Multimedia Transport and Teleservices,
International COST 237 Workshop, Vienna, Austria, November 13 - 15, 1994;
Lecture Notes in Computer Science 882, Springer-Verlag, pp. 219-240

Support for High-Performance Multipoint Multimedia Services

Georg Carle, Jochen Schiller, Claudia Schmidt
Institute of Telematics, University of Karlsruhe,
Zirkel 2, 76128 Karlsruhe, Germany
Phone: +49 721 608-[4027,4021,3414], Fax: +49 721 388097
Email: [carle,schiller,schmidt]@telematik.informatik.uni-karlsruhe.de

Abstract. Existing and upcoming distributed multimedia applications require highly diverse services to satisfy their communication needs. Service integrated communication systems should be capable of providing high-performance real-time multipoint communication service with guaranteed quality of service (QoS). Existing communication systems and known strategies for resource reservation face increasing difficulties in fulfilling these requirements, in particular in high-speed wide area networks. Therefore, new concepts are required to support the variety of emerging applications in a heterogeneous internetworking environment. In this paper, a framework for real-time multipeer services is presented. It is based on the separation of service requirements into network bearer and transfer service requirements. The transfer service enhances the network bearer service in order to meet the service requirements of the applications. Applications and transport systems interact using an enhanced service interface, which offers several QoS parameters. The transfer service is supported by transfer components (layer 2b-4 protocol functions in end and intermediate systems), resource management functions and the integration of specialized VLSI modules for time-critical processing tasks. The hardware components can be parametrized and selected individually dependent on the required service. The transfer system guarantees specified service qualities by assigning processing resources to specific connections. Selection of the appropriate combination of components and their parametrization enhances the bearer service of the underlying network in order to provide the required multimedia service at the transport service interface. Guaranteed services are realized by the reservation of resources both in the network (for guaranteeing network bearer service) and in transfer components (for guaranteeing transfer service). The paper describes a general approach towards high performance multipeer services, and dedicated parts in more detail. Preliminary performance results of VLSI components are presented and compared with measurements of typical software implementations.

1 Introduction

In the recent years, several new distributed applications with diverse service requirements have been developed. Frequently, these emerging applications require both high performance as well as support of a wide variety of real-time communication services. High-speed networks, (e.g., ATM-based networks) are able to fulfill bearer service requirements by providing data rates exceeding a gigabit per second and supporting different kinds of services. However, the service required by applications usually differs from the bearer service. Protocol processing in the end systems is needed to enhance a basic bearer service in order to meet application requirements. Yet, current communication systems (including higher layer protocols) are even facing problems in delivering the available network performance to the applications and demand for enhancements. Different approaches [1], [2], [3], [4], [5], [6] on implementing high performance communication subsystems have been under-

taken during the last few years: software optimisation, parallel processing, hardware support and dedicated VLSI components. Some of the approaches deal with efficient implementations of standard protocols such as OSI TP4 or TCP. Others developed protocols especially suited for advanced implementation environments. The use of dedicated VLSI components is mostly limited to very simple communication protocols (e.g., [7], [8]).

Another issue in the evolution is the increasing importance of group communication scenarios. Upcoming applications, for example in the areas of computer-supported co-operative work (CSCW), distributed systems, and virtual shared memory systems require point-to-multipoint (Multicast, 1:N) as well as multipoint-to-multipoint (Multipeer, M:N) communication [9]. For a growing number of applications such as multimedia collaboration systems, the provision of a multicast service associated with a specific quality of service (QoS) is necessary. If multipoint communication is not supported by the network or by the end-to-end protocols, multiple point-to-point connections must be used for the distribution of identical information to the members of a group. The use of multicasting is beneficial in various ways: It saves bandwidth, reduces processing effort for the end systems and the mean delay for the receivers, and furthermore, simplifies addressing and connection management. Various issues need to be addressed in order to provide group communication services in high-speed networks [10], [11]. Intermediate systems need to incorporate a copy function to support 1:N connections. Communication protocols must be capable of managing multipoint connections, and group management functions need to be provided for the administration of members joining and leaving a group. Another key problem that must be solved for a reliable multipoint service is the recovery from packet losses due to congestion in the network nodes and end systems. In this paper, we present VLSI support that is targeted towards complex multicast protocols. Especially, time-critical and processing intensive functions, e.g., selective retransmission are provided by dedicated VLSI components.

This paper is organised as follows: Section 2 gives an overview of an integrated approach for multimedia communication in high-speed networks and presents the conceptual framework for integrating VLSI components for multicast functions into end systems and group communication servers. The framework is not limited to a certain protocol and allows the use of high-speed protocols with fixed size packet headers. Section 3 presents a performance evaluation of different scheduling mechanisms and of different error control strategies in multicast scenarios. Section 4 discusses the functionality of a dedicated connection processor for managing multicast retransmission and presents some preliminary complexity and performance results of the discussed component. Section 5 summarises the paper and points out some future directions.

2 Multipoint Multimedia Communication Systems

Advanced multimedia applications are sensitive to the quality of service parameters throughput, delay, jitter, and reliability. Furthermore, they need to concurrently process different data streams, e.g., audio, video, and conventional data communication. Each of these streams requires a different QoS combination. As not only point-

to-point but multipoint-to-multipoint communication models with several senders and receivers are needed, specific QoS parameters are associated with all members of a group or only with a subset of receivers.

There is a need to accommodate traditional communication systems to these new application requirements. Therefore, several new components have to be introduced, and existing components of the communication system have to be modified. In the following, the distinct components of an enhanced communication subsystem for multimedia multipoint services are described.

2.1 Enhanced Service Interface

Traditionally, a limited set of QoS parameters was used to describe the requested service, and often no enforcement of these parameters was provided. Moreover, most communication protocols targeted towards pure data transfer, like TCP/IP, offer only a reliable point-to-point service (with the single QoS parameter 'reliability'). The service interface of these protocols no longer reflects the requirements of upcoming multimedia applications. These applications have strong throughput and delay requirements, while they often tolerate a reduced reliability for some communication channels. In particular, for networks under heavy load, certain data losses may be better tolerated by humans than additional delay introduced by retransmissions.

Enhanced service models and interfaces are required in order to serve emerging applications. Applications and the communication system need a common language, so a source is able to specify the traffic characteristics of the flow and, in turn, the communication system guarantees a specific service. The QoS parameters throughput, delay, jitter, and reliability can be specified by a minimum, maximum, and average value. Applications prefer to specify their requirements in an application related syntax and semantics, which usually differs from the one used in the communication system. In this case, a mapping of the parameters used at the service interface valid for TSDUs (e.g., frames of a video source) to parameters inside the communication system valid for TPDUs, is necessary. Additionally, the requested service needs to be described by different service classes, which specify how the service parameters should be guaranteed. Service classes can be classified as deterministic, statistic, and best effort services. Deterministic services guarantee QoS parameters even in the worst case, statistic services guarantee QoS parameters with a given probability, and best effort services are targeted towards applications which have no specific requirements. Real time services based on statistical multiplexing achieve either statistical or best effort reliability. Fully reliable services need additional mechanisms for error correction and are therefore limited to statistical or best effort delay. The complete service is described in a *service contract*. In this contract, the application agrees to limit the traffic passed to the service interface, while the service provider is obliged to reserve the resources required to maintain the specified service quality.

2.2 Resource Management

To deliver a requested service to a particular connection, it is usually necessary to reserve certain resources in all involved communication systems for that connection [12]. Such resources are the bandwidth of a link, processing power, and buffer space.

Resource management is an important feature, not yet included in most existing communication systems, that describes the ability to create and maintain resource reservations. Resource management consists of three different parts:

- *admission control* and
- *reservations* during the connection establishment, and
- *assignment of resources* during the data transfer phase.

Because the resources of communication systems are finite, not all connection requests can be accepted. In order to meet all service commitments, a communication system must contain an admission control algorithm that determines if a connection request with specific QoS requirements can be accepted without violating already accepted requests or if it must be denied. If all admission control tests are positively passed, reservations are made.

During the data transfer phase the communication system must provide mechanisms to assign the resources dependent on the reservations to the connections. Resources can be classified as active and passive resources [13], where active resources execute a process and passive resources hold data of the process. Dependent on the selected service class, the passive resource buffer space can be used exclusively by one connection or can be shared among several connections. In this case it is important to determine which packets have to be dropped if there is no more buffer available. All active resources are exclusive resources and are assigned by a *scheduler*. A scheduler decides which packet is processed next. There exist many proposed scheduling algorithms with different capabilities.

Scheduling Algorithms. Recently, several scheduling disciplines have been proposed [14], [15], [16] and compared [17]. The algorithms differ in the ability to provide guarantees for the QoS parameters throughput, delay, and jitter. In particular for the delay parameter, considerable research has been devoted towards the development of methods for providing a provable analytic upper bound, which will be experienced by all data packets. However, in many cases a very high percentage of packets will observe a delay far lower than the computable worst case bound.

In [17] a classification of scheduling disciplines in *Sorted Priority Queue Mechanisms* and *Framing Strategies* is proposed. Sorted Priority Queue Mechanisms use a state variable for each connection. Upon the arrival of a packet from this connection, the variable is updated and the packet is stamped with the new value. Packets are served in order of increasing stamps. In Framing Strategies, the time axis is divided into periods of some constant length, each called a frame. Bandwidth is allocated to each connection as a certain fraction of the frame time. Sorted Priority Queues are flexible in reserving different delays and bandwidth combinations to connections, while framing strategies couple the allocation of delay and bandwidth. The advantages of framing strategies are an easier implementation and no need of a test for schedulability at connection establishment time. However, for several disciplines of both classes it has been shown that they provide throughput as well as worst case delay guarantees. Although these disciplines guarantee a delay bound, these delays are often very long. Experiments [18] have shown that traffic scheduled by a simple FIFO discipline provides significantly lower delays even for heavy traffic load. If the

implementation overhead of complex scheduling schemes is taken into account, FIFO seem to be a good trade-off for high performance networks.

In this paper we want to present preliminary simulation results of three different scheduling disciplines: FIFO, Virtual Clock [14], and a packet-by-packet version of Round Robin [16]. Virtual Clock is a Sorted Priority Queue Mechanism, which aims to emulate Time Division Multiplexing. Each packet is associated with a virtual transmission time, which is the time at which the packet had been transmitted under Time Division Multiplexing. Virtual Clock is able to guarantee a specified throughput. The packet-by-packet round robin server uses separate queues for each connection and serves in each round the specified bandwidth of a connection. Statistical multiplexing using FIFO scheduling may provide a satisfying mean delay, while allowing a simple implementation. However, its firm delay bounds may often be far longer than applications wish to accept [19]. More sophisticated scheduling algorithms allow to reduce the firm delay bounds, while leading to more complex implementations.

Protocols for Resource Reservation. To reserve resources, the service requirements have to be distributed to all involved end and intermediate nodes. Several special reservation protocols have been developed [12], [20], [21] during the last years. They are characterised by a connection oriented simplex concept. To establish a connection with service commitments, the reservation protocols have to distribute a data structure, the so-called *flowspec*, which describes both, the characteristics of the traffic sent by the source and the service requirements. The protocol entities in the involved communication systems activate the resource management entities, which decide to accept or deny the connection. Furthermore, reservation protocols define rules to modify service parameters or mechanisms to detect errors. To support emerging multimedia applications efficiently, the protocols are not only required to provide mechanisms for guaranteeing a requested service in point-to-point communications, but also to accommodate receivers of a multicast group. Specific problems occur if each of these receivers has different service requirements.

2.3 Reliable Services

Packet Loss due to Congestion. The dominant factor which causes high speed networks to discard packets is buffer overflow due to congestion. In packet switched networks, statistical multiplexing allows a high degree of resource sharing. Short periods of congestion may occur due to statistical correlations among variable bit rate traffic sources, resulting in buffer overflow. The probability for packet loss may vary over a wide range, depending on the applied strategy for congestion control. Packet losses due to buffer overflow are caused by superposition of traffic bursts. Therefore, they do not occur randomly distributed, but in bursts and show a highly correlated characteristic. If a reliable service has to be provided, mechanisms are required which are able to handle this type of error efficiently. For multicast connections, the problem of packet losses is even more crucial than for unicast connections. Collisions of the multicast connections with independent unicast connections may occur

independently at every output port of a node. Therefore, the packet loss probability in multicast connections will be higher than in unicast connections.

Error Control Mechanisms. For applications that cannot tolerate packet losses of the network, error control mechanisms are required. Error control consists of two basic steps: error detection and error recovery. For error recovery, two mechanisms are available: Automatic Repeat ReQuest (ARQ) and Forward Error Correction (FEC). Error control is difficult in networks that offer high bandwidth over long distances. High data rates in combination with a long propagation delay result in high bandwidth-delay products. A large amount of data may be in transit. For example, at a distance of 5000 km and a data rate of 622 Mbit/s, more than 2 MByte may be stored by the link. This causes problems for the following reasons:

- End-to-end control actions require a minimum of one round-trip-delay, and retransmissions require large buffers and may introduce high delays;
- Efficient error control with timer-based loss detection is difficult, because delay variations do not allow very accurate timer setting, causing deterioration of the service quality;
- Processing of error control needs to be performed at very high speeds, if no bottle-neck is to be introduced.

ARQ Methods: ARQ (Automatic Repeat ReQuest) mechanisms are widely used in current data link and transport protocols. In every retransmission based scheme, the transmitter needs to store messages upon acknowledgement. At least the data of one round-trip delay needs to be stored. For go-back-N protocols, implementation of transmitter and receiver may be very simple, and no buffering is required for the receiver. For selective repeat protocols, implementation of transmitter and receiver is more complex, and a large buffer is required by the receiver. Processing overhead of ARQ methods is proportional to the number of data and acknowledgement packets that are processed. For point-to-point communication, ARQ mechanisms are well understood, and a number of protocols for data link layer or transport layer, employing these mechanisms, are known. For multicast communication, there are still many open questions concerning acknowledgement and retransmission strategy, achievable performance and implementation. Large groups require that the transmitter stores and manages a large amount of status information of the receivers. The number of retransmissions is growing for larger group sizes, decreasing the achievable performance. Additionally, the transmitter must be capable of processing a large number of control information. If reliable communication is required to every multicast receiver, a substantial part of the transmitter complexity is growing proportionally to the group size. In addition, individual receivers may limit the service quality of the whole group. To overcome these problems, a scheme that provides reliable delivery of messages to K out of N receivers may be applied (K-reliable service).

FEC Methods: Forward error correction (FEC) methods promise a number of advantages [22]. All FEC methods have in common that redundant information is transmitted to the receiver, together with the original information, permitting detection

and correction of errors. The delay for error recovery is independent of the distance, and large bandwidth-delay products do not lead to high buffer requirements. Therefore, FEC is a promising approach in high-speed networks. In contrast to ARQ mechanisms, FEC is not affected by the number of receivers. However, FEC has three main disadvantages when applied for error correction in high speed networks. It is computationally demanding, leading to complex VLSI components. It requires constantly additional bandwidth, limiting the achievable efficiency and increasing packet loss during periods of congestion. FEC achieves best performance for random errors. However, packet losses frequently occur in bursts [23]. If FEC is used for packets of variable length, parity packets will be calculated for data units of fixed size, leading to inefficient use of parity information.

The question when to apply FEC for real-time applications in high-speed networks requires extended assessments of various trade-offs. While investigations of FEC are subject of our ongoing work, this paper concentrates on multicast error control by ARQ protocols.

2.4 Conceptual Framework for Support of Real-Time Multicast Services

A conceptual framework was developed that allows to select the mechanisms best suited to provide a specific service. Depending on the required combination of delay and reliability for a specific scenario with given distances, and depending on the targets for utilisation of network resources, different strategies are appropriate to achieve the most appropriate trade-off. These strategies differ in the mechanisms for scheduling and resource reservation, and in the error control mechanisms that are applied.

While the integration of specialised multicast components into the end systems represents an important step towards a high performance reliable multicast service, further improvements of performance and efficiency may be achieved by the integration of dedicated servers in the network that provide support for group communication. In many cases of multicasting, the achievable throughput degrades fast for growing group size. A significant advantage can be achieved if a hierarchical approach is chosen for multicast error control.

Group Communication Servers. Figure 1 presents a network scenario with multicast mechanisms in the transport component of end systems and in dedicated servers. The proposed Group Communication Server (GCS) integrates a range of mechanisms that can be grouped into three main tasks:

- Provision of a high-quality multipoint service with efficient use of network resources;
- Provision of processing support for multicast transmitters;
- Support of heterogeneous hierarchical multicasting.

For the first task, performing error control in the server permits to increase network efficiency and to reduce delays introduced by retransmissions. Allowing retransmissions originating from the server avoids unnecessary retransmissions over common branches of a multicast tree. For the second task, the GCS releases the burden of a transmitter that deals with a large number of receivers, providing scalability. Instead

of communicating with all receivers of a group simultaneously, it is possible for a sender to communicate with a single GCS or with a small number of GCSs. The servers may be used hierarchically, where every server provides reliable delivery to a subset of the receivers. Integrating support for reliable high performance multipoint communication in a server allows better use of such dedicated resources. The servers will exploit possible multicast capabilities of the bearer service. For bearer services that do not offer multicasting, it is preferable to integrate a copy function into a server instead of integrating it into the transmitter. If multiple transmitters are connected to a server and a single multicast connection is used originating from the server, the transmitters will receive a copy of their own transmissions. To avoid this, multiple multicast connections originating from the server must be used. For the third task, a GCS may use the potential of diversifying outgoing data streams, allowing conversion of different error control schemes and support of different service qualities for individual servers or subgroups. The group communication server may offer a large range of error control mechanisms. For end systems, it is not required to have VLSI components for multicast error control. It will be sufficient to have access to a local GCS for participation in a high performance multipoint communication over long distances. Then, the error control mechanisms of individual end systems have only negligible influence on the overall performance, as simple error control mechanisms are sufficient for communication with a local GCS. If a priority field is used in the frame format, the server is able to distinguish packets of different importance. One example application would be hierarchically coded video. For information with different importance, specific packets may be suppressed for certain outgoing links. This may be used to reduce unnecessary traffic in case some receivers in a multicast group do not need the complete information. As an example, colour information may be eliminated for receivers that do not use it. Additionally, it may be used to protect packets with higher priority against loss due to buffer overflow.

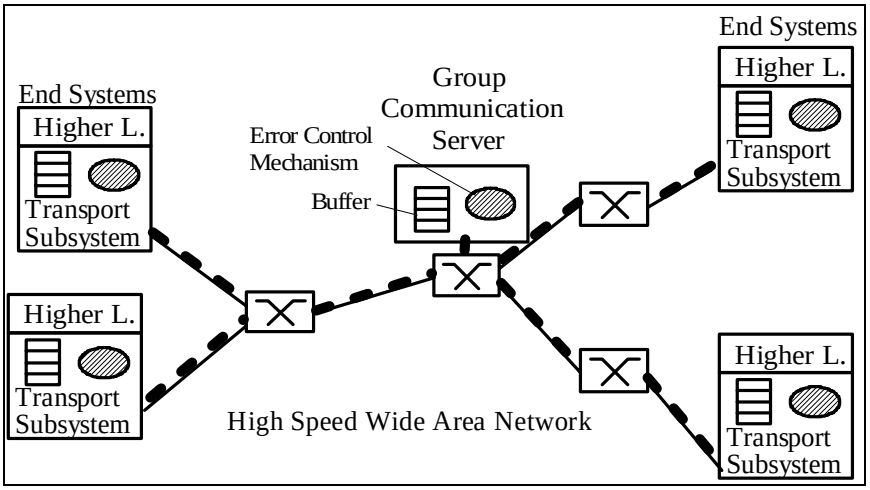


Figure 1. Real-time Multicast Support in Server and End Systems

Configuration and Reservation. While existing protocols for resource reservation are limited to exchange information required for the provision of a bearer service, our framework plans to extend this information by the choice of error control mechanisms that may be applied in intermediate and end systems. The following strategy is proposed for reservation of resources and selection of appropriate error control mechanisms: After a connection setup request is passed from the application across the service interface, the transmitter evaluates its resources that are required in worst case in order to fulfill the required service quality. It preallocates these maximum resources and passes the connection setup request together with the list of preallocated resources and supported error control mechanisms to intermediate and end systems. This allows the receivers to evaluate the combination of bearer service and transfer service best suited for the required service, based on the selection of the appropriate error control mechanism in combination with an economical reservation of resources. The result of the evaluation is passed back to the transmitter, resulting in a two-way handshake for minimum information exchange. Resources that were preallocated in excess are subsequently released. If the application requested to guarantee the QoS only to a subset of the receivers, an additional information transfer from transmitter towards the receivers may be performed, resulting in a three-way handshake for minimum resource allocation for QoS guarantees to K out of N receivers.

3 Performance Evaluation

3.1 Simulation of Scheduling Algorithms

It is of high importance to identify a scheduling algorithm that allows to guarantee a specific service quality, while efficiently using network resources. The following simulations allow to determine differences in the achievable performance. However, the final selection of a suitable scheduling mechanism will not only be determined by a performance evaluation based on simplifying assumptions. Implementation complexity leading to additional processing delay and to high system costs also needs to be considered.

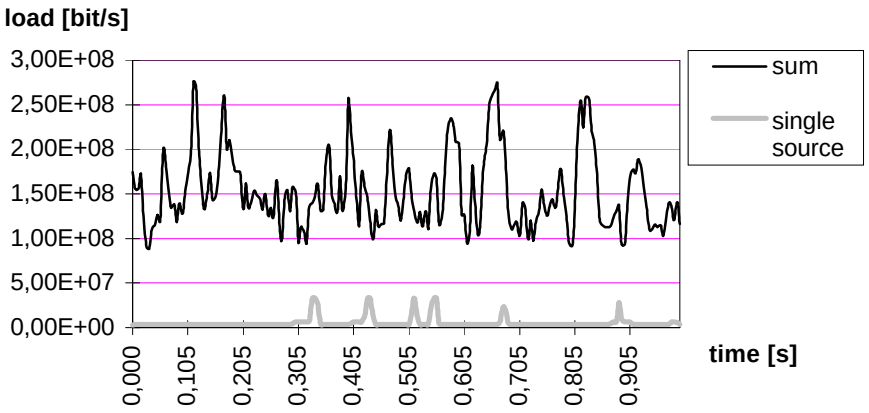


Figure 2. Simulated load (sum = 22 single sources)

A detailed comparison of the following three scheduling algorithms was performed: FIFO, Round Robin, and Virtual Clock. Simulations were performed to investigate the influence of the three scheduling algorithms on mean and maximum delay for real-time services. The sources for the simulation are modeled using Markov modulated Bernoulli processes with three states [24]. The sources represent a typical data stream produced by video applications. The minimum data rate of a source is 3.0 Mbit/s, the average data rate 6.3 Mbit/s, and the peak rate 30.0 Mbit/s. In the first step, the simulation model consists of a single node with several input and output links and only homogenous sources. An almost saturated output link was assumed. 19 or 22 sources were multiplexed for a load of 86% and 98% (155Mbit/s link), respectively. In the simulation, ideal processing delay of the scheduling algorithms was assumed, while the only time consuming process in the node was the sending of data.

The delay distribution of Virtual Clock, Round Robin, and FIFO shown in Figure 3 gives an example of the different behavior of the algorithms under the load shown in Figure 2. They show different end-to-end delay variance and end-to-end delay maxima.

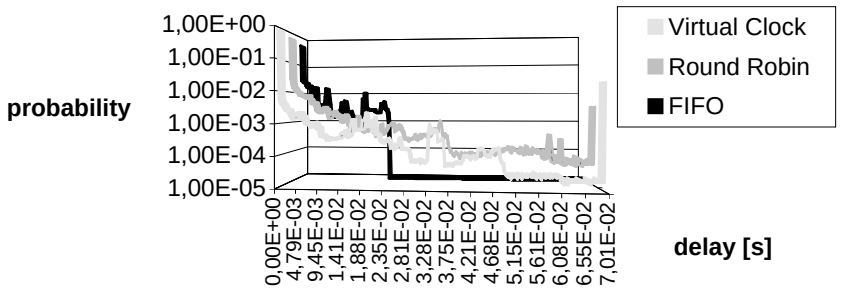


Figure 3. Delay distribution

Although none of the algorithms gives tight delay bounds, only Virtual Clock and Round Robin have a long tail in the delay distribution. This fact also can be seen in Table 1 and Table 2. Using Virtual Clock or Round Robin, most of the packets have a relatively low delay, but some packets have a very high delay. FIFO scheduling leads to a higher mean delay, while avoiding the very high delay of a small fraction of the packets that was observable for the other two scheduling algorithms. For a single node, the following tables present the fraction of packets delivered with the specified delay bound:

load	delay [ms]	algorithms		
		Virtual Clock	Round Robin	FIFO
86.40%	< 0.122	0.854	0.639	0.497
	< 1.0	0.868	0.701	0.574
	> 70.117	0.017	0.003	0
98.16%	< 0.122	0.730	0.379	0.146
	< 1.0	0.745	0.458	0.213
	> 70.117	0.030	0.011	0

Table 1. Delay distribution for different load

The Virtual Clock algorithm is based on packets with time stamps. A large maximum delay is the result of the sometimes fast growing gap between the virtual and the real clock. In these cases, timestamps of some packets are very large.

The simulations show that under certain conditions even very simple algorithms may have a satisfying behavior. The implementation complexity of a simple algorithm, such as Round Robin or, in particular, FIFO, is much lower than, for example, Virtual Clock. For the latter, complex search and insert operations for the packet queue have to be implemented.

delay [ms]	load	algorithms		
		Virtual Clock	Round Robin	FIFO

minimum	86.40% 98.16%	0.0054 0.0054	0.0054 0.0054	0.0054 0.0054
average	86.40% 98.16%	3.4585 8.1802	3.4585 8.9743	3.4585 9.1624
maximum	86.40% 98.16%	134.5072 182.1402	80.7198 113.5866	22.4315 31.2514

Table 2. Absolute delay of packets in a single node

Preliminary hardware designs showed that for all three algorithms, implementations for more than 1 million packets/s are feasible based on standard semi-custom VLSI technology. A detailed hardware design of a more complex type of round robin, the hierarchical round robin algorithm, was developed using the hardware description language VHDL. The implementation allows for a maximum of 1024 different connections, 16 levels of hierarchy, 64 slots per frame, up to 15 packets per connection queued in the node, and a maximum of 1024 packets over all. Gate level synthesis and simulation has shown that the implementation can easily handle more than 2.5 million packets/s for 1.0 µm CMOS technology, the area is 12696 gates for the control logic and 66784 bits of memory for control registers. Processing effort of the scheduling algorithms is per packet. Increasing the packet size will increase the achievable throughput, if the component for processing of the scheduling algorithm happens to be the bottleneck.

3.2 Delay Distribution for Retransmissions

Simulations were performed and analytical methods were applied in order to evaluate the influence of the proposed framework on delay and throughput for the envisaged multicast scenarios. The deployment of GCSs permits a significant delay reduction in case of packet losses and allows to increase efficiency substantially.

Figure 4 shows how a GCS may save one round-trip-time from transmitter to GCS for errors that occur between GCS and receiver. For a GCS serving many groups, extremely large retransmission buffers might be necessary to fulfill all retransmission requests. For GCSs with limited retransmission buffers, negative acknowledgements may be passed to the transmitter if requested packets are not available in the buffer of the GCS. Simulations were performed for a variable bitrate video source, a GCS with limited retransmission buffer, and a network element under congestion. For the simulation scenario, cf. Figure 5. Figure 6 shows the packet delay times observed by the receiver for a distance of 1000 km between transmitter and GCS, and a distance of 100 km between GCS and receiver. In the simulation, between 2000 and 2500 packets were transmitted for every encoded image. The majority of packets was delivered without retransmissions, while some packets retransmitted by the GCS observed a small delay, and other packets retransmitted by the source were significantly delayed.

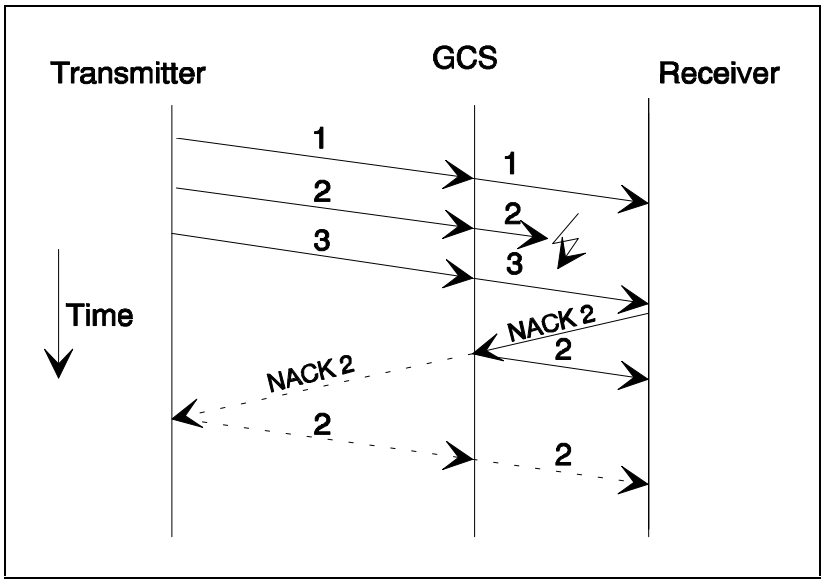


Figure 4. Delay Reduction by Retransmissions from GCS

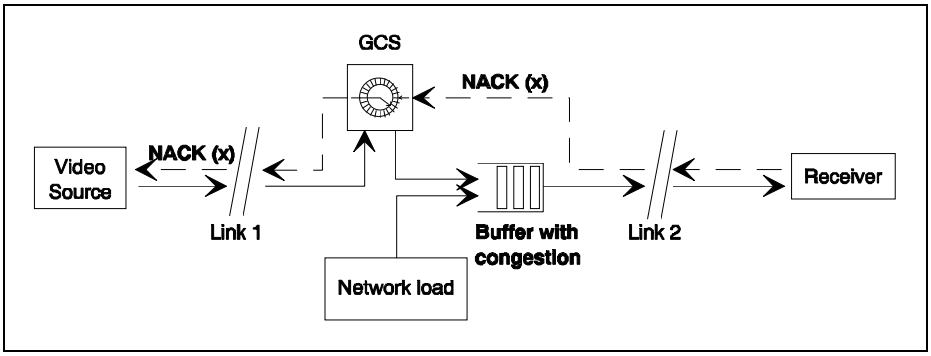


Figure 5. Simulation Scenario

3.3 Efficiency Analysis of Error Control Scenarios

Using Weldon's approximation and G/G/1 queuing models, the achievable performance of RMC (Reliable MultiCast)-AAL in selective repeat and go-back-N modes and the potential gain by deployment of GCSs was evaluated. The analysis is based on the following assumptions: protocol processing times may be neglected and acknowledgements are transmitted over a reliable connection. In correspondance with the results of [25], the throughput efficiency of a memoryless multicast go-back-N protocol may be expressed to

$$\eta_{1:N,GBN} = \frac{1}{\overline{m}} = \frac{1-Q}{1+sQ}$$

In this formula, $\overline{m}$ denotes the mean of the number of transmission attempts for successful transmission of a packet, Q denotes the loss probability of a packet, and s denotes the number of packets in transmission. In correspondance to [26] and [27], the efficiency of a selective-repeat protocol for a receiver buffer size of one path capacity may be expressed to

$$\eta_{1:N,SR} = \frac{1}{\overline{m}} = \frac{1-Q}{1+sQ^2}$$

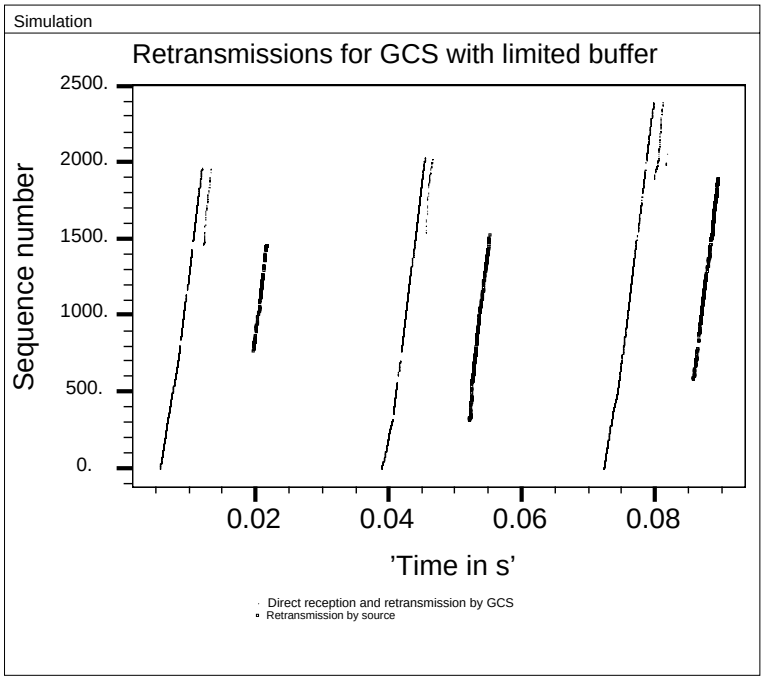


Figure 6. Reception with Retransmissions from GCS and Source

Figure 7 shows the efficiency of the two retransmission modes in three different scenarios. Scenario 1 represents a basic 1:N multicast without GCS. Scenario 2 represents 1:N multicasting with a GCS that performs retransmissions as multicast. In scenario 3, the GCS uses individual VCs for retransmission. The analysis is based on the following assumptions: protocol processing times may be neglected, acknowledgements are transmitted over a reliable connection, and buffers are sufficiently large. A group of 100 receivers and a data rate of 622 Mbit/s are assumed. Two cases are distinguished. The upper diagram of Figure 7 shows the efficiency for an overall distance of 1000 km (distance of 500 km from GCS to the receivers), and the lower diagram shows an overall distance of 505 km (distance of 5 km from GCS to the re-

ceivers). The analysis shows that in all cases, the efficiency is increased significantly by the GCS. Highest efficiency may be achieved for scenario 3 and selective repeat. Scenario 2 improves significantly for a shorter distance between GCS and the receivers. Go-back-N retransmissions show acceptable performance only for moderate bandwidth-delay products. Regarding efficiency, scenario 3 and selective repeat should be selected. However, this solution requires the highest implementation complexity for end systems and GCS.

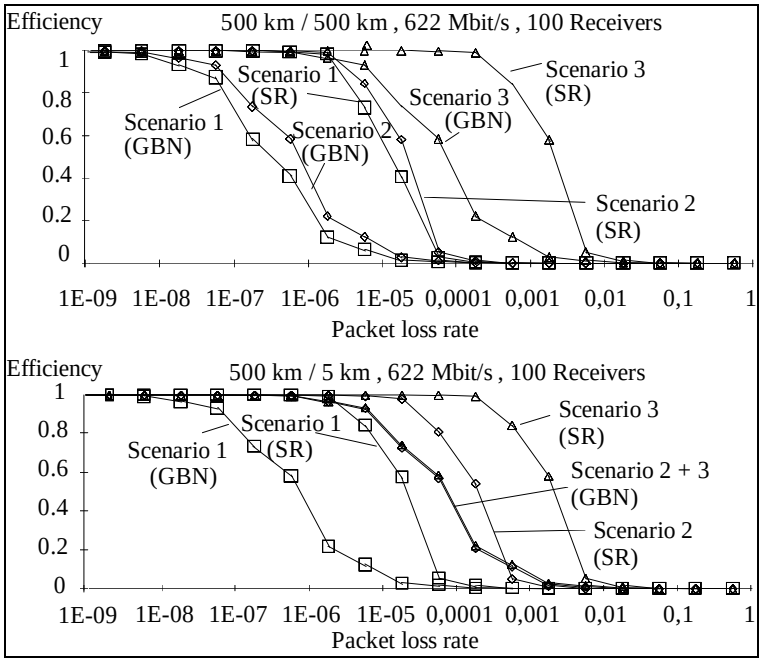


Figure 7. Efficiency Analysis for Go-back-N and Selective Repeat Retransmission Modes

4 VLSI for Retransmission Support

The processing overhead associated with handling selective retransmissions and the required data structures may be extremely high compared to other protocol functions. As an example, the achievable performance of an XTP [2] implementation on Digital Alpha Computers (150 MHz, 6.66 ns cycle time) may be regarded. In best case, the function to insert a new gap in a list needs 824 4-byte commands and takes, therefore, approximately 5.4 µs. The best case occurs if the new entry can be inserted at the beginning of the list. If the new entry has to be inserted after the first 10 entries, it needs 4054 commands or approximately 27.03 µs due to the search operations in the list. These calculations assume that the processor is not interrupted during execution of this function and all data is stored in the fast processor cache. XTP was implemented using C without special inline assembly code.

If data is sent at a rate of 1 Gbit/s, this results in more than 122,000 1024-byte packets per second. If the retransmission of each packet has to be controlled, this results in a new entry in the list in less than 10 µs.

Clearly, retransmission support forms a time critical task especially in a multicast environment. Therefore, we are implementing dedicated VLSI support for this task. The retransmission support presented in the following can handle negative selective, positive selective, and positive cumulative acknowledgements. It can be used for gaps managed by the receiver to support the acknowledgement function or for gaps managed by the transmitter to support the retransmission mechanism. The ALU has a set of commands to set, delete, insert, and read gaps for unicast connections and to manage multicast groups.

4.1 Logical Representation of Data

A dynamic linked list stores gaps of transmitted data in the following representation: *[seq_no_1, seq_no_2]* with *seq_no_1* and *seq_no_2* representing the beginning and ending of a gap. These gaps are connected via linked lists (cf. Figure 8). For every multicast group (MC) the pointers to the connections participating in that group are stored. A connection can be a member in different groups at the same time. For every connection the ALU stores a pointer to the appropriate list of gaps. Additionally, the ALU manages special lists for every multicast connection.

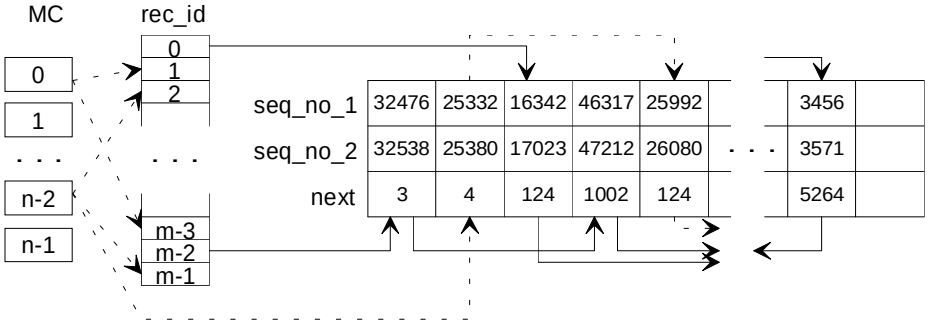
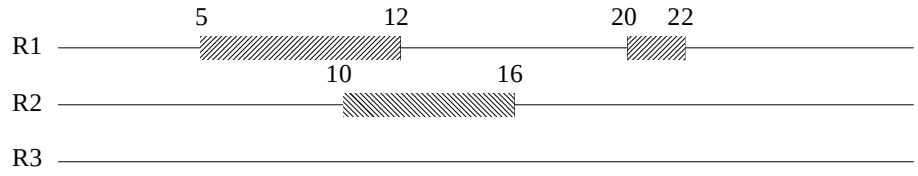


Figure 8. Logical Structure of Linked Lists for Retransmission Support

A simple example is given in Figure 9. First, the gap lists of the receivers R1, R2, and R3 are shown. The receiver R1, for example, has two gaps in transmitted data,

1) Lists of gaps for unicast connections



2) List of gaps to support k-reliability

Figure 9. Example of unicast and multicast lists of gaps

one from sequence numbers 5 to 12 and one from 20 to 22. The second type of lists stores the gaps needed to support full reliability or *k*-reliability of a multicast connection. In the example the gap (10,12) is stored, because less than 2 receivers have received this data.
Depending of the implemented protocol, retransmission of data can be performed by a multicast to the group or by individual retransmissions to the appropriate receivers.

4.2 Operations of the Retransmission ALU

The following table shows the operations supported by the specialised ALU.

operation	input parameters	output parameters	comment
`init_list`	`rec_id, seq_no`		initializes a new list for the connection *rec_id* with the initial sequence number *seq_no*, sets the error flag if *rec_id* is already in use
`close_list`	`rec_id`		closes the list for connection *rec_id*, sets the error flag if the list does not exist
`init_mcg`	`mc_con_id, rel`		initializes a new multicast group with the identification *mc_con_id* and the reliability *rel* (*rel* ≥ number of connections denotes full reliability)
`close_mcg`	`mc_con_id`		closes a mulitcast group and deletes all linked lists
`add_mcg`	`mc_con_id, rec_id`		adds a new connection *rec_id* to an existing multicast group *mc_con_id*

del_mcg	mc_con_id, rec_id		deletes an existing connection from a multicast group
set_rel	mc_con_id, k		sets the value *k* for the reliability of the multicast group *mc_con_id*
set_high_ack	rec_id, seq_no		sets the *high_ack* register to the value of *seq_no*; sequence numbers less than *high_ack* have been already acknowledged
shift_high_ack	rec_id, length		shifts the *high_ack* register to *high_ack* + *length*
set_high_seq	rec_id, seq_no		sets the *high_seq* register to the value of *seq_no*; *high_seq* represents the highest sequence number in use
shift_high_seq	rec_id, length		shifts the *high_seq* register to *high_seq* + *length*
set_gap_1	rec_id, seq_no, length		inserts new entry (*seq_no*, *seq_no* + *length*); overlapping entries are automatically joined or deleted, respectively
set_gap_2	rec_id, seq_no_1, seq_no_2		analogous to *set_gap_1*, but the new entry is of the form (*seq_no_1*, *seq_no_2*)
del_gap_1	rec_id, seq_no, length		deletes an existing entry, a part of an existing entry, or several existing entries, the deleted part is of the form (*seq_no*, *seq_no* + *length*); if necessary an entry is divided into two new entries
del_gap_2	rec_id, seq_no_1, seq_no_2		analogous to *delete_gap_1*, but the deleted part is of the form (*seq_no_1*, *seq_no_2*)
read_reg	rec_id, reg_id	cont	reads the contents *cont* of the register *reg_id* (e.g. high_ack, high_seq, number_of_gaps)
read_mc_reg	mc_con_id, reg_id	cont	reads the contents *cont* of the multicast register *reg_id* (e.g. number_of_gaps)

get_gap_1	rec_id, ptr	seq_no, length, next	reads the entry *ptr* points to; if *ptr* = 0, the first gap is read out, if *next* = 0 the entry represented by (*seq_no, length*) is the last one, otherwise *next* point always to the next entry of the list
get_gap_2	rec_id, ptr	seq_no_1, seq_no_2, next	analogous to *get_gap_1*, but the entry is represented by (*seq_no_1, seq_no_2*)
get_mc_gap_1	mc_con_id, ptr	seq_no, length, next	analogous to *get_gap_1*, but now the entries of the multicast group *mc_is* are read out
get_mc_gap_2	mc_con_id, ptr	seq_no_1, seq_no_2, next	analogous to *get_gap_2*, but now the entries of the multicast group *mc_is* are read out

Dimensioning of the component: *rec_id, k* ∈ [0, 255]; *mc_con_id* ∈ [0, 63]; *seq_no, seq_no_1, seq_no_2, length, cont* ∈ [0, 2^{32}-1]; *reg_id* ∈ [0, 15]; *ptr, next* ∈ [0, 2^{16}-1]

Table 3. Operations of the Retransmission ALU

Every operation sets the error flag if it failed due to memory overflow or violation of several conditions, such as *high_ack* ≤ *seq_no* ≤ *high_seq* and other range checking.

4.3 Implementation Architecture for Retransmission Support

Figure 10 shows an overview of the internal structure of the ALU. The retransmission ALU consists of 5 memory banks (A through E) that store sequence numbers representing gaps (memory A and B), pointers of linked lists (memory C), and state information, such as connection and multicast identification, register number of an anchor element, and other flags indicating the state of a connection (memory D and E). The I/O-bus connects the input/output-port (32 bit) of the retransmission ALU with the 5 register banks (Ai through Ei, 0≤i≤3). From these registers data can be transferred to the memory.

Two simple ALUs (*ALU A*, 32 bit and *ALU D*, 8 bit) perform operations like OR, XOR, AND, ADD, SUB, and NEG. Two specialized 32 bit modulo 2^{32} comparators (*comp A* and *comp B*) perform fast comparisons needed for list operations. The ALUs and the comparators can work concurrently if no data dependencies exist.

For the command `set_gap_2`, for example, first of all the command itself and the connection identification are read from the I/O-bus into the registers E and D, respectively. In the next two cycles the central control unit reads the sequence numbers

(`seq_no_1`, `seq_no_2`) into the registers A and B, respectively. After reading the complete command and several range checking operations the loop for searching the right position to insert the new entry in the list starts. Therefore, the first entry of the appropriate list is loaded into the registers A and B, respectively, and compared with the new entry. The loop terminates if the new entry fits, otherwise, the next entry is loaded and compared.

4.4 Microcode Examples of the Retransmission ALU

To provide a maximum of flexibility all functions of the *Retransmission ALU* are translated into a sequence of microcode operations. These operations are specially adapted to the implementation architecture shown in Figure 10. The *central control unit* controls the microprogram via a special micro sequencer.

Example microcode operations of the *Retransmission ALU* are listed in Table 4. In addition, there are the microcode operations of the *ALU D* and complex comparison operations of the two comparators *comp A* and *comp B*. The operations of the ALUs and the comparators are always executed in parallel in one clock cycle.

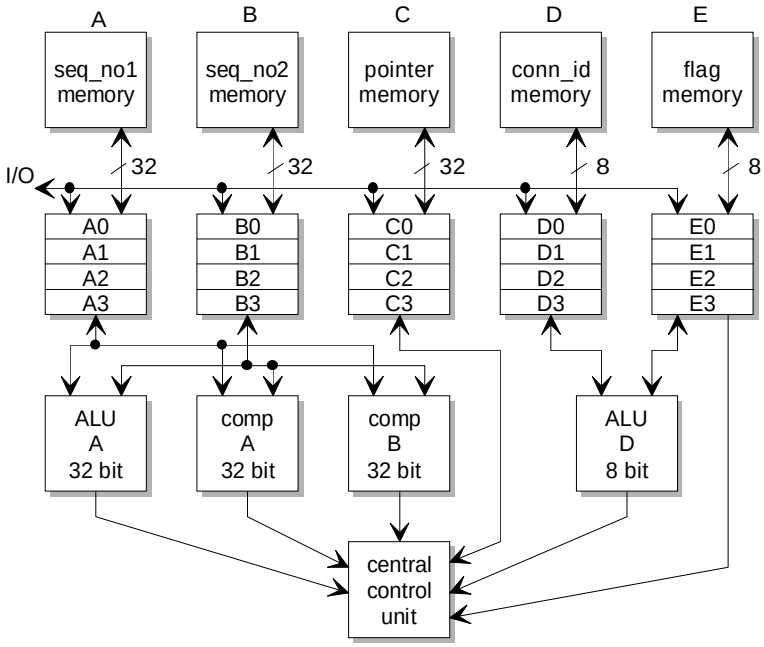


Figure 10. Structure of the Retransmission ALU

operations	comment
`RMOVE S, D`	move a complete row of entries from the registers or RAM into the registers or RAM. S, D ∈ {Ri, RAM; 0 ≤ i ≤ 3}, Rn = (An, Bn, Cn, Dn, En), S ≠ D
`ANOP`	no operation, ALU A

AMOVE I/O, D	move data from the I/O-bus into the register D; D ∈ {Ai, Bi; 0 ≤ i ≤ 3}
ACLR D	clear register D; D ∈ {Ai, Bi; 0 ≤ i ≤ 3}
AINC S, D	S + 1 -> D; S, D ∈ {Ai, Bi; 0 ≤ i ≤ 3}
ADEC S, D	S - 1 -> D; S, D ∈ {Ai, Bi; 0 ≤ i ≤ 3}
AMOVE S, D	S -> D; S, D ∈ {Ai, Bi; 0 ≤ i ≤ 3}
AADD S, D	S + D -> D; S, D ∈ {Ai, Bi; 0 ≤ i ≤ 3}
ASUB S, D	S - D -> D; S, D ∈ {Ai, Bi; 0 ≤ i ≤ 3}

Table 4. Microcode Examples of the Retransmission CPU

4.5 Implementation Detail: A 32 bit Modulo 2^{32} Comparator

Due to the complexity of the complete design, only a selected part is presented in detail. This section shows the design of a 32 bit modulo 2^{32} comparator (cf. *comp A* and *comp B* in Figure 10). Such a comparator is needed to compare three 32 bit values at the same time. This comparison is done while searching for the right position in the retransmission list to insert a new entry or to delete an existing entry. Due to the modulo 2^{32} arithmetic and the need for fast search operations in the dynamic lists common comparators with only two inputs cannot be used.

Four functions are calculated at the same time:

le_le (a, b, c) = true $\Leftrightarrow ((a \le c) \wedge (a \le b \wedge b \le c)) \vee ((a > c) \wedge (a \le b \vee b \le c))$

le_lt (a, b, c) = true $\Leftrightarrow ((a < c) \wedge (a \le b \wedge b < c)) \vee ((a > c) \wedge (a \le b \vee b < c))$

lt_le (a, b, c) = true $\Leftrightarrow ((a < c) \wedge (a < b \wedge b \le c)) \vee ((a > c) \wedge (a < b \vee b \le c))$

lt_lt (a, b, c) = true $\Leftrightarrow ((a < c) \wedge (a < b \wedge b < c)) \vee ((a > c) \wedge (a < b \vee b < c))$

a, b, c ∈ $\mathbb{N}$ mod 2^{32}; le: less or equal; lt: less than

The complete architecture and its components are described with the standardized hardware description language VHDL[28]. This allows for simulation and synthesis based on the same language.

```
library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_arith.all;

entity MOD_KOMPA is
  port ( A, B, C : IN std_logic_vector (31 downto 0);
         le_le, le_lt, lt_le, lt_lt: OUT boolean);
end MOD_KOMPA;

architecture BEHAVIORAL of MOD_KOMPA is
begin
  process (A,B,C)
  variable h1, h2, h3, h4, h5, h6 : boolean;
  begin
   h1 := A < B; h2 := A <= B;
   h3 := A < C; h4 := A >  C;
   h5 := B < C; h6 := B <= C;
```

```
    le_le <= (NOT(h4) AND (h2 AND h6)) OR (h4 AND (h6 OR h2));
    le_lt <= (h3      AND (h2 AND h5)) OR (h4 AND (h5 OR h2));
    lt_le <= (h3      AND (h1 AND h6)) OR (h4 AND (h6 OR h1));
    lt_lt <= (h3      AND (h1 AND h5)) OR (h4 AND (h5 OR h1));
  end process;
end BEHAVIORAL;
```

The above listed VHDL description was synthesized into a gate level description using a high level synthesis tool. The area of the design is 1084 gates and the estimation of the critical path 9.6 ns. The implementation of only the *le_le* function on an Alpha processor needs 20 4-byte commands which results with 6.6 ns cycle time (150 MHz) in a duration of more than 132 ns.

5 Summary and Future Work

Within this paper, a framework for the provision of high performance real-time multicast services has been presented which has the potential to fulfill the requirements of upcoming distributed multimedia applications. It is based on VLSI components dedicated to specific processing tasks that are to be integrated in end systems, special network elements called group communication servers, and the deployment of resource reservation in intermediate and end systems. Simulation results on the suitability of selected scheduling algorithms for services with guaranteed delay are presented. A performance evaluation is given which shows the potential benefits of selective retransmissions in multipoint connections, and potential improvement of efficiency if GCSs are integrated into the network. Implementation details of multicast retransmission support have been discussed. It was shown how certain system and support functions may be implemented efficiently by the use of dedicated hardware.

Not only high performance, efficient use of network resources and resource reservation, but specifically service integration will be a major requirement for forthcoming communication subsystems. System components that may be selected and parametrized based on the requested application service will be important for providing a high degree of flexibility. The presented framework thus may be viewed as a hardware implementation of the function-based communication subsystem F-CSS presented in [29]. Currently, the implementation of additional components for FEC and memory management are under development. A more detailed evaluation of the achievable performance is also subject of ongoing work, including investigation of the influence of processing times and of limited buffers.

Acknowledgement. The authors would like to thank Martina Zitterbart, Torsten Braun, and Burkhard Stiller for valuable discussions. The support by the Graduiertenkolleg „Controllability of Complex Systems“ (DFG Vo287/5-2) is also gratefully acknowledged.

6 References

[1] Ito, M.; Takeuchi, L.; Neufeld, G.; *Evaluation of a Multiprocessing Approach for OSI Protocol Processing;* Proceedings of the First International Conference on Computer Communications and Networks, San Diego, CA, USA, June 8-10, 1992

[2] Strayer, W.T.; Dempsey, B.J.; Weaver, A.C.; *XTP: The Xpress Transfer Protocol;* Addison-Wesley Publishing Company, 1992

[3] Feldmeier, D.C.; *An Overview of the TP++ Transport Protocol;* in: Tantawy A.N. (ed.): High Performance Communication, Kluwer Academic Publishers, 1994

[4] Sterbenz, J.P.G.; Parulkar, G.M.; *AXON Host-Network Interface Architecture for Gigabit Communications;* in: Johnson, M. J. (ed.): Protocols for High-Speed Networks, II, North-Holland, 1991, pp. 211-236

[5] Braun, T.; *A Parallel Transport Subsystem for Cell-Based High-Speed Networks;* Ph.D. Thesis (in German), University of Karlsruhe, Germany, VDI-Verlag, Düsseldorf, 1993

[6] Braun, T.; Zitterbart, M.; *Parallel Transport System Design;* in: Danthine, A.; Spaniol, O. (eds.): High Performance Networking, IV, IFIP, North-Holland, 1993, pp. 397-412

[7] Krishnakumar, A.S.; Kneuer, J.G.; Shaw, A.J.; *HIPOD: An Architecture for High-Speed Protocol Implementations*; in: Danthine, A.; Spaniol, O. (eds.): High Performance Networking, IV, IFIP, North-Holland, 1993, pp. 383-396

[8] Balraj, T.; Yemini, Y.; *Putting the Transport Layer on VLSI - the PROMPT Protocol Chip;* in: Pehrson, B.; Gunningberg, P.; Pink, S. (eds.): Protocols for High-Speed Networks, III, 1992, North-Holland, pp. 19-34

[9] Heinrichs, B.; Jakobs, K.; Carone, A.; *High performance transfer services to support multimedia group communications*; Computer Communications, Volume 16, Number 9, September 1993

[10] Waters, A. G.; *Multicast Provision for High Speed Networks*; 4th IFIP Conference on High Performance Networking HPN'92, Liège, Belgium, December 1992

[11] Bubenik, R.; Gaddis, M.; DeHart, J.; *Communicating with virtual paths and virtual channels*; Proceedings of the Eleventh Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM'92, pp. 1035 - 1042, Florence, Italy, May 1992

[12] Ferrari D., Banerjea A., Zhang H.; *Network Support for Multimedia - A Discussion of the Tenet Approach;* Technical Report TR-92-072, International Computer Science Insitute, Berkeley, California, November 1992

[13] Herrtwich R.G.; *An introduction to real-time scheduling*; Tech. Rept. TR-90-035, International Computer Science Institute, Berkeley, July 1990

[14] Zhang L.; *Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks*; Proceedings of ACM SIGCOMM'90, Philadelphia, September 1990

[15] Parekh A.K.; *A Generalized Processor Sharing Aproach to Flow Control in Integrated Services Networks*; PhD Thesis, Department of Electrical Engineering and Computer Science, MIT, February 1992

[16] Kalmanek, C.R.; Kanakia, H.; Keshav, S.; *Rate controlled servers for very high-speed networks;* in proceedings of GLOBCOM ‘90, San Diego, December 1990

[17] Zhang H., Keshav S.; *Comparison of Rate-Based Service Disciplines;* Proceedings of SIGCOMM '91, Communications Architecture and Protocols, Zurich, Switzerland, September 1991

[18] Kurose, J. F.; *Open Issues and Challenges in Proving Quality of Service Guarantees in High-Speed Networks;* ACM Computer Communication Review, Vol. 23, No. 1, January 1993, pp. 6-15

[19] Partridge, C.; *Gigabit Networking;* Addison-Wesley, 1994

[20] Topolcic, C. (Editor).; *Experimental Internet Stream Protocol, Version 2 (ST-II);* Internet Request for Comments RFC 1190, October 1990

[21] Zhang L., Deering S., Estrin D., Shenker S., Zappala D.; *RSVP: A New ReSource ReserVation Protocol*; IEEE Network Magazine, Vol. 9, No. 5, September 1993

[22] McAuley, A.; *Reliable Broadband Communication Using a Burst Erasure Correcting Code*; Presented at ACM SIGCOMM '90, Philadelphia, PA, U.S.A., September 1990

[23] Biersack, E. W.; *Performance Evaluation of Forward Error Correction in an ATM Environment*; IEEE Journal on Selected Areas in Communication, Volume 11, Number 4, pp. 631-640, May 1993

[24] Kleinewillinghöfer-Kopp, R.; Lehnert, R.; *ATM Reference Traffic Sources and Traffic Mixes;* RACE 1022 BLNT Workshop, Munich 1990

[25] Gopal, I.; Jaffe, J.; *Point-to-Multipoint Communication Over Broadcast Links*; IEEE Trans. Commun., Vol. Com-32, No. 9, pp. 1034-1044, September 1984

[26] Sabnani, K.; *Multidestination Protocols for Satellite Broadcast Channels*; Ph. D. Thesis, Columbia University, NY, U.S.A., 1982

[27] Wang, J.; Silvester, J.; *Performance optimisation of the go-back-N ARQ protocols over broadcast channels*; Computer Communications Vol. 14, No. 7, pp. 393-402, September 1991

[28] IEEE; *Standard VHDL Language Reference Manual;* IEEE Std 1076-1987

[29] Zitterbart, M.; Stiller, B.; Tantawy, A.; *A Model for Flexible High-Performance Communication Subsystems;* IEEE-JSAC, May 1993, pp. 507-518