Architectures of Multimedia Communication Subsystems

Erich Rütsche
Institute Eurecom, Sophia-Antipolis
ruetsche@eurecom.fr

Matthias Kaiserswerth
IBM Research Division, Zurich Research Laboratory
kai@zurich.ibm.com

Abstract

This paper analyzes the particular requirements that multimedia communication imposes on the network adapter and the I/O subsystem of a workstation. We show the drawbacks of current (parallel) communication subsystems and develop new architectural concepts applicable to multimedia communication subsystems. The key ideas are the separation of isochronous and asynchronous traffic, parallelism between sender and receiver, and the execution of per-byte operations in a hardware pipeline. We adapt these concepts to the design of a Gb/s adapter and to the design of a light-weight, slower speed adapter and present some scenarios for their application.

1. Introduction

New types of networks such as FDDI, ATM, and FCS provide bandwidth in the range of 100 Mb/s to 1 Gb/s to a single workstation. Multimedia communication requires broadband communication of data, still images at high resolution, and moving pictures (video, animated graphics). The aggregation of these data generates not only high-volume continuous data streams but also dynamic bursts of data. An example of this is cooperative teleworking with video connections between multiple parties who also exchange high-resolution images, e.g., doctors holding a teleconference discussing X-ray images.

The communication of these data imposes a new type of load on the network, on the network adapter, and on the I/O subsystem of the workstation. Processing requirements of multimedia protocols are very different from the requirements of traditional transport protocols. Isochronous multimedia traffic requires the transmission of a large amount of data with low delay and bounded delay jitter but may accept relatively high bit-error or packet loss rates [Hehmann 89]. In a video, for example, bit errors or even lost frames are hardly visible, whereas a delay jitter of more than 10 ms is perceived as disturbing. Asynchronous traffic, for example file transfer or a remote procedure call, requires comparatively less throughput but tolerates no errors. In the conference example above, the video connection between the doctors could tolerate errors, whereas the X-ray images must be displayed error-free.

Conventional protocol processing on a workstation is limited by the available processing power and the I/O bottleneck [Ramakrishnan 93]. Communication subsystems based on single processors such as Nectar [Steenkiste 92a] or based on multiprocessors, such as the Parallel Protocol Engine (PPE) [Giarrizzo 89, Kaiserswerth 93] and similar architectures proposed by [Zitterbart 89], [Braun 92], and [Ulrich 93] were built to off-load protocol processing from the workstation. These systems were successful for transport protocol stacks at a network speed of up to 150 Mb/s. However, these approaches do not fulfill the quality of service requirements of real-time multimedia data streams transported in higher speed networks. A successful multiprocessor subsystem for continuous multimedia at moderate speed was demonstrated in [Blair 93].

In this paper we analyze the new requirements that the transmission of multimedia data impose

on the network adapter and the I/O system of the workstation. We show why architectures such as our PPE are not suitable for high-speed multimedia traffic, and hence develop two architectural concepts for multimedia communication subsystems.

In the next section we will try to assess the PPE architecture in light of these new requirements. In Section 3 we present the architectural concepts and what we believe are the key components of a new subsystem architecture. In Section 4 and 5 we describe the design of two multimedia adapters, one for Gb/s speeds and one for lower speeds, based on these concepts. Section 6 describes application scenarios for these new adapters. Section 7 is the conclusion.

2. The PPE Architecture

The Parallel Protocol Engine was developed to investigate parallel protocol implementation on a communication subsystem which would off-load protocol processing from a workstation to remove the protocol processing bottleneck.

The PPE uses two separate memories, one for transmitting, one for receiving data. Two or more processors (T425 transputers) are connected to each of these shared memories. Both shared memories are also mapped into the address space of the workstation.

In the implementation of the ISO 8802.2 LLC [Kaiserswerth 91] and of TCP/IP [Rütsche 92] on the PPE, the protocol processing performance was commensurate with the network bandwidth (120 Mb/s). The successful use of the PPE in parallel protocol execution was possible because the delay and overhead for interprocessor communication on the PPE was small compared to the execution times of the individual protocol functions. Parallelism was best exploited in full duplex communication between the receive and transmit side of the PPE. Pipelining the execution between the protocol layers TCP and IP was unsuccessful because of the vastly different processing requirements. The analysis of the measured execution times showed that a two-processor solution, with only one processor on the receive and one on the transmit side, would make more sense, as it would allow a speed-up of 1.7 over a single-processor implementation.

The architecture of the PPE using a separated receive and transmit data memory helped to reduce memory contention when simultaneously receiving and sending at network speed. The distributed shared memory for protocol state variables, which was implemented using transputer links between sender and receiver, had only a minor impact on the performance.

The protocol processing performance on the PPE was somewhat limited by per-byte operations such as copying. Computing the TCP checksum in software was switched off because of its performance penalty. For a next generation interface these per-byte functions must be supported by dedicated hardware.

The main problem of the PPE was to make its processing power available to the application on the host. The protocol processing performance of the PPE was in the range of 100 Mb/s whereas the host interface only provided a throughput in the range of 27 Mb/s. The poor performance of the host interface was due primarily to the interface hardware but also to the processing of the application programming interface on the host. The performance of this interface was clearly not sufficient for high-speed multimedia communication. For a subsystem connected to a higher speed network with multimedia requirements, the architecture must be changed because the hardware components such as processor and memories cannot be scaled to higher speeds.

3. Concept

For a new generation of communication subsystems the network I/O bottleneck must be considered first of all. The throughput of a network interface is limited mainly by the interrupt service time and the copy time of the I/O bus. Ramakrishnan showed that for a specific architecture a speedup of at most two can be achieved going from a non-programmable (shallow) to a programmable (deep) adapter [Ramakrishnan 93]. For multimedia communication, e.g., video, the required I/O bandwidth is higher than for classical computer communication. Thus for multimedia communication we must find a way to alleviate this I/O bottleneck.

We develop now the concepts for a new communication subsystem architecture that avoids the bottlenecks of the PPE and supports high-speed multimedia traffic. These concepts also build on the previous work of [Steenkiste 92b] and [Blair 93].

Separation of Isochronous and Asynchronous Traffic

The quality of service requirements of isochronous multimedia traffic and of asynchronous data traffic are very different. Separate paths for the processing of the two traffic types help provide sufficient resources to handle isochronous traffic. Protocol processing *and* the workstation interfaces are separated for both paths. A dedicated workstation interface for isochronous traffic is crucial because the workstation interface is the critical bottleneck in high-speed protocol processing.

Sender/Receiver Parallelism

Our TCP/IP implementation on the PPE showed that exploiting the parallelism between sender and receiver was the most successful approach in applying parallel processing to a protocol implementation. The communication between sender and receiver required in connection-oriented protocols can be accommodated with (the transputer's) low-speed serial message links. Splitting the memory into a sender and a receiver part reduces memory contention and allows the use of less expensive and slower memory devices.

Pipelining in hardware

Per-byte operations such as copying and computing a checksum limit the protocol processing performance. Therefore these operations should be implemented in a closely coupled pipeline of hardware devices that overlaps header processing. This approach has been implemented by [Kanakia 88] and [Steenkiste 92a].

3.1. Architecture Concept

Figure 1 shows the architectural concept. The communication adapter is split into similar send and receive sides. Both sides consist of a hardware pipeline and separate processing units and host interfaces for asynchronous and isochronous traffic.

On the receiver side, the pipeline demultiplexes asynchronous and isochronous packets arriving over the network. The network access unit implements the medium access control (MAC) functions and provides a frame-oriented interface to the next pipeline step. The protocol filter, described in detail below, is the key device that performs the demultiplexing functions. It analyzes protocol headers and extracts a unique connection number (CN). The checksum/CRC[1] unit uses the CN to calculate the appropriate check sequence required by the given connection. The check sequence is calculated on the fly as the data are passed to the DMA unit. The DMA unit on the receiver uses the CN to forward isochronous and asynchronous data to dedicated processing units. Each unit uses a dedicated interface to the

1. We will henceforth use the more generic term ***check sequence*** whenever we speak of a checksum as in TCP/IP or a Cyclic Redundancy Check sequence (CRC) as in OSI TP4.

workstation.

On the sender side, the pipeline multiplexes the two data streams to the network. Each processing unit gets its data from a dedicated workstation interface and forwards them to the transmit pipeline. In the pipeline, the check sequence is again calculated on the fly as the data is moved to the network access unit. The CN is used to determine which check sequence algorithm is to be used and where in the packet the calculated check sequence is to be stored.

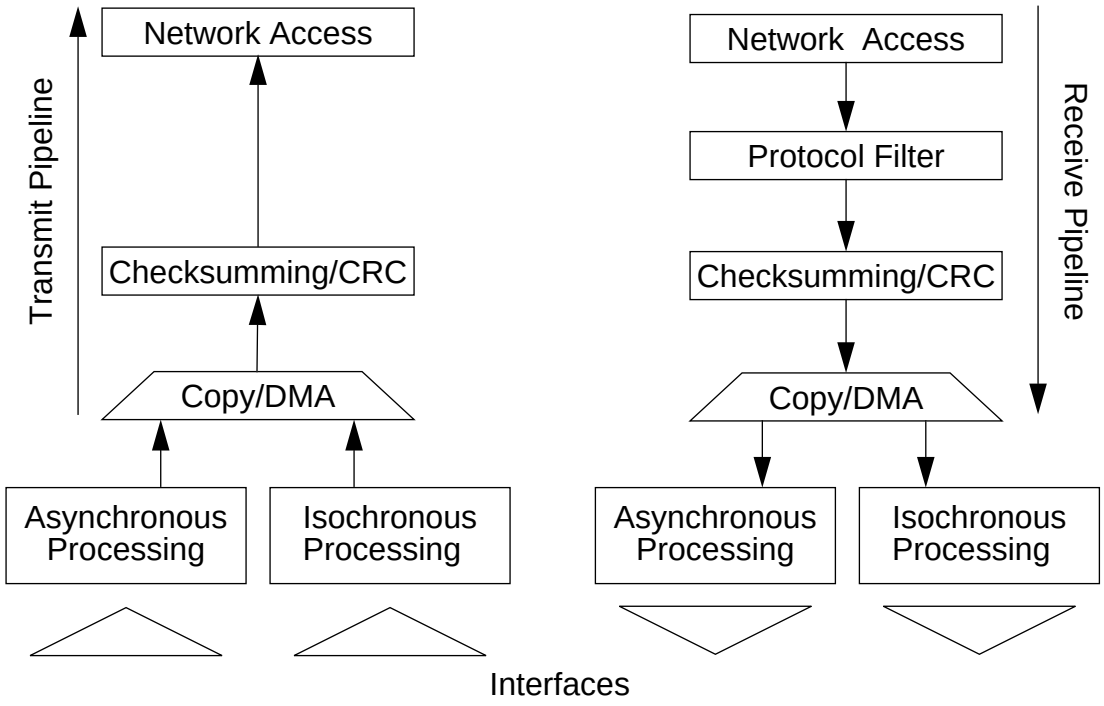

Figure 1. Architecture Overview

3.2. Protocol Filter

The Protocol Filter (PF) is a central device of the architecture needed to separate isochronous and asynchronous traffic. It scans each incoming packet header and extracts the information defining a connection. If a known connection is found, the protocol type and the connection number are returned. The connection number is a unique number that can be used as a pointer to the connection control information.

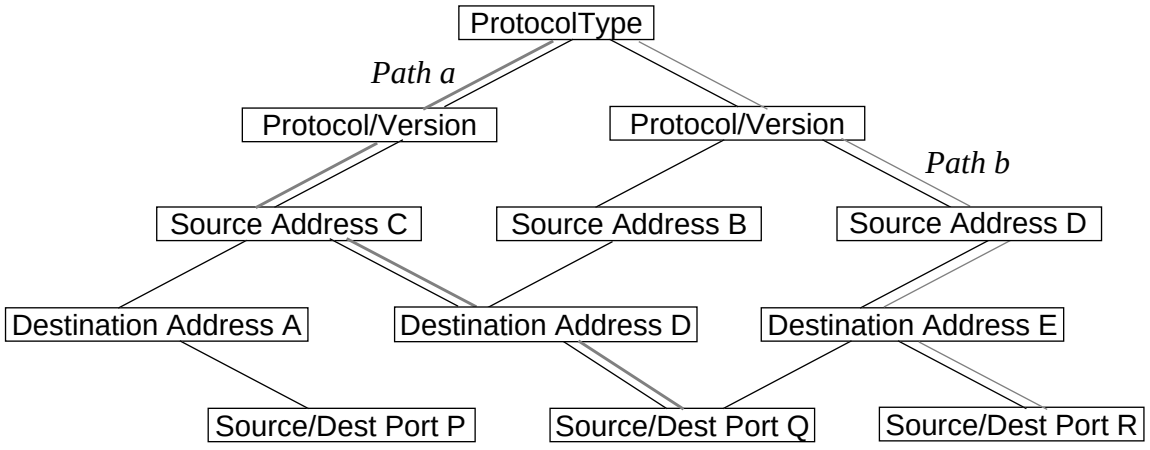

Figure 2. Protocol Address Tree Structure (Internet Protocols)

The addresses and protocol specific information used in the various layers of a protocol stack form a tree. A connection is defined by the path through the tree (see Figure 2). This path is stored in a content addressable memory (CAM). A CAM row contains the information of the tree level, the protocol type and the address information. For each branch of the tree, the CAM returns the address of the matched word. These addresses are concatenated to form a unique connection number.

The architecture of the PF is shown in Figure 3. The PF consists of two hardware state machines built around the central CAM. The Mask Generator extracts the relevant header fields from the protocol header and generates a mask that is compared in the CAM. The Connection Number Builder reads the addresses given by the CAM and builds a unique CN. As the connection detection works in O(1) time, the PF can be built to run at network speed. A detailed description of the PF is given in [Rütsche 93a].

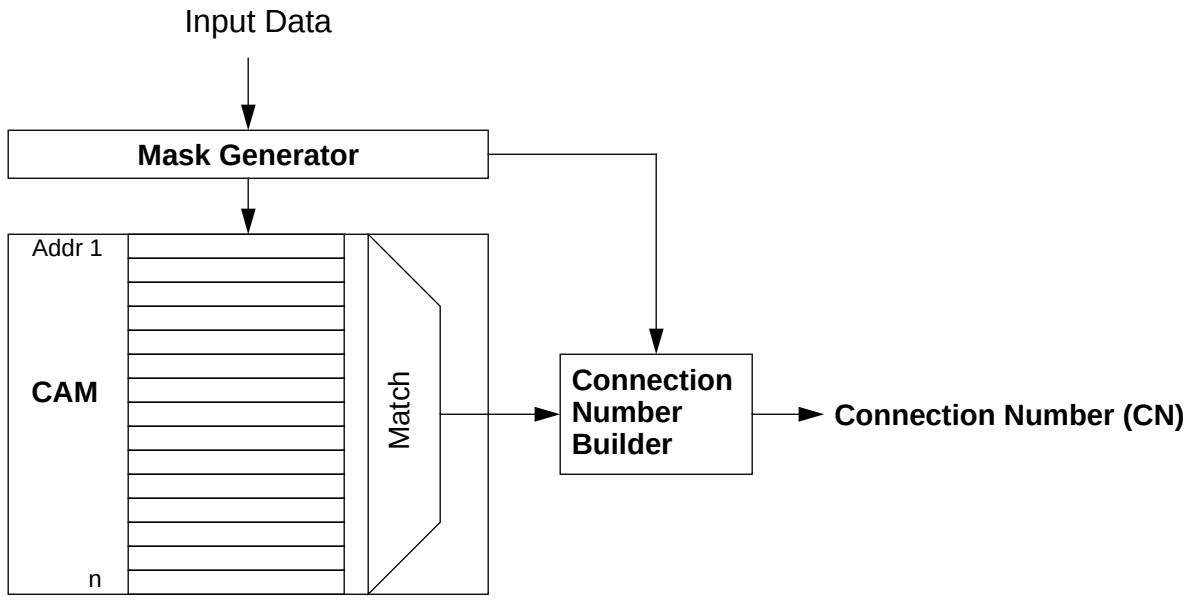

Figure 3. Protocol Filter

4. A Gb/s Adapter

In this section we describe the application of the presented architectural concepts to a Gb/s network interface. For such networks protocol processing must be performed on the subsystem, because current workstations are unable to cope with this processing load and run applications at the same time[1]. Therefore we propose an adapter architecture that implements protocol processing of isochronous and of asynchronous traffic on the adapter. The architecture of this Multimedia Protocol Adapter (MPA) for a 622 Mb/s network[2] is described in more detail in [Rütsche 93b].

For isochronous traffic, we consider the Internet Stream Protocol ST-II [Topolcic 90]. ST-II supports the delivery of multimedia data streams with a guaranteed quality of service. In the data transfer phase the protocol only requires the detection of the packet header without further processing. Therefore we do not need a dedicated processor for multimedia. Instead, the DMA

1. An optimized TCP/IP implementation on a workstation still takes a fixed overhead of 110 µs/data packet [Dalton 93].
2. 622 Mb/s would, for example, be the line rate of a STM-3 or SONET OC-12 ATM line.

unit forwards isochronous data directly to the multimedia host-interface. This interface is a multimedia FIFO connected to a multimedia bus or a multimedia device. Compression/ decompression is not supported on the adapter because we believe that these functions would not be required for Gb/s networks.

For the processing of asynchronous data traffic we foresee a protocol processor to support different protocols flexibly (e.g., TCP/IP, TP4/CNLP, or XTP).

The analysis of TCP/IP on the PPE showed that two protocol processors provide optimal speed-up. We therefore suggest that two processors be used that are required to process only asynchronous traffic. To achieve optimal performance we partition the memory system to guarantee the processors fast access to the most frequently used data. The memory is separated into a header memory that holds packet headers and a data memory that holds the payload of a packet. The header memory is closely coupled to the processor and can be accessed similar to a data cache without wait states. The processors need no access to the data memory in the normal data transfer phase because all per-byte operations are implemented in the hardware pipeline. Therefore the full power of the protocol processors can be exploited in the most common case.

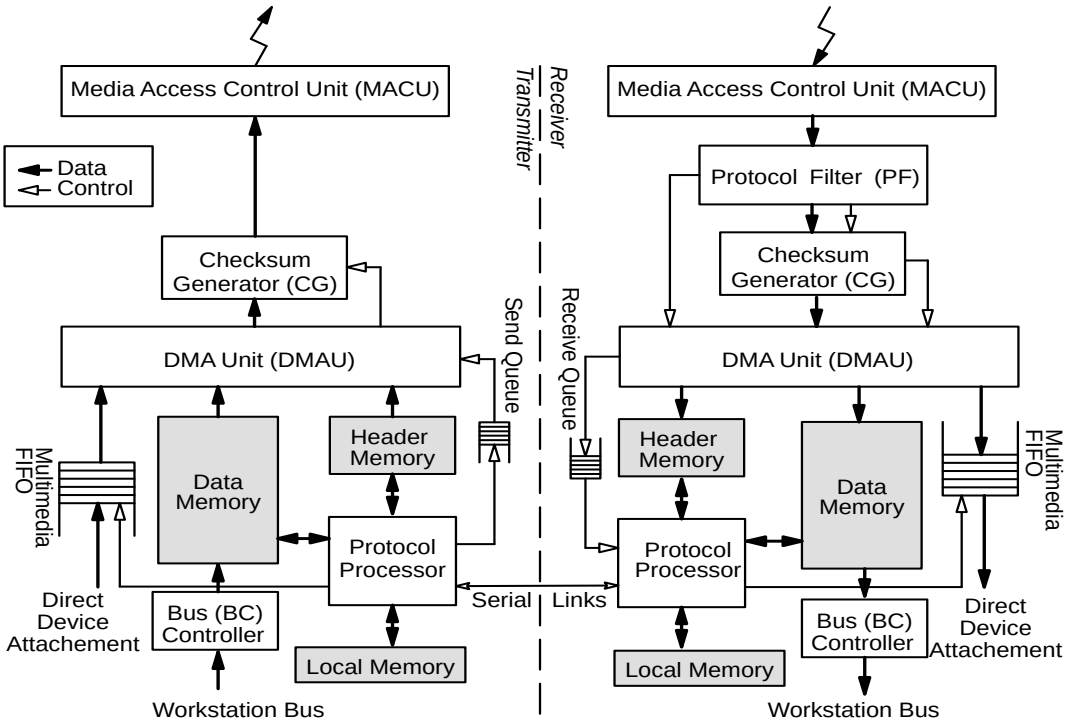

Figure 4. Architecture of a Gb/s Multimedia Adapter

The architecture of the MPA is shown in Figure 4. The pipelines are implemented as proposed in the architectural concept presented above. All the pipeline steps are implemented in hardware. The protocol filter parses the headers of incoming packets and generates a unique CN for each known connection. This CN is used by the following stages of the receive pipeline. The checksum/CRC generator (CG) uses the CN to calculate the appropriate check sequence of the packet. The DMA unit determines the packet format via the CN and splits it into its header and data parts. Isochronous data are written to a multimedia FIFO and the header is discarded. For each active connection a separate multimedia FIFO is provided to reduce queueing at the multimedia interface. For asynchronous data, the header is written to the header memory and the data part is written into the data memory. The pointer to the header is placed

in a receive queue. The DMA unit adds the calculated check sequence to the receipt information maintained in the header memory.

On the transmit side, the protocol processor writes commands to transmit a packet to the send queue. The DMA unit serves this send queue and triggers the CG to calculate the check sequence on the fly as the packet is written to the network access unit. The DMA unit gathers a packet header from header memory and the data from a multimedia FIFO or data memory.

For multimedia applications that require a reliable transport service, the protocols can be processed on the MPA. The processor then moves the data to a multimedia FIFO. The bus controller implementing the asynchronous host interface can be built with a powerful busmaster DMA controller or with an *Afterburner* interface [Dalton 93].

4.1. Performance Evaluation

The performance of this architecture can be evaluated by scaling our measurements of the PPE. From the measured execution times on the PPE, the execution times on the new architecture are estimated by excluding the number of instructions replaced by dedicated hardware and by then scaling the remaining instruction cycles with the faster processor (Inmos T9000 instead of T425) and improved memory interface. A detailed evaluation of this process is given in [Rütsche 93b].

The TCP/IP process pipeline for bidirectional traffic is shown in Figure 5. The throughput is limited by the receive processing of TCP (*tcp_recv*) and of IP (*ip_demux*), which add up to 26.7 µs. Transmit processing is also less costly than in the original PPE implementation because of the faster network speed.

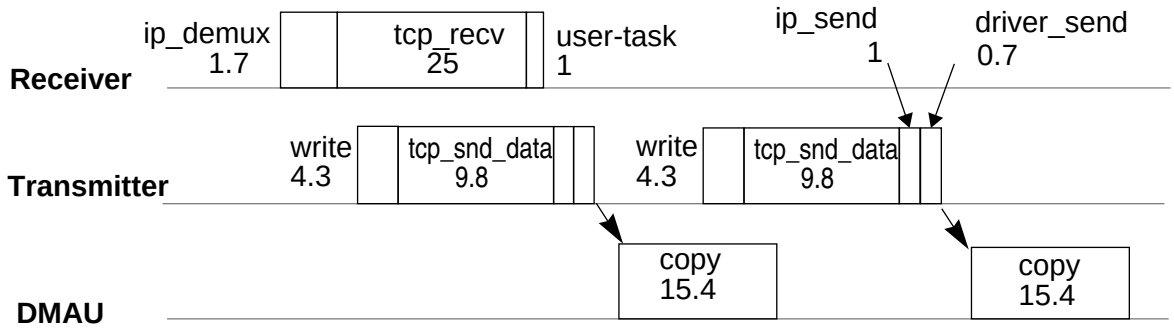

Figure 5. TCP/IP Execution Time (µs)

The throughput calculated for unidirectional TCP/IP traffic between two MPA systems is 35720 TCP segments/s. For bidirectional traffic, the throughput is 20290 segments/s if an acknowledgment packet is sent for every eight packets. The throughput numbers are independent of the packet size because all data copying is done in hardware overlapped with the protocol processing. However for large packets, the network speed becomes the bottleneck (4 kByte segments would require more than 1 Gb/s network throughput). If we assume a segment size of 1024 bytes, the throughput is 292 Mb/s, which is already more than most current workstations can handle.

For isochronous traffic, we assume the use of ST-II. In the normal data transfer only the connection information must be analyzed; no further processing is necessary. As the connection is detected in the PF hardware, ST-II can be processed at network speed.

5. A Light-Weight Adapter

At more moderate network speeds in the range of 100 - 155 Mb/s, the design criteria for adapters are different than for Gb/s speeds. Lumley [Lumley 92] argues that protocol processing on an adapter is too costly for commercial use. Traditional transport protocols can be processed at these speeds by the powerful (RISC) processor of a workstation. However the quality of service requirements of multimedia data still call for protocol support on the adapter. Therefore we propose an architecture that supports processing of isochronous multimedia traffic on the adapter but leaves the protocol processing of asynchronous traffic to the workstation.

One solution, the light-weight multimedia adapter (LWMA), is designed to be cost effective. The adapter performs only those protocol functions that are too costly for execution on the workstation, namely header parsing and per-byte operations such as check sequence calculation. The design of an ATM network interface based on these principles is shown in Figure 6. The Network Access Unit provides a frame interface to the PF and the DMA Unit. The PF is again used to analyze the headers of received packets. The packets and their CN are forwarded to a decoder. The decoder is a digital signal processor (DSP) on the adapter that implements the checksum/CRC unit of the architecture model. DSPs are suitable for this task because of their short interrupt latency. Their instruction set and cycle time allows fast frame check sequence calculation. For isochronous traffic the DSP can also be used to decode the compressed multimedia data in real-time. For asynchronous traffic the packets with the calculated check sequence and the CN are forwarded to the host.

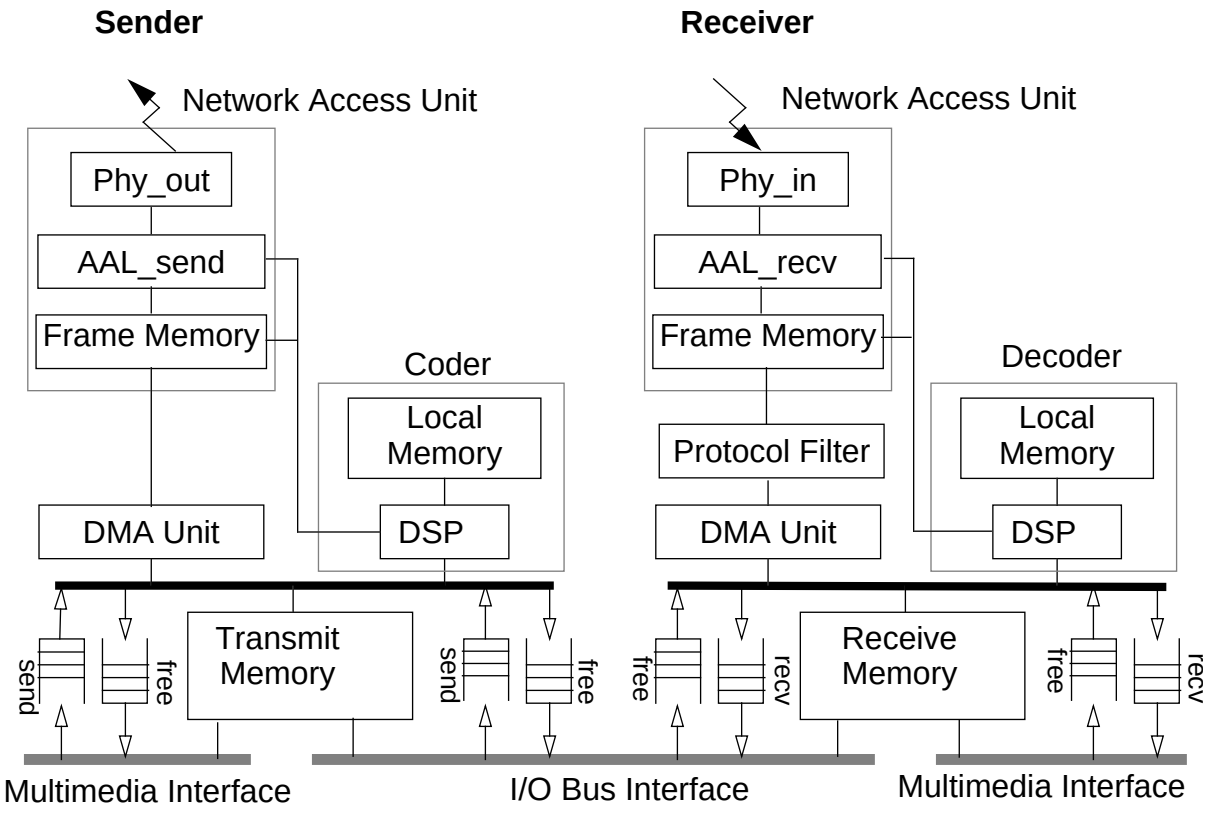

Figure 6. Architecture of the Light-weight Multimedia Adapter

On the sender side the DSP implements coding functions to calculate the check sequences for

asynchronous traffic or a compression algorithm for isochronous traffic. The DSP also acts as a controller for the network access unit. For example, for an ATM network, the network access unit would implement the ATM Adaptation Layer (AAL) protocols and the DSP processes the signalling protocol.

The host interface is also split into an asynchronous and an isochronous interface. It is similar to the Afterburner [Dalton 93]. To separate isochronous and asynchronous traffic we propose two interfaces with separate hardware queues implemented with FIFOs that control access to the memory of the sender and of the receiver. Thus asynchronous and isochronous data reside in the same memory. The fast memory can provide sufficient bandwidth to provide concurrent access from the interfaces, the DMA unit and the coder/decoder.

To send data an application gets a pointer to a buffer from the free buffer queue. It writes the data to the buffer in the transmit memory and writes the buffer pointer to the send queue. On reception the decoder gets a receive memory buffer from the free queue and writes the pointer to the processed buffer to the receive queue. Again, each active multimedia connection has its dedicated send and receive queues.

5.1. Performance Considerations

The hardware elements of the pipeline implemented on the LWMA run at network speed. A modern DSP can calculate a checksum and even a CRC in real-time. Video coding and decoding functions, e.g., JPEG or MPEG, are not yet possible at the required speed. Dedicated VLSI implementations of these algorithms could be used to provide these functions. Isochronous protocols and conversions could then be executed at network speed. The processing speed of asynchronous protocols is also improved by the architecture. Computing of the check sequence and header analysis, which make up a considerable part of the processing load, are implemented on the adapter and off-loaded from the host. The CN can also be used in protocol processing on the host. The CN can be used as a pointer to the protocol control information and for *protocol bypassing* [Thia 92]. In other words, a dedicated processing path is executed for a known connection.

The performance of asynchronous protocols like TCP/IP can further be enhanced by using a *Single-Copy Implementation* [Dalton 93] which can improve protocol performance by about a factor of two.

6. Application Scenario

Distributed multimedia applications change the requirements on a computer. As these requirements are so different, it is better to talk about a multimedia station whose task is not primarily computing but multimedia communication between a user and a high-speed network. For this purpose the architecture of the multimedia station must provide sufficient communication channels between the multimedia devices and the networking devices. Röthlisberger, for example, suggests the use of a dedicated multimedia bus that connects the network adapter to the multimedia devices [Röthlisberger 92]. The multimedia adapter architectures proposed above would fit well into such a system (see Figure 7).

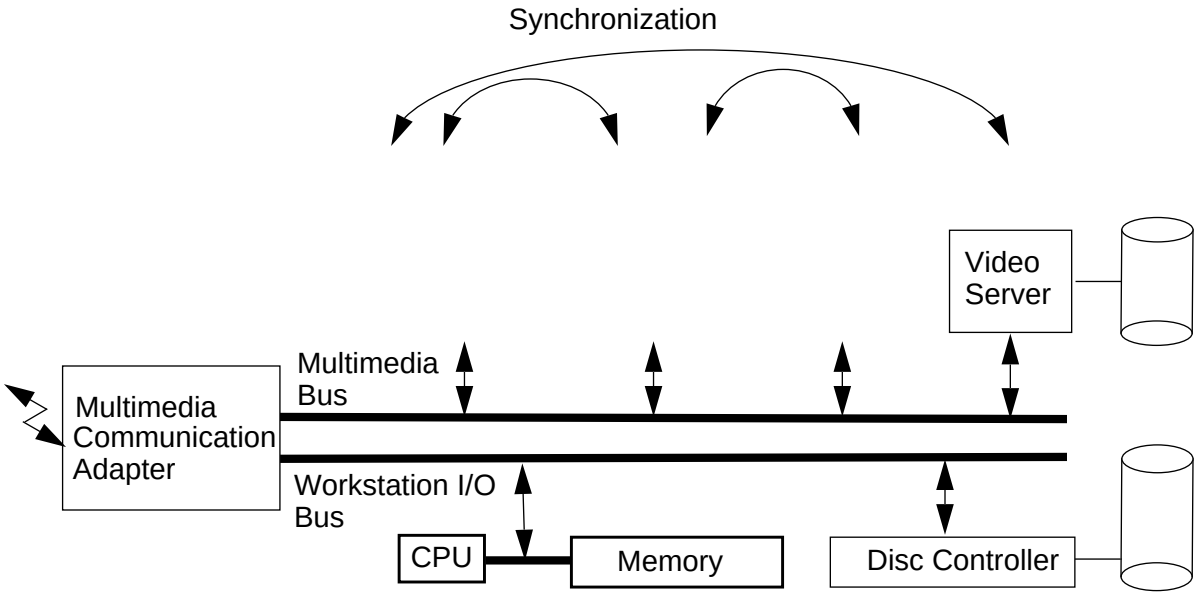

Figure 7. Application of a Multimedia Adapter in a Multimedia Station

The Lancaster Multimedia Network Interface (MNI) [Blair 93] uses a similar setup by connecting multimedia I/O devices directly to the communication subsystem. Protocols are executed on the subsystem. The workstation acts as a controller and is not concerned with the handling of the high-bandwidth multimedia streams.

While the MNI is a multimedia terminal, separate from the workstation, the presented multimedia communication adapters support a close integration of real-time multimedia into a multimedia station. Figure 7 shows the application scenario of a multimedia adapter. The separated host interfaces are connected to the I/O bus and the multimedia bus of a multimedia station. The multimedia bus connects all the multimedia end-devices and adapters. The multimedia adapter can provide coding and decoding functions that then need not be implemented on the device adapter. With the addition of coding/decoding hardware the multimedia adapter could be enhanced to a multimedia communication desktop collaboration system [Chen 92].

The multimedia communication adapter can synchronize networked data streams. The adapter processors delay the reception or transmission of data until a predefined synchronization point between the data streams is reached. However, for efficient real-time synchronization at high speed, hardware support would be useful. The synchronization with workstation-internal data streams may be supported by the multimedia bus, e.g., with a timing signal.

Processing of multimedia data that require a reliable transport service is performed on the asynchronous transport path. The potentially necessary retransmissions to make a service reliable do not allow isochronous service. Taking the example of teleradiology, the multimedia adapter would move the isochronous video directly between the network and the devices (digital camera or display memory). For X-ray data a reliable transport protocol, e.g., TCP/IP, would be processed before the image is written to the memory of a high-resolution display.

Application Areas

Complex architectures such as the MPA are necessary for high-speed multimedia. Applications of this type of adapter can be found in areas where high speed is truly required and where low price is not a primary requirement. Examples are video and multimedia servers, visualization stations and communication subsystems for supercomputers [Steenkiste 92b].

For applications where price is a major consideration, the LWMA is more suitable. For an ATM network the DSP on the adapter can implement the signaling protocol and the coder/ decoder functions. The additional hardware required to enhance an *intelligent* ATM adapter to a multimedia adapter is not too complex nor costly. If only isochronous multimedia support is required can the architecture of the LWMA be scaled even to Gb/s networks.

7. Conclusions

User-programmable communication subsystems were not very successful in the past because they were too expensive and complicated for widespread use. Careful protocol implementation on workstations was sufficient for most applications. However, for new classes of applications relying on multimedia data transported over high-speed networks, the current shallow subsystems are insufficient. For high-speed multimedia communications the adapter must provide support to guarantee a certain quality of service.

We have proposed a new architectural concept for multimedia communication subsystems that supports isochronous multimedia traffic. The header analysis in hardware is the key function that allows the early separation of isochronous and asynchronous traffic on the adapter. The two types of traffic use different processing paths and host interfaces: the standard I/O interface and a dedicated multimedia interface. Instead of using many parallel processors, we propose that closely coupled hardware pipelines be used that perform header parsing and all per-byte protocol functions. Parallelism is exploited primarily between the sender and receiver side of the adapter. For isochronous traffic, e.g., ST-II, all protocol processing functions are realized in hardware. How asynchronous data streams are processed depends on the network speed supported.

For Gb/s networks, the MPA is a powerful architecture that implements protocol processing on the subsystem. For lower speed networks (100 - 155 Mb/s), the LWMA offers support only for protocol processing functions, such as computing a check sequence and header parsing, that are too costly and time-consuming on a workstation. The protocol processing itself is performed on the host.

The proposed communication subsystem architecture addresses the network I/O bottleneck. However, for good integration of multimedia into workstations, the workstation architecture must also be changed and adapted to the high I/O bandwidth requirements.

8. References

[Blair 93] Blair, G., Campbell, A., Coulson, G., Garcia, F., Hutchinson, D., Scott, A., Shepherd, D., "A Network Interface Unit to Support Continuous Media," IEEE Journal on Selected Areas in Communication, Vol. 11, No. 2, Febr. 1993.

*[Braun 92]*Braun, T., Zitterbart, M., "Parallel Transport System Design," IFIP Conference on High Performance Networking, Liege (Belgium), 1992.

[Chen 92] Chen, M.-S., Shae, Z.-Y., Kandlur, D.D., Barzilai, T.P., Vin, H.M., "A Multimedia Desktop Collaboration System," in Proceedings GLOBECOM 92, IEEE, Dec. 1992.

[Dalton 93] Dalton, C., Watson, G., Banks, D., Calamvokis, C., Edwards, A., Lumley. J., "Afterburner," IEEE Network, July 1993.

[Giarrizzo 89] Giarrizzo, D., Kaiserswerth, M., Wicki, T., Williamson, R. "High-Speed Parallel Protocol Implementation," in Protocols for High-Speed Networks, Elsevier/North-

Holland, Amsterdam, 1989.

[Hehmann 89] Hehmann, D., Salmony, M., Stüttgen, H., "High-Speed Transport System for Multimedia Applications," in Protocols for High-Speed Networks, Elsevier/North-Holland, Amsterdam, 1989.

[Kaiserswerth 91] Kaiserswerth, M., "A Parallel Implementation of the ISO 8802.2 Protocol," Proceedings of the IEEE Conference on Communications Software: Communication for Distributed Applications & Systems, TriComm '91, Chapel Hill, NC, April 18-19, 1991.

[Kaiserswerth 93] Kaiserswerth, M., "The Parallel Protocol Engine," to appear in IEEE/ACM Transactions on Networking.

[Kanakia 88] Kanakia, H., Cheriton, D.R., "The VMP Network Adapter Board: High Performance Network Communication on Multiprocessors," ACM SIGCOMM 88.

[Lumley 92] Lumley, J., "A High-Throughput Network Interface to a RISC Workstation," IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems, Tucson , Arizona, Febr. 1992.

[Ramakrishnan 93] Ramakrishnan, K.K., "Performance Considerations in Designing Network Interfaces," IEEE Journal on Selected Areas in Communications, Vol. 11, No. 2, Febr. 1993.

[Röthlisberger 92] Röthlisberger, U., "A Network for a Multimedia Communication System," IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems, Tucson , Arizona, Febr. 1992.

[Rütsche 92] Rütsche, E., Kaiserswerth, M., "TCP/IP on the Parallel Protocol Engine," Proceedings, IFIP Conference on High Performance Networking, Liege (Belgium), Dec. 1992.

[Rütsche 93a] Rütsche, E., "Multimedia Communication Subsystems: Architectures, interfaces and Implementations", Ph.D Thesis ETH Zürich, Nr. 10228, VDI Verlag, Reihe 10, Nr. 257, Düsseldorf 1993.

[Rütsche 93b] Rütsche, E., "The Architecture of a Gb/s Multimedia Adapter," ACM Computer Communication Review, Vol. 23, No. 3, July 1993.

[Steenkiste 92a] Steenkiste, P.,"Analysis of the Nectar Communication Processor", Proc. IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems, Tucson, Az, Febr. 1992.

[Steenkiste 92b] Steenkiste, P., et al., "A Host Interface Architecture for High-Speed Networks", Proceedings IFIP Conference on High Performance Networking, Liege (Belgium), Dec. 1992.

[Thia 92] Thia, Y.H., Woodside, C.M., "High-Speed OSI Protocol Bypass Algorithm with Window Flow Control," 3rd. Int IFIP WG.6.1. Workshop on Protocols for High-Speed Networks, Stockholm, May 1992.

[Topolcic 90] Topolcic, C. (Editor), "Experimental Internet Stream Protocol, Version 2 (ST-II)," RFC 1190, Oct. 1990.

[Ulrich 93] Ulrich, R., "Untersuchungen an einem OSI-Kommunikationswerk auf Transputer-

basis," Dissertation, Arbeitsberichte des Institus für Mathematische Maschinen und Datenverarbeitung der Universität Erlangen-Nürnberg, Band 26, Nr. 11, Sept. 1993.

[Zitterbart 89] Zitterbart, M., "High-Speed Protocol Implementation Based on a Multiprocessor-Architecture," in Protocols for High-Speed Networks, Elsevier/North-Holland, Amsterdam, 1989.