OSI or Special Protocols for High Speed Networks ?

Torsten Braun, Burkhard Stiller
Institute of Telematics
University of Karlsruhe
Am Zirkel 2, P.O. Box 6980
D-W7500 Karlsruhe
Phone: +49 721 608-3983
Fax: +49 721 60 60 97
E-Mail: [braun|stiller]%i70vca@iravcl.ira.uka.de

Martina Zitterbart
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598, U.S.A
Phone: +1 914 784-6952
Fax: +1 914 784-6219
E-Mail: zit@watson.ibm.com

1. Introduction

In communication systems during the last few years, transmission rates of communication networks have evolved faster than the performance of protocol processing. Often nowadays, the bottleneck in communication systems is the protocol processing of layers above the media access layer.
Recently, applications such as multi-media-applications are coming up, which require very high performance communication systems. High data rates of more than 100 Mbit/s as well as low delays and delay restrictions according to isochronous data traffic are required. To provide the performance requirements of those applications there are two approaches:

- Existing protocols, e.g. OSI protocols or TCP/IP, are implemented efficiently by the use of special implementation techniques, hardware, or multiprocessor architectures.
- So called light weight protocols such as the Xpress Transfer Protocol (XTP) and the Versatile Message Transaction Protocol (VMTP) or special purpose protocols have been developed. Several mechanisms of conventional protocols are often considered as unsuitable for high speed networks. Moreover, conventional protocols often restrain efficient protocol implementations in hardware or on multiprocessor architectures.

This paper describes an implementation of XTP on a multiprocessor architecture and the simulation of VMTP on such an architecture. The protocols are evaluated under the aspects on usability for efficient implementation on multiprocessor architectures based on the performance results obtained from measurements and simulations. The results show that XTP and VMTP are more appropriate for implementations on parallel architectures than comparable OSI protocols.

2. An Overview of the Analyzed Protocols

2.1 VMTP

The realization of the protocol VMTP has been driven by the *client-server-model*. In this model the client addresses a *request-message* to a server. This server will return a *response-message*. The two used *data formats* for the request- and response-message differ in small parts only. The packet header and trailer are of fixed length. The maximum amount of transmitted data at a time will be restricted up to 4 Mbyte and will be divided into 256 *packet groups*, where each group will contain 32 *packets*. The combination of packet groups into a single transmission unit is called *run*, which simplifies the acknowledgements.
The exchange of these two specific messages is called *transaction*. Each communication system running the VMTP includes a *manager*, which is dealing with the transaction- and state-information management using the VMTP mechanisms recursively.
The VMTP does not include an explicit connection set-up and release phase. The transmission of a response-message establishes a connection implicitly. The returning of the last response-message releases this connection but does not lead to the throw-away of the connection state information. The possibility to maintain a connection forces the existence of a data structure called *client state record* (CSR). Sender and receiver store the connection state information for a dedicated period in the CSR. If a CSR does not exist in a communication node, the VMTP manager will solve the lack of connection state information. The so called *probing* mechanism will be initiated by the requested manager using the VMTP transmission protocol recursively. The CSR will be kept in the memory of a communication node for a long time to minimize the number of probings. Following transactions between the same communication users will use the old CSRs and do not have to set-up a new connection.
The *acknowledgement* of request-messages will be done by the returned response-message implicitly. Any following request-message will acknowledge the preceding response-message. The arrival of messages on time will be controlled using different server and client timers. The *rate-based flow control* will employ the interpacket gap to control the flow of data. The VMTP specification will be found in a partial incomplete and inconsistent version 0.7 in [1].

2.2 XTP

XTP [2] is a transfer protocol designed for the use in high speed networks from 100 Mbit/s to 1 Gbit/s, which has functions of the transport and the network layer according to the OSI reference model. A protocol processing hardware to support XTP is currently developed in the Protocol Engine Project [3].
The design of XTP was influenced by the following ideas:
The receiver part is simplified, because receiving is mostly the bottleneck in protocol processing. XTP offers a connection oriented service, but it is also appropriate for connectionless data transfer. Datagrams are supported by so called short-lived connections. Single bits in the packet header can cause connection establishment and release. Packet formats of fixed size for control and information packets are used.
XTP provides functionalities for the use as an universal protocol on the transport and the network layer. Multicast mechanisms as well as error, rate, and window-based flow

control are realized. Retransmission can be performed by the go-back-n or the selective method. Moreover, it supports several addressing mechanisms, e.g. OSI, Internet.
The XTP protocol architecture consists of a couple of finite state machines (FSMs), which are designed to run in parallel. There are several types of FSMs that perform different tasks. Some FSMs are multiplied for every context and some for every route. A context is a set of state variables that describes a connection state at one endpoint. A route is an identifier for a virtual channel comprising one or more XTP connections.
The FSMs cooperate on common data, e.g. context data, and by signalling events to other FSMs. FSM state transitions are triggered by those events and are dependent on current context data. The central FSM is the context manager state machine (CMSM) that becomes active when a new connection has to be created. Ten further FSMs are specified in [2]: output, synchronization, rate control, control send, input, reader, write close, read close, router state machine, and the route manager.

3. Protocol Simulations and Implementations

3.1 The VMTP Simulation Model

The VMTP simulation model [4] provides a slightly reduced functionality of the VMTP specification, because of the reduction of simulation time.

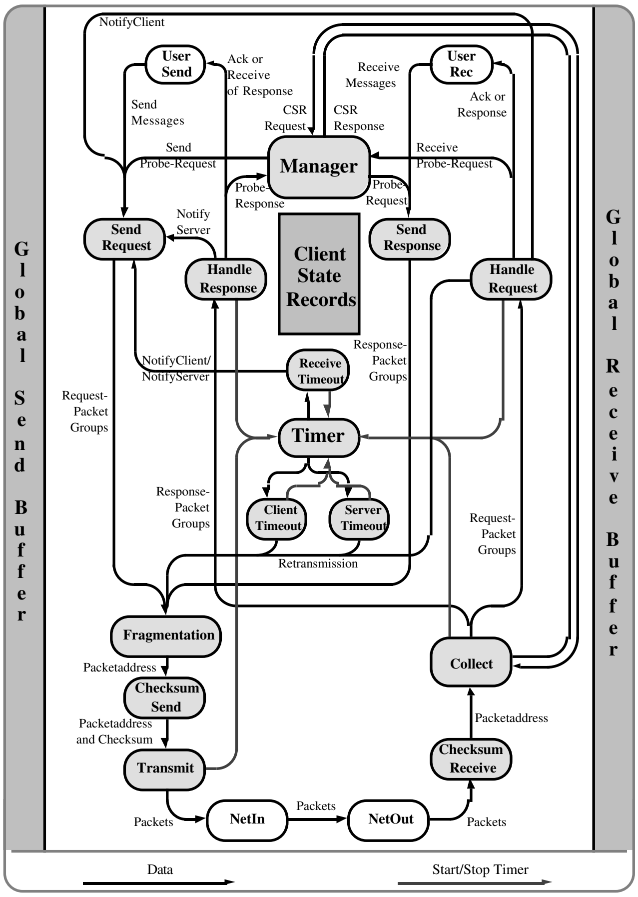

Figure 1: VMTP Simulation Process Configuration

But still the important mechanisms are supported; these are runs, probing, rate-based flow control using the interpacket gap, the optional checksum calculation, datagram service provided via the DGM-flag, manager based selective retransmissions, priorities, and different request codes. The simulation model allows the treatment of lost, altered, and garbled packets. The implementation of the address generating mechanism is mandatory, because the VMTP-manager and a VMTP-user are two different users of the VMTP in every communication node.
Simulation results will be presented to show the performance of the high-speed protocol VMTP. These results will be regarded for different situations, where two communication nodes are connected. The maximum throughput for two communication systems connected by a 10 Gbit/s link, representing a network that accepts data at almost every time, will be delineated exemplarily. The checksum calculation still influences the throughput heavily. For that reason, the throughput drops from about 285 MBit/s to 135 MBit/s for large messages. Effects concerning the probing mechanism will be pointed out.

3.2 XTP-Implementation on Transputers

The XTP implementation is designed for insertion into an implementation of a protocol stack on a multiprocessor architecture according to concepts presented in [5]. Movements of data packets between different layers in a protocol stack are avoided by providing a global data memory for all layers.

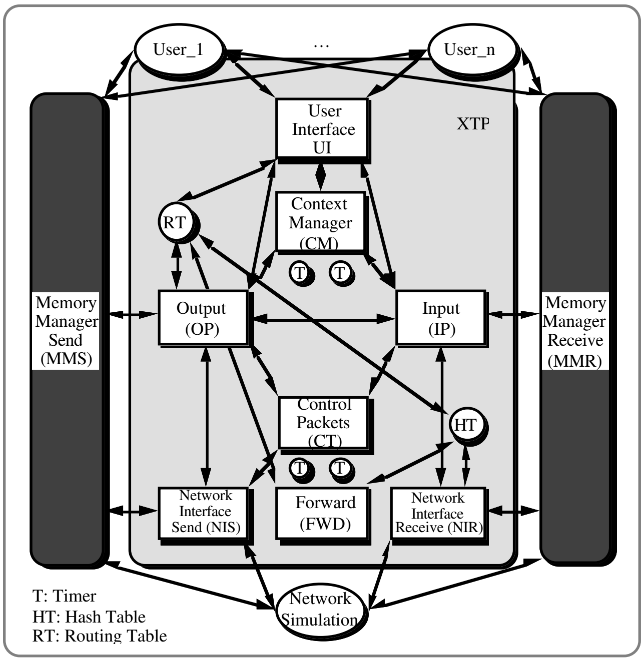

Figure 2: Parallel XTP-Implementation on Transputers

To implement the XTP FSMs always a separate process for every FSM type is created. This process is responsible for every created context or route. State transitions of FSMs are triggered by events and predicates. These predicates are mainly context data, which are common for several FSMs. Event signalling between processes is done by passing messages across channels. The different protocol processes operate on common context data. To allow all processes to run on different processors, the context data is distributed as well as possible to those processes that mainly operate on it. Context data needed by several processes have to be exchanged whenever any context values change.
The original XTP FSMs have been mapped to eight processes [6]: user interface (UI), input (IP), output (OP), forward (FWD), control packet (CT), context manager (CM) process, and two network interface processes for the send (NIS) and receive (NIR) parts. Additionally, there are several subprocesses for managing data structures such as hash and

route tables. Every process or subprocess can run on a single Transputer.
The XTP implementation is currently analyzed, some improvements are performed [7]. The most time consuming processes are the NIS process on send side and the NIR process on receive side. NIS takes ca. 850 µs and NIR ca. 750 µs for processing a single packet. These times include calculations as well as communications with other processes. Generation of events an communications have still to be improved further.
The maximum achievable throughput comes up to 1150 packets/s for sending and 1350 packets/s for receiving. Currently, these results are achieved nearly by configurations on eight transputers. For simplex traffic a speed-up of up to 3 can be achieved in comparison with the implementation on a single processor.
Especially, parallel processing of control packets and information packets is very efficient. Generation and parsing of control packets can be done nearly independent of information packet processing.

3.3 Implementations of OSI-TP4 and CLNP

In [5] an implementation of the OSI connectionless network protocol (CLNP) is described. It is shown that the throughput of CLNP implementation on eight processors is nearly four times higher than the throughput of the implementation on a single processor concerning duplex traffic. With this implementation a throughput of more than 2000 packets/s is achieved. The transport protocol class 4 (TP4) is implemented on a Transputer network also. The achieved performance (about 400 packets/s) was much lower, because the protocol is much more complex than CLNP. Implementations of the OSI protocols showed that mainly pipelined parallelism leads to performance gain. Spatial parallelism is mainly achievable by separating protocol processing in send and receive parts.

3.4 Comparison

By the use of parallelism considerable performance gains can be achieved. One reason for differencies in performance values of high speed protocols and OSI protocols is the use of different implementation techniques. Another reason is that XTP is suitable for parallelism, e.g. by the XTP's concept of several parallel finite state machines. In addition, XTP combines functionality of layer 3 and 4. Integration of several layers reduces layering overhead and supports parallelism.
Compared to OSI protocols high speed protocols such as XTP have similar functionalities. They differ in a few adapted mechanisms only. Some mechanisms are added to XTP in comparison with OSI protocols: The delay of connection establishment can be reduced by implicit connection set-up. Datagrams and connections are handled uniformly. Rate control can support the flow control function. Several retransmission methods can be selected. Furthermore, new protocols like XTP have functions such as multicast that are required by new applications. But the different protocol mechanisms and methods often have poor effect on performance increase only. The bottleneck is mostly protocol processing itself.
The suitable parallelism for the VMTP should be seen first between the receive and send processes. Second, the duplicating of specific slow functions, e.g. the checksum calculation, will lead to a better performance of the protocol. The impacts on the throughput considering different situations depends on the transmitted message size, the traffic type, or the existence of a CSR. To reach small delays the CSRs have to be stored for a long time. This requires memory at every communication node. The common header format of request- and response-messages forces a more simple and faster header analysis. For the future the impacts of each single mechanism will be evaluated in detail to argue for the achievable performance gains in particular. Comparing the VMTP to the OSI-Basic Reference Model the functionality of VMTP will be similar to the transport layer; different mechanisms had been chosen.

4. Conclusions

In general, the protocols VMTP and XTP are all purpose protocols. That is, no special optimization for a specific traffic type has been done and, therefore, no functional reduction has been dealt with. In comparison with OSI protocols they offer similar services and functionality. Main difference to OSI is the protocol architecture supporting efficient implementation on appropriate processing elements.
Especially the VMTP and the XTP are more suitable for parallel processing, although the offered functionality is not restricted compared to OSI protocols. The influence of the protocol processing elements is very important to achieve high performance. But protocols must be appropriate for fast processing elements, which can be realized in VLSI or by multiprocessor architectures.

5. References

[1] D. R. Cheriton: "VMTP: Versatile Message Transaction Protocol", Protocol Specification Version 0.7, Computer Science Department, Stanford University, 1988.
[2] Protocol Engines Inc. : "XTP Protocol Definition", Revision 3.4, 17 July 1989.
[3] G. Chesson: "The Protocol Engine Chipset", Transfer Newsletter, Vol. 4, No. 3, May/June 1991.
[4] J. Weigl: "Ein VMTP Simulationsmodell mit SIMSCRIPT II.5", Diplomarbeit, Universität Karlsruhe, März 1991.
[5] M. Zitterbart: "High Speed Transport Components", IEEE Network, January 1991.
[6] A. Schörk: "Parallele Realisierung von XTP auf Transputern", Diplomarbeit, Universität Karlsruhe, Oktober 1990.
[7] Th. Weber: "Untersuchung und Adaption des XTP-Protokolls auf Transputer-Netzwerken", Diplomarbeit, Universität Karlsruhe, Juli 1991.