The Architecture of a Gb/s Multimedia Protocol Adapter

Erich Rütsche
IBM Research Division,
Zurich Research Laboratory
Säumerstrasse 4, 8803 Rüschlikon, Switzerland

Abstract

In this paper a new multiprocessor–based communication adapter is presented. The adapter architecture supports isochronous multimedia traffic and asynchronous data traffic by handling them separately. The adapter architecture and its components are explained and the protocol processing performance for TCP/IP and for ST–II is evaluated. The architecture supports the processing of ST–II at the network speed of 622 Mb/s. The calculated performance for TCP/IP is more than 30000 segments/sec. The architecture can be extended to protocol processing at one Gb/s.

Keywords: Multimedia Communication Subsystems; Network Protocols; Parallel Protocol Processing

1. Introduction

As data transmission speeds have increased dramatically in recent years, the processing of protocols has become one of the major bottlenecks in data communications. Current experimental networks provide a bandwidth in the Gb/s range. New multimedia applications require that networks guarantee the quality of service of bulk data streams for video or HDTV. The protocol processing bottleneck has been overcome by dedicated communication subsystems which off–load protocol processing from the workstation. Many of such communication subsystems proposed in the literature are multiprocessor architectures [Braun 92, Jain 90, Steenkiste 92, Wicki 90]. In this paper we present a new multiprocessor communication subsystem architecture, the Multimedia Protocol Adapter (MPA), which is based on the experience with the Parallel Protocol Engine *(PPE)* [Kaiserswerth 92] and is designed to connect to a 622 Mb/s ATM network. The MPA architecture exploits the inherent parallelism between the transmitter and receiver parts of a protocol and provides support for the handling of new multimedia protocols.

The goal of this architecture is to speed up the handling of multiple protocol stacks and of multimedia protocols such as the Internet Stream Protocol *(ST–II)* [Topolcic 90]. Multimedia traffic often requires isochronous transmission in contrast to conventional asynchronous traffic for file transfer or for remote procedure call. To guarantee the isochronous processing of multimedia data streams, the asynchronous and isochronous traffic are handled separately. A Header Parser scans incoming packets, detects the header fields and extracts the header information. This information is used to separate isochronous and asynchronous traffic and to split the header and the data portions of a packet. Dedicated header and data memories are used to store the header and data portions of a packet. The separation of receiver, transmitter and the dedicated memories decreases memory contention.

In Section 2 the concepts of the MPA architecture are presented. Section 3 explains protocol processing on the MPA. In Section 4 the performance of the MPA is evaluated by adapting the measurements of our TCP/IP implementation on the PPE to the MPA architecture. The last section gives the conclusions.

2. Architecture

The architecture of the MPA is based on our experiments with the PPE [Kaiserswerth 92],[Rütsche 92]. The PPE is a four–processor system based on the transputer T425 with a network interface running at 120 Mb/s. On the separate transmit and receive side two processors, the host system and the network interface use a shared memory for storing and processing protocol data. Transmit and receive side are only connected via serial transputer links.

2.1 Concept

The protocol processing requirements of multimedia protocols are very different from the requirements of traditional transport protocols. Isochronous multimedia traffic may require the processing of bulk data streams with low delay and low jitter but may accept bit errors or packet loss. Asynchronous traffic, such as file transfer or remote procedure call, requires more moderate throughput but tolerates no errors. In a file transfer between a file server and a client the throughput is limited by the I/O bus and the disk speed. Errors in the data are not acceptable, whereas a bit–error in uncompressed video is not visible.

To guarantee the requirements of multimedia connections the processing of multimedia data must be separated from the processing of asynchronous data. A Header Parser detects the connection to which an incoming packet belongs. Multimedia packets are then forwarded to dedicated multimedia devices while other packets go through normal protocol processing.

Protocol processing must be done in software to handle a multitude of protocols. Only functions that are common to all or most of the protocols are implemented in hardware. The MAC layer for ATM and the ATM Adaptation Layer *(AAL)* must be implemented in hardware or firmware to achieve the full network bandwidth of 622 Mb/s.

Our measurements of TCP/IP on the PPE have shown that the processors were not equally loaded because of the different processing requirements of the protocol layers and because of the very high costs of the memory operations [Rütsche 92]. The loose coupling via serial links between the receive and a transmit part had only minor impact on the performance. An optimal speedup of 1.7 was calculated for two processors. Therefore we chose a two–processor architecture for the MPA. One processor on the transmit side is connected via serial links to one processor on the receive side. The processors are supported by an intelligent Direct Memory Access Unit and dedicated devices for header parsing and checksumming. The memory of both parts is split into a header memory and a data memory to lower memory contention. The two halves of the MPA are only connected by serial message links.

2.2 Main Building Blocks

The MPA is split into two parts, a receiver and a transmitter, as shown in Figure 1. The various components and their function are presented in the following.

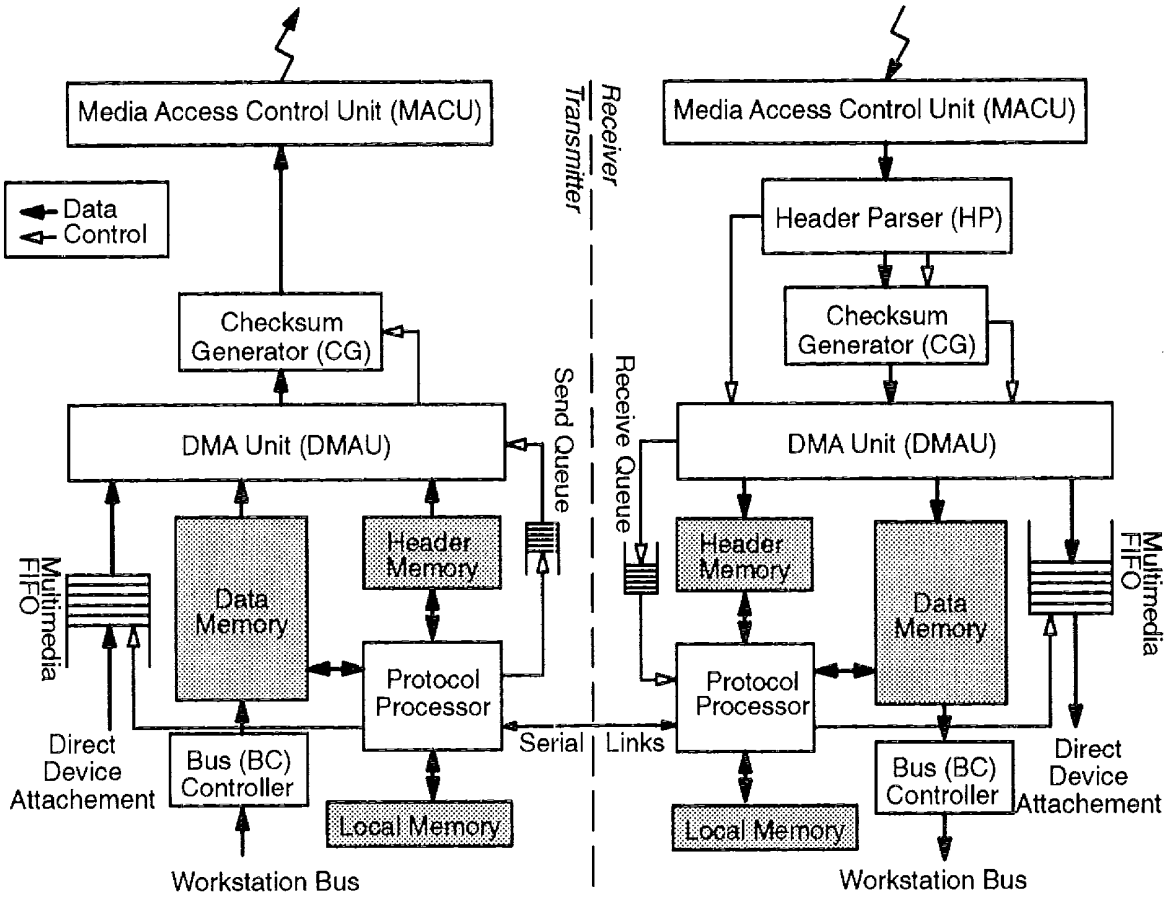

Figure 1. MPA Architecture

Media Access Control Unit (MACU): The MPA is designed to be connected to any high–speed network. The design of the MAC is beyond the scope of this paper. [Traw 91] for example describes an interface to the Aurora ATM network.

Header Parser (HP): The HP is similar to the ProtoParser[1] [Chin 92]. The HP detects on the fly the protocol type of an incoming packet and extracts the relevant header information. This information is forwarded to the DMA Unit and the Checksum Generator.

Checksum Generator (CG): The CG is triggered by the HP to calculate the appropriate checksum or Cyclic Redundancy Check *(CRC)* for the packet on the fly. The algorithms are implemented in hardware and selected by decoding the HP signal. On the sender side the CG is triggered by the DMA unit. [Birch 92], for example, describes a programmable CRC generator which is capable of processing 800 Mb/s.

The Protocol Processor T9000: The selection of the inmos[2] T9000 [inmos 91] is based on our good experience with the transputer family of processors in the PPE. The most significant improvements of the T9000 over the T425 for protocol processing are faster programmable link interfaces, a faster memory interface, and a cache. The serial message passing link provides a transmission speed of 100

1. ProtoParser is a trademark of Protocol Engines, Inc.
2. inmos is a trademark of INMOS Limited.

Mb/s plus a set of instructions to use the links for control purposes. The peek and poke instructions issue read and write operations in the address space of the second transputer connected to the other end of the link. These commands allow distributed 'shared memory' between transputers. Two transputers may allocate a block of memory at identical physical addresses in their local memory. Whenever a value is written into the local copy of the data structure, the address of the variable and its value are also sent via a control link to the second transputer.

The Memories: The memory is split into dedicated parts for each flow of data through the MPA to lower memory contention and to provide high bandwidth to those components that access the memory most. The following memory split is used:

***Header memory*:** stores the protocol headers. Fast static memory operating at cache speed is used to avoid wait cycles.

***Data memory*:** stores the data part of the packets. Inexpensive video memory *(VRAM)* is used. The serial port of the VRAM provides guaranteed access via the DMA Unit to the network. The parallel port of the VRAM is used in normal processing by the Bus Controller only. The processor can accesses the parallel port, e.g. for exception handling.

***Local memory*:** stores the program code of the processor and the control information of the connections.

***Multimedia FIFO*:** stores multimedia data and is the interface to a multimedia device. It can be controlled by the processor for synchronization with asynchronous data streams. Multiple multimedia FIFOs can be arranged in parallel.

The design does not employ physically shared memory between the transmitter and the receiver, because the implementation costs are too high compared to a software implementation using transputer links.

Memory Access; Processor to	Memory Type	Average Access Time
Data Memory	Video RAM	60 ns
Header Memory	Static RAM	30 ns
Local Memory	Dynamic RAM	60 ns

Table 1. Memory Access Time

Direct Memory Access Unit (DMAU): The DMAU directs the in– and outgoing data streams to the correct destination. The DMAU splits an incoming packet into its header and data part and moves the parts to the respective memories. A pointer to the header structure is written to the receive queue. To send a packet the DMAU gathers the data from the data memory and the header from the header memory. For multimedia traffic the data are gathered from the multimedia FIFO. The memory buffers are handled in a linked list format. The DMAU handles this linked list in hardware and thereby off–loads part of the memory management from the protocol processor.

Bus Controller (BC): The BC is a programmable busmaster DMA controller. It provides a small FIFO and a table for DMA requests. The FIFO contains a pointer to the linked list of source data and a connection identifier. The BC determines the destination memory address through the connection identifier in the table. The list format is the same for the BC and the DMAU. In the transmit BC the host writes to the

FIFO and the protocol processor to the table. In the receive BC the protocol processor writes to the FIFO and the host to the table.

2.3 Packet Processing

Packets are processed in a hardware pipeline which runs at network speed. The pipelined packet processing is shown in Figure 2.

Receiver

The MACU receives cells from the ATM network, processes the AAL, and triggers the receive pipeline to start. The receive pipeline is run by the DMAU. The HP and the CG process the data as they are copied from the MACU to the destination address in the memories or to the multimedia FIFO. The HP extracts the relevant header information from the packet and forwards the information to the DMAU and the CG. The CG uses this information to detect which checksum or CRC it must calculate. The CG calculates the checksum on the fly as the packet is copied by the DMAU and forwards the result to the DMAU. The DMAU uses the information generated by the HP to determine the format and the connection of the packet. For a multimedia connection the DMAU removes the header from the packet and writes the data part to the Multimedia FIFO.

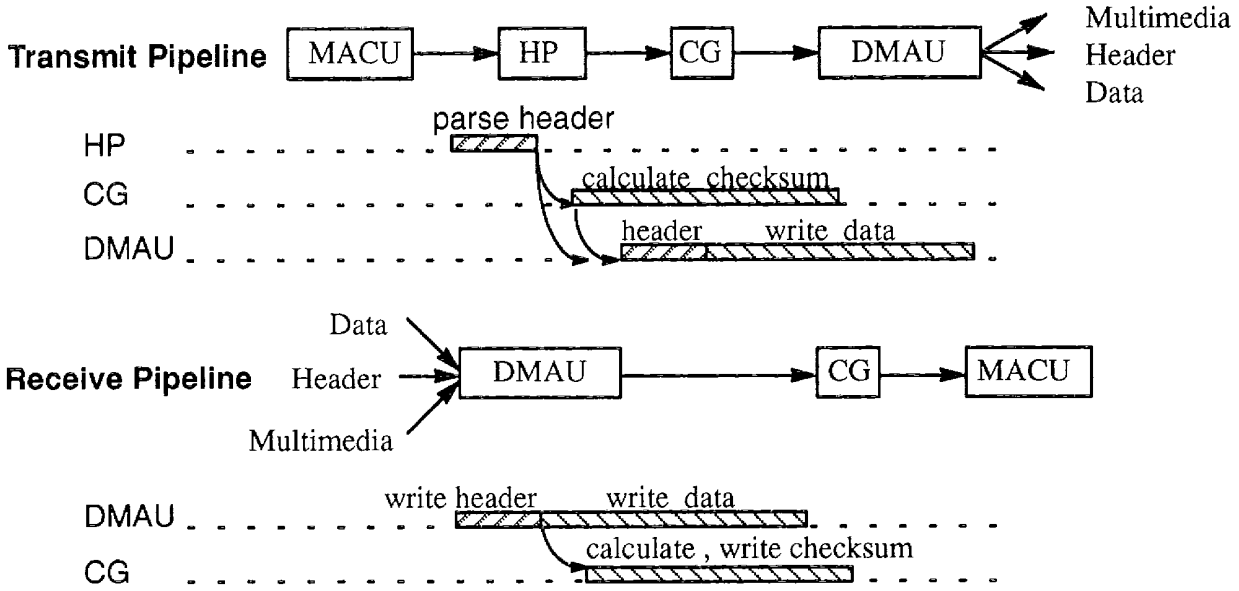

Figure 2. Pipelined Packet Processing

For asynchronous traffic the DMAU writes a structure to the header memory which holds the header, the header information extracted by the HP, the checksum calculated by the CG, and the pointer to the data in data memory. The data part of the packet is written to the data memory. The DMAU writes a pointer to the header structure to the receive queue. The protocol processor is then responsible for processing of the header structure. The addresses of free buffers in header and data memory are obtained from a linked list of free buffers.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.