PAPER Special Issue on Multimedia on Demand

Communication Processing Techniques for Multimedia Servers*

Mitsuru MARUYAMA†, Kazutoshi NISHIMURA††, *and* Hirotaka NAKANO††, *Members*

SUMMARY Three techniques are proposed for reducing the time required for protocol processing: protocol data unit management using page management, assembly and disassembly of data packet header and contents in hardware, and rescheduling of protocol processing. These techniques were shown to be feasible by applying them to the TCP/IP over a fiber-distributed data interface network. The maximum communication throughput was 91.6 Mbps; the total throughput for 64 sessions was 89.6 Mbps, only 2% less than the maximum. These techniques will enable the development of more efficient video-on-demand systems.

key words: *high-speed protocol processing, multimedia communication, TCP/IP, FDDI*

1. Introduction

Demand is growing for video-on-demand systems capable of simultaneously serving video and audio to client terminals connected by a high-speed network, such as a fiber-distributed data interface (FDDI) network or the forthcoming broad-band integrated services digital network (B-ISDN). The performance of a server's communications processing unit during multiprocessing of protocols supporting high-speed communications is a vital factor in the ability of such systems to simultaneously serve multiple client terminals with real-time media.

Several researchers are working to clarify the overhead in the transmission control protocol/internet protocol (TCP/IP) processing on general-purpose workstations [1]–[3]. Based on these results, reports have appeared on methods of improving TCP/IP throughput of an FDDI network, e.g.[4]. These methods use independent hardware add-ons and processing methods within the OS (operating system) for single-client connections for computer communication. However, no reports have appeared showing improvements in multiprocessing performance through the use of communications processing units for high-speed multimedia networks in the servers.

The main problem with the multiprocessing performance of such units is that transmission efficiency is reduced due to the overhead needed for switching between the protocol-processing tasks of multiple sessions. Other vital issues are reducing the overhead for checksum calculation in TCP/IP processing and for transferring information between different memory spaces. Improving the performance of basic one-to-one communication is also required.

This paper describes two new hardware technologies for high-speed protocol processing: protocol data unit (PDU) management using page management and assembly and disassembly of data packet header and contents in hardware. We have also developed a technique for improving the degree of multiplexing, called rescheduling protocol processing, and a prototype communications board that uses these new technologies; we measured its processing performance and examined its effectiveness by comparing its performance with that currently achieved on general-purpose UNIX systems.

2. Target Protocol for Video-on-Demand System

Our prototype video-on-demand system is configured with multiple the versa module europe (VME) boards as shown in Fig. 1; its main elements are shown in Table 1 [5]–[7].

We use FDDI for the low-layer protocol and the widely used TCP/IP for the high-level communications

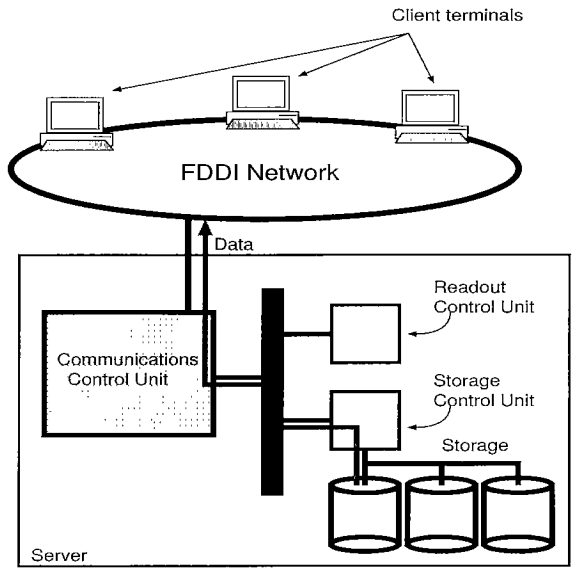


Fig. 1 Structure of prototype video-on-demand system.

Manuscript received December 20, 1995.
Manuscript revised March 22, 1996.

†The author is with NTT Software Laboratories, Musashino-shi, 180 Japan.
††The authors are with NTT Human Interface Laboratories, Musashino-shi, 180 Japan.

*This paper was presented at NTT Software Laboratories.

Table 1 Main elements of prototype system.

Item	Function/Element
media	MPEG-1/2
coding rate	1.5/6 Mbps
storage	3.5-in. SCSI magnetic disks
readout control unit	search processing, multiple readout, video recording
client terminal	commercial DOS/V PC with MPEG-1/2 decoder, FDDI board
communications processing unit:	
line	FDDI DAS standard
line-control LSI chip	FDDI MAC controller
host connection bus	VME (control), VSB (data)
board size	VME triple height (9 U)
control processor	32-MIPS RISC, 128-KB cache
program memory	4-MB RAM, 2-MB ROM
communications buffer	8 MB, 256-KB FDDI buffer
operating system	Real-time OS

VSB: VME subsystem bus

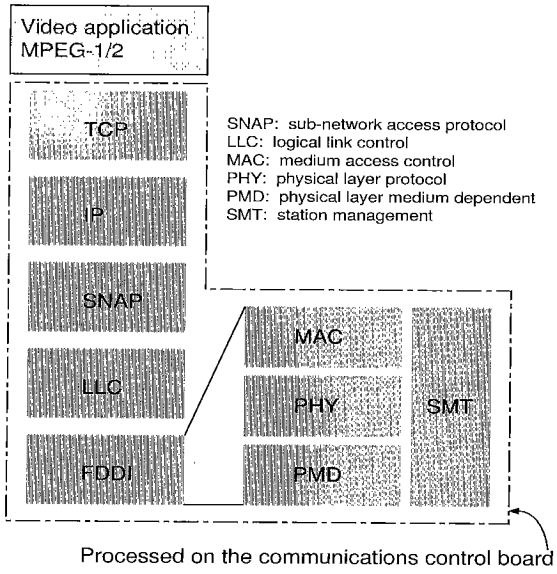


Fig. 2 Protocol layers.

protocol between the server and client terminals; this permits the use of commercial hardware and software at the client terminal end. Each of the layers (Fig. 2) is processed on the communications control board. A UNIX OS socket is used for the interface with the other control units.

This system uses the TCP's flow control, sequence control, and retransmit-on-error functions, as well as various congestion control functions. The flow control function is used to regulate the transmission rate from the server in order to accommodate the video/audio encoding bit rate. The video/audio encoding format is MPEG-1/2 (Motion Picture Experts Group-1/2) compression, and the sequence control and retransmit-on-error functions are used for recovering missing packets or errors.

The TCP/IP protocol can be implemented in a variety of ways. This system is based on the 4.3 BSD tahoe [8] implementation.

3. Conventional Protocol Processing Performance

In this section we examine the performance of conventional TCP/IP protocol processing by considering the speed of protocol processing for single-client communications and the performance for multiplexed processing. A conventional VME board with general-purpose real-time operating system was used to evaluate the single-client protocol processing speed and a UNIX workstation was used to test multiprocessing performance. We chose the conventional VME board with the same CPU type and same speed as our prototype board. The workstation was chosen to have nearly the same CPU speed as our prototype board.

Table 2 Sample study of protocol processing speed for TCP/IP transmission and reception of a 4-KB packet.

Processing type	Checksum calculation (%)	Memory movement management operations (%)	Protocol (and others) (%)	Total time (μsec)
socket input/ output	—	91	9	581
TCP data output	63	—	37	155
TCP ACK output	14	—	86	28
TCP data input	70	—	30	140
TCP ACK input	4	62	34	146

Measurement environment: 32-MIPS RISC, 128-KB cache, 4-MB RAM, real-time OS, Client terminal WS's speed: 50.2 SPECint92

3.1 Protocol Processing Speed

Table 2 shows the results of a sample investigation of TCP/IP processing speed on the conventional VME board with a reduced instruction set computer (RISC) CPU, tested with a logic analyzer. The limit on protocol processing speed was determined by the checksum calculation overhead, the complex protocol memory management, and the memory transfer overhead in the socket layer [1]–[4]. By increasing the speed of these it was possible to increase the basic single-client protocol processing speed. The TCP checksum is performed by adding the complement of one to each 16-bit unit. Overhead was generated by the CPU reading and calculating all the data in both the data packet header and contents.

Generally, the list structure called mbuf is used for

TCP memory management. This structure is used so that data buffers can be linked together by using their pointers, making it easy to allocate a variable-length data buffer. When separating or combining data within a protocol, however, allocating and opening the buffer takes time due to the re-copying of the data. During data transmission, the CPU copies data byte-by-byte from the sequential user memory space to the chained mbuf structure in the kernel space. During data reception, the reverse occurs: the CPU copies data byte-by-byte from the kernel space to the user space. These both take time [4].

3.2 Multiprocessing Performance

With a UNIX workstation, the throughput for simultaneous transmissions from a server with active TCP/IP sessions connected to multiple client terminals is as shown in Fig. 3. The horizontal axis shows the number of connections, and the vertical axis is the throughput. The transmission throughput is the single-connection processing performance multiplied by the number of connections. The performance for a single connection was approximately 34 Mbps; as the number of connections increased, the performance dropped sharply, so with 50 connections throughput fell to 35% of that with only one. The causes of this, which were also cited as sources of protocol processing overhead by P. Druschel et al. [3], were mainly the task-switching overhead due to multiprocessing and the interrupt overhead due to asynchronous reception.

The task-switching overhead with multiprocessing arises because there is a server process running for each client terminal. As the number of simultaneous processes grows, context switching between the user and kernel spaces becomes more frequent, which is overhead in UNIX.

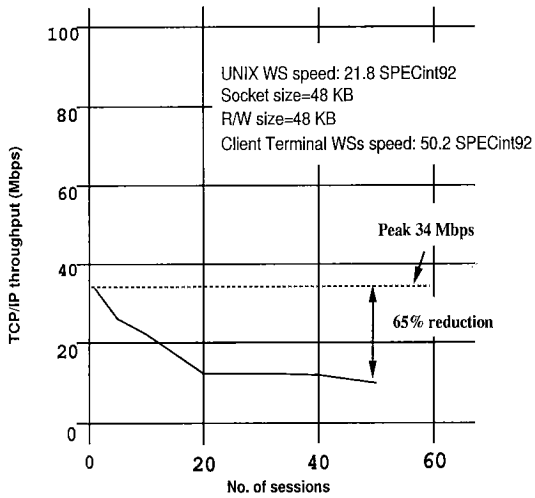


Fig. 3 Multiprocessing performance on a general-use UNIX workstation.

The interrupt overhead due to asynchronous reception arises because interrupts are used to notify the OS asynchronously of acknowledges (ACKs) and other reception data from the client terminals, which causes the OS to immediately initiate reception processing. With TCP, when an ACK reception verifies normal reception by the remote node, the transmission memory is cleared. A context switch occurs in the kernel each time an ACK is received. If the ACKs are received at irregular intervals, efficient transmission becomes impossible because the transmission processing is continuously interrupted.

4. Performance of Prototype Communications Processing Board

In this section we describe several techniques for increasing the protocol processing speed and a combined technique for increasing the degree of multiplexing.

4.1 Techniques for Increasing Protocol Processing Speed

The operation of our prototype communications board during data transmission and reception is shown in Fig. 4. The board transfers the data from the storage control unit across the data bus to the transmission data buffer. In the same transfer cycle, the data is also transferred to the checksum calculation circuit. This circuit then calculates the checksum of the content part. Based on the IP address, the TCP port number, and other information, the control CPU performs TCP/IP header processing including header checksum calculation and writes the result to header memory. Next, the CPU calculates the TCP checksum value from the header checksum and the content checksum. Then the assembly unit, which has a direct memory access (DMA) function, joins the header information to the data in the transmission data buffer and transfers it to the FDDI MAC controller.

When receiving data, the FDDI MAC controller captures the data frames from the line. The disassembly unit simultaneously writes the data to the reception data buffer and header memory. The checksum calculation circuit for reception also monitors the data in the bus for the same period as the cycle when data is written to the reception data buffer, and calculates the checksum. The CPU begins analysis of the header at the same time that the leading portion of the data packet is transferred to header memory. When all of the received data is stored in the reception data buffer, the reception checksum calculation results are read and compared with the value received in the header portion. Once verified, the data is transferred to the host data bus by DMA processing.

The new techniques are described below.

1. PDU management using page management

   Since our board treats continuous data streams,

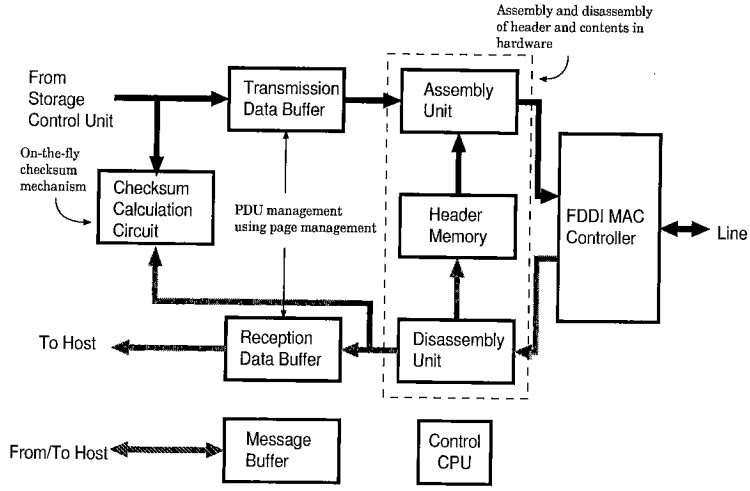


Fig. 4 Prototype board operation during data transmission and reception.

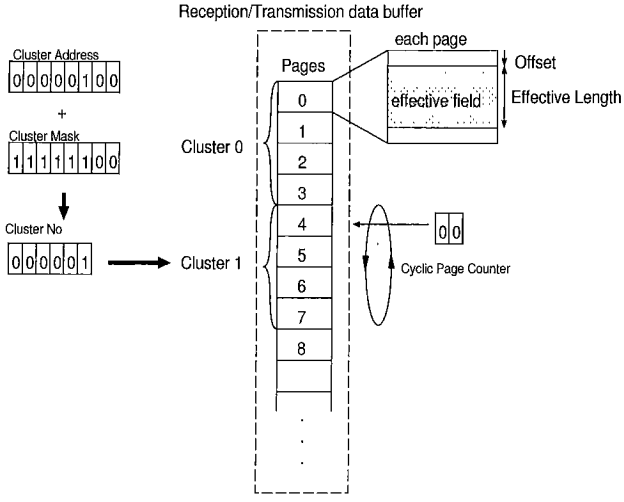


Fig. 5 Memory management architecture.

such as MPEG, that can be divided into fixed segment size, the PDU management functions can consist of page management and a simple counter. The reception and transmission data buffers, as shown in Fig. 5, are data areas divided into pages of fixed size. The contents of each page are managed according to their offset addresses and effective lengths. The method for allocating and clearing each page number is administered according to the cluster address and cluster mask.

The effective bits of the cluster address are specified according to the cluster mask, and thus the size of the pages allocated in a cluster can be changed automatically. This enables application-specific page allocation. To specify a page number in a cluster, a cyclic page counter is used, which can be achieved with simple counter hardware.

An example of this method is shown in Fig. 5. In this example, each cluster controls four pages because the cluster mask has six bits.

Using these pages, managed by their offset and effective length, and the method for allocating and clearing memory, complex buffer management can be replaced by the mbuf chain. To perform PDU management using this page management method, this prototype board uses two programmable LSI logic chips.

2. Assembly and disassembly of header and contents in hardware

   The assembly and disassembly units (Fig. 4) consist of two LSI programmable chips that separate and reassemble the data packet header (LLC, SNAP, IP, and TCP header) and contents in hardware. The board architecture physically separates the memory for the contents and header portions of the protocol and automatically reassembles and separates them during DMA data transfer. In this way, data transfer never competes with the CPU for memory access, even when the CPU is performing header operations.

3. On-the-fly checksum calculation mechanism

   This mechanism is in the TCP layer and is part of the checksum calculation unit that calculates the transmission and reception checksums during data transfer (Fig. 4). This mechanism implements the TCP checksum algorithm in hardware. It samples the data in the bus during the memory transfer cycle and calculates the checksum values within the same cycle. The control CPU reads and uses the calculated results, which are stored in the register after transfer. When fragmentation occurs and a segment is divided into several parts, which is rare, the checksum of each part is calculated by the control CPU. Furthermore, since error detection methods may differ if other protocols are implemented,

this prototype board has a calculation circuit that uses a programmable LSI logic chip.

4.2 Combined Technique for Increasing the Degree of Multiplexing

Pipeline protocol processing draws upon the parallel nature of multiple transfers by ordering the memory transfer and CPU protocol processes. Specifically, during transmission, protocol processing is divided into three stages: (a) data input, (b) protocol processing, and (c) output processing. These are achieved through pure pipeline operations. Stage (b) corresponds to CPU processing time. If this pipeline operation is an ideal, the overhead of copying data and protocol processing is negligible. To further facilitate this pipelining, our board uses a general-purpose real-time operating system for fast task switching and combines "rescheduling protocol processing" with "reducing task-switching overhead by reducing the number of tasks used".

1. Rescheduling protocol processing

   The processing of ACK signals received at irregular intervals and the consequent clearing of memory are delayed so they do not affect pipeline operations. As shown in the upper part of Fig. 6, data is received asynchronously from the line during transmission. When an ACK is received and the resulting clearing of memory occurs, all the CPU time is utilized and protocol processing ((b) in the figure) is delayed. As a result, the output operation (c) is delayed. With rescheduling, the lower part of Fig. 6, when an interrupt arrives, only enough processing is performed to confirm that it is an ACK packet and to set a flag that an interrupt has occurred. ACK processing is postponed until normal protocol processing has finished. After normal protocol processing, the flagged ACKs are processed. This prevents degradation in pipeline performance.

2. Reducing task-switching overhead by reducing the number of tasks used

   If task switching occurs frequently, it consumes CPU time and delays the pipeline cycle. In our board, the individual server tasks for each of the client terminals are combined into a single task so that operation occurs without task switching. Tasks that should be processed within a set time are processed sequentially, without status management within the task, so that task switching is not required.

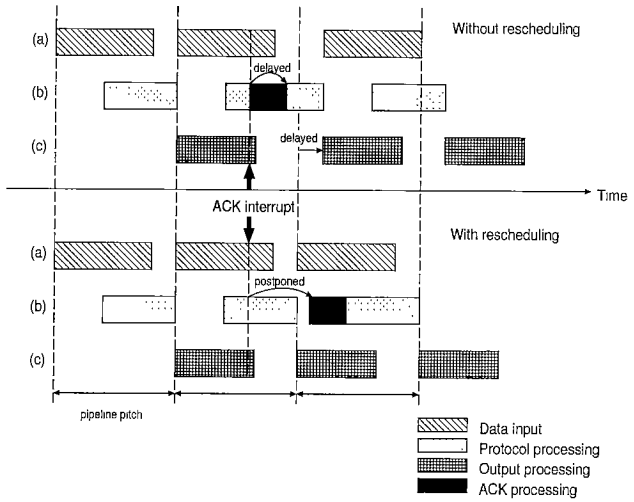


Fig. 6 Rescheduling of protocol processing.

5. Performance Evaluation

5.1 Improving Single Protocol Processing Performance

We evaluated the performance of our prototype communications board for the transmission of 4 KB of data and reception of ACK packets. A 14-fold improvement over conventional methods in checksum calculation performance was achieved in the transmission of 4-KB data (Fig. 7). The time needed to process the other TCP/IP protocol tasks was reduced by 21% as a result of using assembly and disassembly of data packet header and contents in hardware and PDU management using page management. The time needed for memory clearing of ACK packets was reduced by about 64% as a result of using PDU management using page management. The nominal time for memory transfer was reduced by 40% in the transmission of 4-KB data as a result of using PDU management using page management; the actual improvement is much higher because the pipelined functioning lets memory transfer occur in parallel with CPU processing. It takes an average of 340 μs to transmit 4 KB of data when using the pipeline pitch in Fig. 6. Since the data-input transfer rate from the data path was 321 μs, the protocol processing overhead was only 19 μs. Clearly, an average transmission rate within the board of 96 Mbps is attainable using our techniques.

5.2 Improving Multiprocessing Performance

The multiprocessing performance of our board is shown in Fig. 8. The data was collected from the prototype board transmitting simultaneously to a number of off-the-shelf workstations via a network analyzer. The figure shows the results for multiple simultaneous transfers to the workstations. With only one connection, only 55 Mbps was reached with or without reschedul-

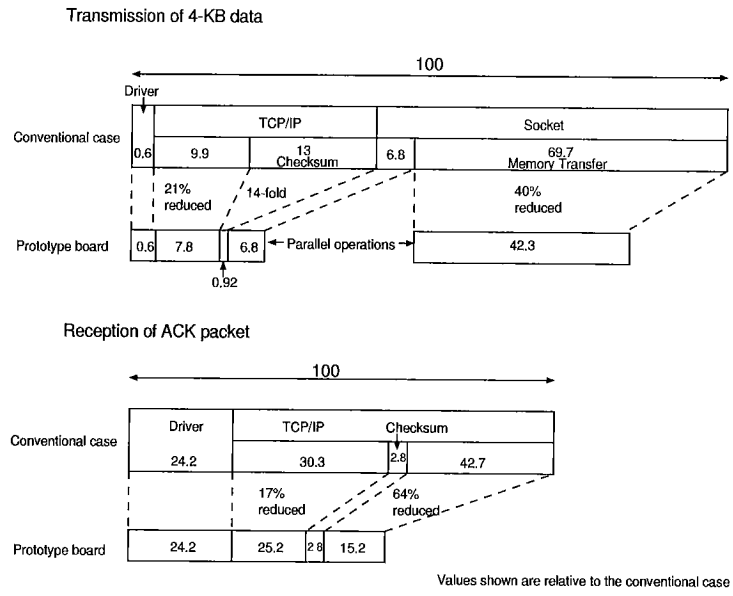


Fig. 7 Single-protocol processing performance.

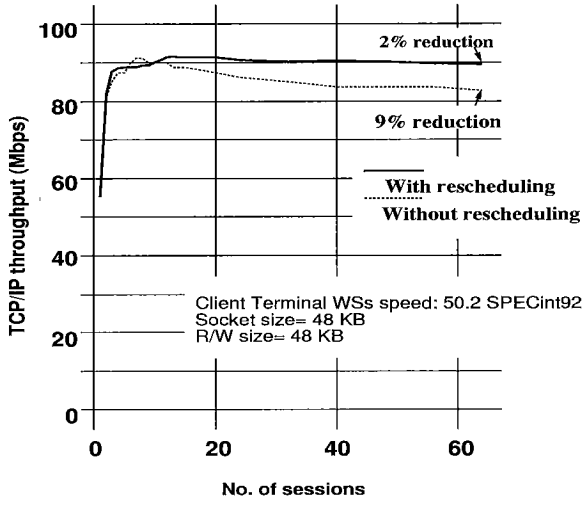


Fig. 8 Multiprocessing performance on the prototype board.

ing. This is due to limitations imposed by the commercial workstations' performance. The throughput peak with rescheduling was is 91.6 Mbps with 13 connections and it decreased only 2% with 64 connections. On the other hand, the throughput peak without rescheduling was 91.2 Mbps with 8 connections and it decreased 9% with 64 connections. Applying rescheduling gave a throughput drop with 64 connections that was 7% better than without rescheduling. These results show that by rescheduling protocol processing, pipelining can provide near-ideal operation.

6. Conclusion

We have developed several communications protocol processing techniques that increase processing speed with a high degree of multiplexing. They were designed with the goal of creating a video-on-demand system. The data transfer rate was measured for a prototype communications board using these techniques connected to several workstations over an FDDI network. The main results were as follows:

- By using our techniques, we achieved a single-connection TCP/IP protocol processing performance within the board of 96 Mbps, with a throughput peak of 91.6 Mbps.
- The time needed to process the TCP/IP was reduced by 21–64% as a result of using assembly and disassembly of data packet header and contents in hardware and PDU management using page management.
- By using an on-the-fly checksum calculation circuit, the speed of checksum calculations in TCP protocol processing was increased 14-fold over that of conventional methods.
- By rescheduling protocol processing, performance degraded by only 2% during multiprocessing, even with 64 simultaneous connections, and the throughput drop with 64 connections was 7% better than without rescheduling.

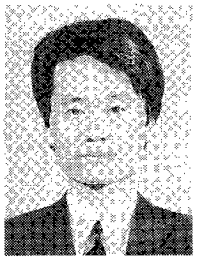
Our techniques are not restricted to use with FDDI networks; they are also suitable for connections using B-ISDN or over high-speed LANs.

Acknowledgments

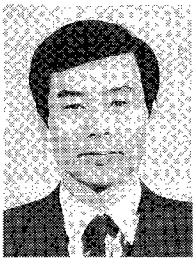
The authors thank all of their associates at the Multimedia Systems Laboratory for their assistance with this study.

References

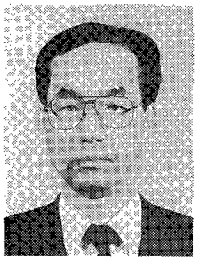
[1] D.D. Clark, V. Jacobson, J. Romkey, and H. Salwen, "An analysis of TCP processing overhead," IEEE Communication Magazine, pp.23–29, June 1988.
[2] C. Papadopoulos and G.M. Parulkar, "Experimental evaluation of SUNOS IPC and TCP/IP protocol implementation," IEEE/ACM Transactions on Networking, vol.1, no.2, pp.199–216, 1993.
[3] P. Druschel, M.B. Abbott, M.A. Pagels, and L.L. Peterson, "Network subsystem design," IEEE Network Magazine, vol.7, no.4, pp.8–17, 1993.
[4] C. Dalton, G. Watson, D. Banks, C.Calamvokis, A. Edwards, and J. Lumley, "Afterburner: A network-independent card provides architectural support for high-performance protocols," IEEE Network Magazine, vol.7, no.4, pp.36–43, 1993.
[5] K. Nishimura, T. Mori, Y. Ishibashi, and N. Sakurai, "System architecture for digital video-on-demand services," Proc. IEEE ICIP '92, pp.602–606, 1992.
[6] T. Mori, K. Nishimura, Y. Ishibashi, and H. Nakano, "Video-on-demand system using optical mass storage system," JJAP, vol.32, Part 1, no.11B, pp.5433–5438, 1993.
[7] H. Sakamoto, K. Nishimura, Y. Ishibashi, and H. Nakano, "Multimedia integrated switching architecture for visual information retrieval systems," Proc. IS&T/SPIE Electronic Imaging '93, vol.1908, pp.123–132, 1993.
[8] S.J. Leffler et al., "The Design and Implementation of the 4.3BSD UNIX Operating System," ISBN 0-201-06196-1, Addison-Wesley, 1989.
[9] M. Maruyama, H. Sakamoto, Y. Ishibashi, and K. Nishimura, "High-speed hardware architecture for high-definition videotex system," Journal of Electronic Imaging, vol.1, no.4, pp.349–357, 1992.



Mitsuru Maruyama Senior Research Engineer in the Global Computing Laboratory, NTT Software Laboratories. He is currently studying fast-protocol processing systems. Since joining the ECL system in 1985, he has been engaged in research and development of a high-definition videotex system and video-on-demand systems. He received an M.S. degree in electrical engineering from the University of Electro Communication in 1985. He is a member of the IEEE Computer Society and the Information Processing Society of Japan.



Kazutoshi Nishimura Research Group Leader in the Multimedia Systems Laboratory, NTT Human Interface Laboratories. He is currently engaged in research on video-on-demand systems. Since joining the ECL system in 1973, he has been involved in research and development of a magnetic tape recording system and an optical recording system. He received his B.S. degree from Kumamoto University in 1973. He is a member of the Japan Society of Applied Physics.



Hirotaka Nakano Executive Manager, Multimedia Systems Laboratory in the NTT Human Interface Laboratories. He is currently engaged in research and development on video information systems. Since joining the ECL system in 1977, he has been engaged in research and development of picture production systems and videotex systems. He received a Ph.D. degree from the University of Tokyo in 1977.