols, we observe significant packet loss, and the video becomes jerky if there are more than 2 video streams. Note that the network utilization for TCP, NetBIOS, and SPX all decrease for more than 2 clients. This is due to collisions at the Ethernet which reduce the network utilization. Figure 7 shows similar result for server CPU utilization. Clearly, VTP has a much better performance than the three protocols we tested for video transmission.

3.2 Packet loss recovery

The packet loss probability at the client stations depends on the instantaneous rate and burst size of video packets transmitted by the server. The packet loss probability also depends on the network adapter used. Table 2 summarized the packet loss probabilities for various network cards and interleave granularity g. Clearly, ISA-bus network adapter loses more packets than PCI-bus network adapter, and the loss probability increases with g. However in all cases, all packet loss can be recovered by the TCR scheme, resulting in a smooth video playback. From our statistics, nearly all packet loss are recovered in one retransmission, and the amount of duplicate retransmission is negligible.

Netcards (10 Mbps)	g (packets)	Packet loss w/o TCR	Packet loss w/ TCR
ISA-bus	1	~10%	0%
PCI-bus	5	2%	0%
PCI-bus	2	1%	0%
PCI-bus	1	< 0.5%	0%

Table 2: Packet loss probabilities

3.3 Client buffer

From system parameters, we have $N_B \geq 4$ if there were no packet loss. Incorporating packet loss calculations, we obtained v=4 and s=2 in the case of ISA-bus adapter. The client buffer size is therefore $N_B \geq 9$ blocks (540KB) according to (5). This confirms with our experimental results that using 9 or more blocks of buffer, we do not observe any video discontinuity. The logging of the client buffer occupancy shows that the derived buffer size is in fact more than sufficient (Figure 8), as our derivation is a worst-case calculation. The video prefetch delay is around 3 seconds under full system loading.

3.4 Server array utilization

To demonstrate the scalability of server array, we run a test to benchmark the server CPU utilization for server arrays with 1, 2, and 4 servers. Figure 9 shows that a single P5-90 server can support up to 10 client stations on two Ethernet segments, and the server CPU utilization is roughly proportional to the number of active clients. On the other hand, the CPU utilization is reduced in a linear fashion when more servers are added. Note that the system capacity is CPU limited in this case. We believe that similar result applies with other configurations where the system is network or disk limited.

4. Conclusion

In this paper, we considered a novel server array approach in designing a video-on-demand system. We described in detail the design and implementation of a working VOD system. While other traffic control schemes focused on network aspects, our results applied at the application level and on an end-to-end basis. We demonstrated that even the underlying network (Ethernet) is unreliable and with no QOS control, a careful design of higher layer protocols can result in a successful application. Further work is in progress on the fault tolerant issues, and on the extension to wide area VOD network.

Acknowledgements

This research is partly supported by the Hong Kong Government Industrial Department, and the Ho-Sin-Hang Educational Endowment Fund.

References

[1] W. Hodge, S. Mabon, J.T. Powers, Jr., "Video On Demand: Architecture, Systems, and Applications," *SMPTE Journal*, September 1993, pp. 791-803.

[2] W.D. Sincoskie, "System Architecture for a Large Scale Video on Demand Service," *Computer Networks and ISDN Systems*, North-Holland, vol. 22, 1991, pp. 155-162.

[3] H. Armbrüster, K. Wimmer, "Broadband Multimedia Applications Using ATM Networks: High-Performance Computing, High-Capacity Storage, and High-Speed Communication," *IEEE Journal on Selected Areas in Communications*. vol. 10(9), 1992, pp. 1382-1396.

[4] S. Gibbs, D. Tsichritzis, A. Fitas, D. Konstantas, Y. Yeorgaroudakis, "Muse: A Multi-Media Filing System," *IEEE Software*, 4(2):4-15, March 1987.

[5] R.R. Thomas, H.C. Forsdick, T.R. Crowley, R.W. Schaaf, R.S. Tomlinsin, V.M. Travers, G.G. Robertson, "Diamon: A Multimedia Message System Built On a Distributed Architecture," *Computer*, 18(12), December 1985, pp. 65-78.

[6] F.A. Tobagi, J. Pang, "StarWorks™ - A Video Applications Server," *IEEE COMPCON Spring '93.*

[7] D. Patterson, G. Gibson, R. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," *Proceeding of the ACM SIGMOD Conference*, June, 1988, pp. 109-116.

[8] Y.B. Lee, P.C. Wong, "VIOLA - Video On Local-Area-Networks," *Proceeding to the 2nd IASTED/ISMM International Conference on Distributed Multimedia Systems*, Stanford, California, 1995.

[9] P.V. Rangan, H.M. Vin, S. Ramanathan, "Designing an On-Demand Multimedia Service," *IEEE Communications Magazine*, Vol.30, No.7, July 1992, pp. 56-65.

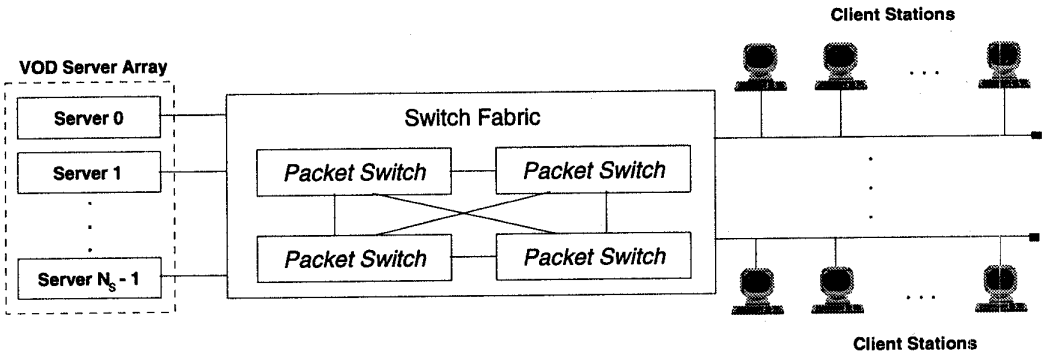

Figure 1: A server array VOD system

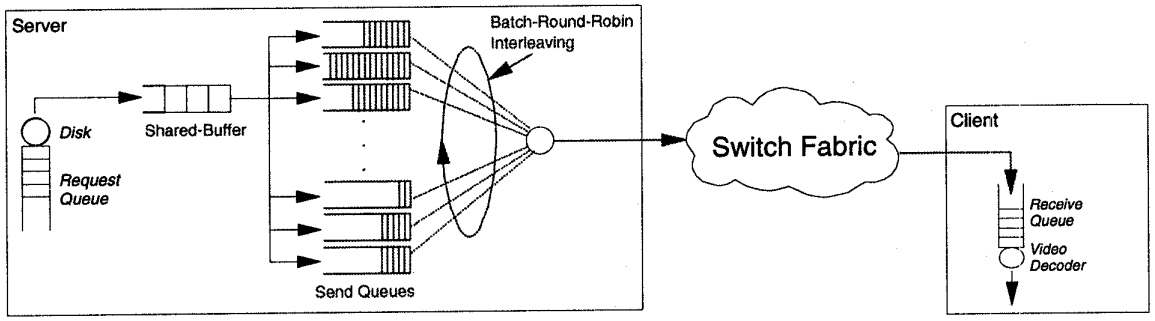

Figure 2: Communications between servers and clients

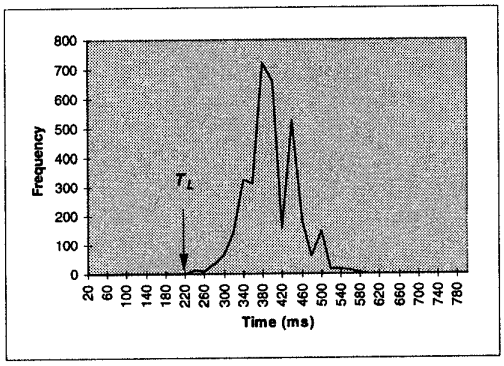

Figure 3: Video block playback time distribution

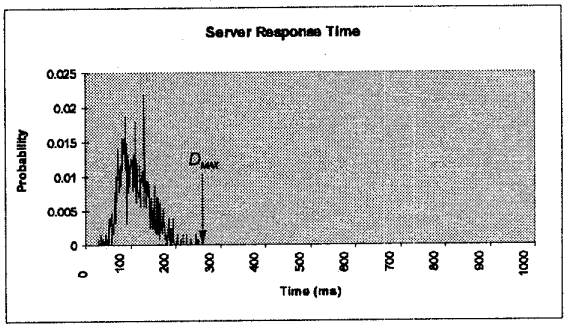

Figure 4: System delay distribution

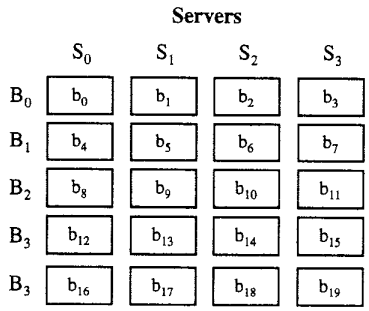

Figure 5: Video block placement policy

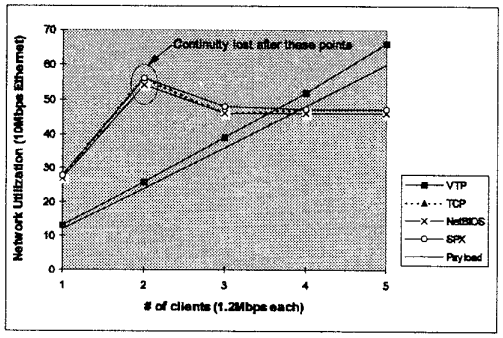

Figure 6: Network utilization comparisions

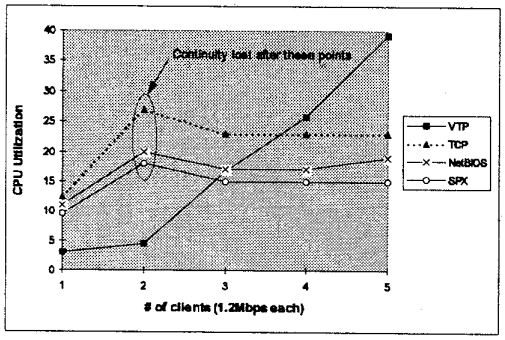

Figure 7: Server CPU utilization comparisions

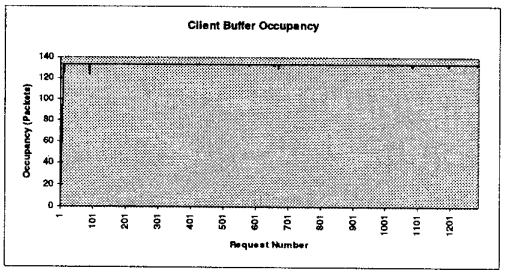

Figure 8: Client buffer occupancy

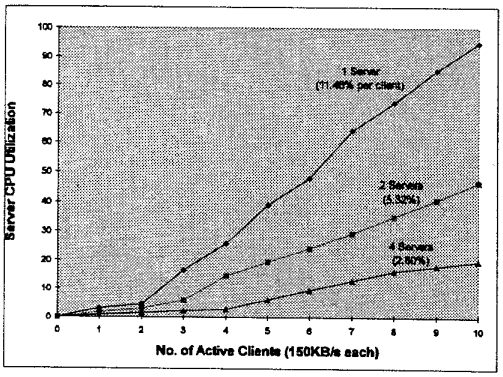

Figure 9: Server array CPU utilization