GROUP COMMUNICATION SERVERS WITH VLSI SUPPORT FOR HETEROGENEOUS HIGH-PERFORMANCE NETWORKS

Georg Carle, Jochen Schiller

High Performance Networking Group
Institute of Telematics, University of Karlsruhe, Germany
[g.carle, j.schiller]@ieee.org

Abstract

This paper presents a framework for the provision of high performance multicast services which has the potential to fulfill the requirements of upcoming distributed applications. The conventional approach of error control performed in end systems is appropriate for small groups, low error probability, and low path capacities. Where these conditions are not met, the deployment of Group Communication Servers offers superior scaling properties. In heterogeneous internetworks, reliable multicasting involving Group Communication Servers can be based on XTP. In a homogeneous ATM environment, a reduced overhead in comparison with XTP can be achieved by the deployment of the Reliable Multicast Service Specific Transfer Protocol RMC-SSTP on top of AAL5 CPCS. This paper also presents a highly modular implementation architecture for a communication co-processor that is especially targeted towards the support of protocols like XTP or RMC-SSTP. Key features are high flexibility and performance due to parametrizable and programmable VLSI components. Promising synthesis results show that a much higher performance can be reached with this approach compared to conventional systems.

1 Motivation

Emerging applications mostly require both, high performance as well as support of a wide variety of communication services. For example, audio, video, and data transmission may require highly different services. An additional challenge arises by the growing demand for multipoint communication services. ATM networks are capable of satisfying the basic application requirements by providing multipoint bearer services [1] with data rates exceeding a gigabit per second. However, current communication subsystems (including higher layer protocols) that provide reliable services are not able to deliver the available network performance to the applications [2], [3]. In particular in multipoint communication scenarios, severe degradations of service quality may be observed. Additional problems need to be faced in scenarios where QoS requirements and processing capabilities of individual receivers differ.

To provide high performance integrated service communication subsystems, suited implementation platforms are increasingly required [4]. They may comprise of dedicated VLSI components for time-critical processing tasks, such as retransmission support or memory management. In the paper, a framework for the provision of multipoint multimedia services in ATM networks and heterogeneous internetworks is presented. It provides support for processing of layer 3 and 4 protocol functions in end and intermediate systems for enhancing the network bearer service. The concept of Group Communication Servers (c.f. figures 2 and 8) allows the efficient provision of reliable multipoint services for large, widespread groups. With multicast error control capabilities, they allow increased throughput and reduced delay. They provide processing support for multicast transmitters and reduce the acknowledgment implosion problem. They also support groups consisting of end systems with direct ATM access, as well as end systems connected over heterogeneous internetworks.

A modular VLSI implementation architecture designed around specialized components allows for service flexibility [5]. The components can be reprogrammed using microprograms and may be selected individually depending on the service required by the application. The architecture is specifically designed for implementing complex connection oriented protocols with advanced protocol mechanisms [6], [7]. Specific support for processing intensive functions is given. For example, selective retransmission is provided by a dedicated VLSI component. The architecture is highly independent of the specific protocol to be implemented and, thus, forms a general implementation platform for high-performance communication protocols. The Xpress Transfer Protocol (XTP) [8] is a good candidate for the envisaged applications. For full exploitation of the benefits of ATM networks, a new protocol called the Reliable Multicast Service Specific Transfer Protocol (RMC-SSTP) is proposed. The dominant factor which causes high speed networks to discard packets is buffer overflow due to congestion. The probability for packet loss may vary over a wide range, depending on the applied strategy for congestion control. For multicast connections, the problem of packet losses is even more crucial than for unicast connections. It is more difficult to ensure a low packet loss rate. Losses occur more frequently and every loss causes costly processing for a multicast transmitter.

For applications that cannot tolerate packet losses of the network, error control mechanisms are required. Error control is a difficult task in networks that offer high bandwidth over long distances, where a large amount of data may be in transit [9]. Two mechanisms are available for error correction: Automatic Repeat ReQuest (ARQ) and Forward Error Correction (FEC).

In contrast to retransmission schemes, FEC promises a number of advantages [10]. 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 [11]. The latter limits the usefulness of FEC in many cases. For an accurate assessment of FEC it must be considered that its best performance is achieved for random errors, while packet losses frequently occur in bursts [12]. 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.

Protocols based on ARQ mechanisms are widely used in current data link and transport protocols [13]. However, for high-performance multicast communication, there are still many open questions concerning acknowledgment and retransmission strategy, achievable performance and implementation. Retransmissions may be performed as go-back-N [14], [15] (e.g., in TCP), or as selective repeat [16] (e. g., offered in XTP [8] and PATROCLOS [17]). While go-back-N schemes are appropriate for point-to-point communication with low error rates and moderate path capacities, selective repeat schemes are essential for high-performance multicast communication in wide-area networks that may observe congestion [18]. 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. In large groups, the well-known implosion problem can be observed, where processing of acknowledgements leads to a severe performance bottle-neck. If reliable communication to every multicast receiver is required, a substantial part of the transmitter complexity is growing proportionally with the group size. In addition, individual receivers may limit the service quality of the whole group. To reduce the severeness of these problems, a scheme that provides reliable delivery of messages to K out of N receivers may be applied ([19]; K-reliable service).

This paper is organized as follows. The following section gives an overview of multipoint communication in high-speed networks and presents the conceptual framework for integrating VLSI components for multicast support into end systems and Group Communication Servers. Thereafter, a section discusses the functionality of a dedicated coprocessor for managing retransmission, and presents performance results as well as implementation complexity of the discussed component. Finally, the last section summarizes the paper and points out some future directions.

2 High-Performance Multicast Servers

In order to meet the QoS requirements of many real-time applications, it is a common approach to meet the application reliability requirements without performing error control mechanisms. In situations where it is difficult to provide a network bearer service which meets the reliability requirements of the application directly, the following strategy may be applied: providing high protocol processing capability with a low latency to ensure that the real-time requirements are met even after one or two retransmissions of a message. This strategy potentially offers a way for a better utilization of network resources.

In [20], it was shown by simulation that a real-time retransmission scheme is feasible within the end-to-end delay constraints of packet voice transmissions for overall one-way delays with an average of 12 ms and a maximum of 36 ms. While the authors of [20] used relatively high network access delays and protocol processing delays in transmitter and receiver for modeling the one-way delay, the propagation delay of 5 ms for a fiber-optic transmission over a distance of 1000 km shows that retransmission schemes for real-time applications may also be applied for relatively large distances.

A conceptual framework was described [18] for the use of error control mechanisms best suited for a specific multipoint communication scenario at locations that allow highest performance. The integration of specialized 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 into the network that provide support for reliable group communication. A significant advantage can be achieved if a hierarchical approach is chosen for multicast error control.

The framework on which this paper is based applies to protocols that use packets with byte sequence numbers and byte length for identification of the payload, that use gaps specified by byte sequence numbers for positive and negative acknowledgments, and that allow selective retransmissions to multiple receivers. This error control functionality may be part of a transport protocol, such as XTP Revision 4.0 [21] in combination with a connectionless network layer. This functionality may also be part of a transfer protocols combining layer 3 and layer 4 functions, such as XTP [8] and PATROCLOS [22] with multicast extensions. A transfer protocol may be used over a conventional LLC service, or over an adaptation layer service as for example offered by AAL5.

Figure 1 presents a protocol architecture with multicast mechanisms in the Service Specific Convergence Sublayer of the Adaptation layer of end systems, and in dedicated servers. The term Group Communication Server describes an intermediate system with multicast error control capability which may be attached to conventional subnetworks or to an ATM network. It may be combined with routing functionality of, e.g., an XTP router (c.f. Figure 2).

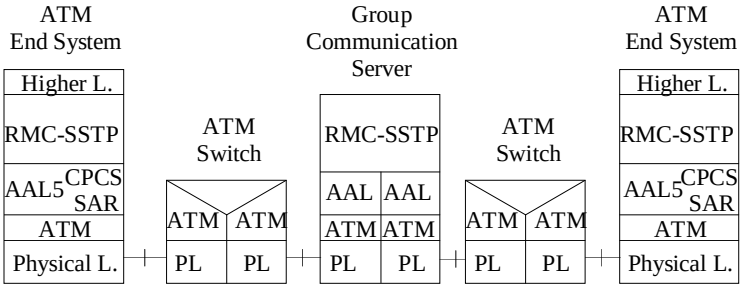

Figure 1: RMC-SSTP in a homogeneous ATM network

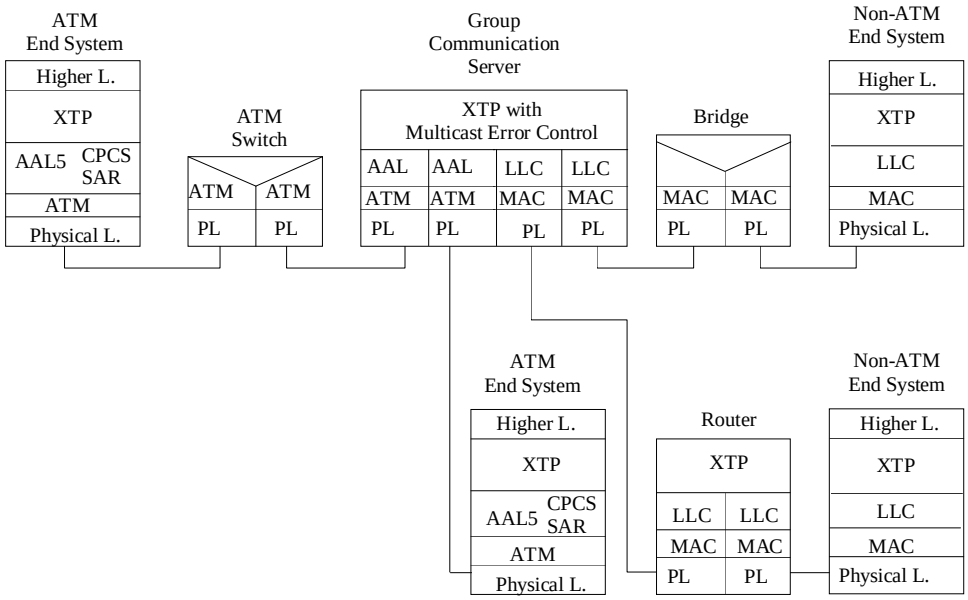

Figure 2: XTP-based Group Communication Server in a heterogeneous internetwork

The Group Communication Server (GCS) presented in this paper may integrate a number 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. In order to ensure low delay, the server does not guarantee an in-sequence forwarding of packets. Instead, it will forward every packet to the receivers as soon as possible. In combination with a network node with copy function, it is not required that the server processes a packet before forwarding it to the receivers. Instead, copies may be forwarded in parallel to the server and the receivers, as shown in Figure 3 (bypass mode). This guarantees minimal delay while allowing that the server detects losses prior to the receiver and initiates a retransmission by the sender.

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 small number of GCSs, where each of them provides reliable delivery to a subset of the receivers. Integrating hardware support for reliable high performance multipoint communication into a server allows better use of dedicated resources such as coprocessors. For end systems, it is not required to have VLSI components for multicast error control. For an end system, 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.

For the third task, a GCS may use the potential of diversifying outgoing data streams, allowing support of different qualities of service for individual servers or subgroups. The GCS may also apply filtering functions for specific data streams. It can support different subnetworks, such as FDDI and ATM, where a different set of protocol parameters may be appropriate, and may also support different error control schemes, such as Go-back-N and selective repeat.

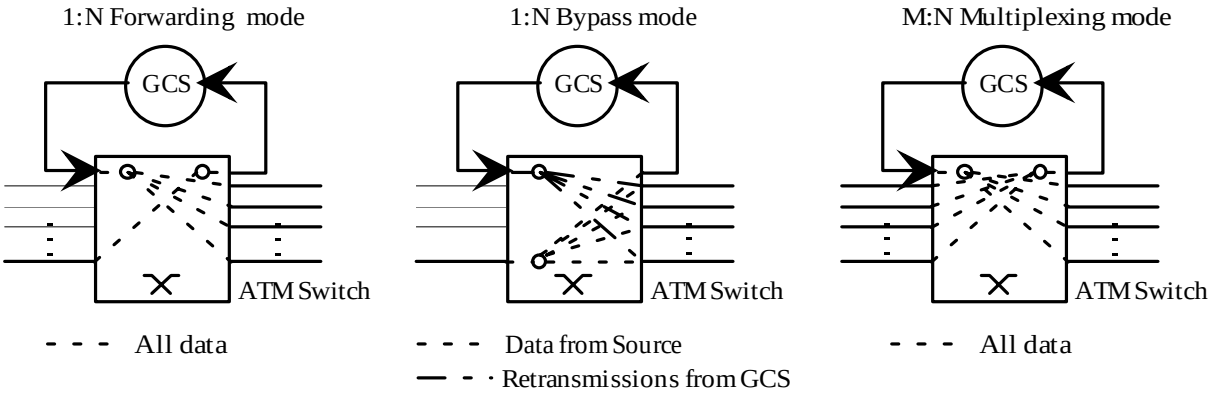

Figure 3: Operation modes of the Group Communication Server

2.1 Server modes

The Group Communication Server may operate in three different modes, as shown in Figure 3. In the forwarding mode, every frame is processed first by the GCS before being forwarded to the receivers. In case of simple 1:N multicasting, increased performance may be achieved in the bypass mode. In this mode, an ATM switch that supports multicasting will forward data directly to the server and the receivers, reducing the processing load of the server and the overall latency.

In both modes, the GCS detects errors earlier than the receivers, and can notify the source quicker that an error occurred. Both modes also support processing of acknowledgments. For this purpose, every receiver may maintain an individual virtual channel to the GCS. The GCS will either perform the required retransmissions, or will forward retransmission requests to the source. If a window-based flow control scheme is enforced that includes the GCS, the GCS may guarantee to perform the retransmissions. However, buffer limitations in the GCS may limit performance in this case.

The third case is more complex, but allows the provision of a multipeer service with multiplexing of messages from different transmitters over a single virtual connection. In all three modes, a hierarchy of servers allows for good scaling properties for large groups and high path capacities.

2.2 Reliable Multicast Service Specific Transfer Protocol RMC-SSTP

The transfer protocol XTP is a very general protocol that may be used in heterogeneous internetworks. With a packet header of 40 bytes and a trailer of 4 bytes [21], even very short messages require more than a single-cell frame if XTP is used in combination with AAL5. Additional problems arise because connection management of XTP is substantially different to connection management in ATM networks. Therefore, a protocol with new data formats was designed for the Service Specific Convergence Sublayer of AAL5 that allows a more efficient use of network resources, and that allows to make use of the functionality of signaling protocols in ATM. The protocol is designed to offer the following services: assured delivery of messages to every member of the group (full reliability), assured delivery to at least K receivers of a group (k-reliability), and a real-time service that offers retransmissions to the receivers subject to deadlines. Figure 1 shows the deployment of RMC-SSTP in a homogeneous ATM network.

Figure 4 shows the format of a data frame of RMC-SSTP. In order to offer stream-oriented services, the payload of a PDU is identified by information equivalent to the information of an XTP PDU. The first byte of the payload is identified by a byte sequence number (BSN). The SDU is identified by a SDU sequence number (SDU-SN). Every frame carries the total length of a PDU (SDU-Len.). This is equivalent to the identification of the payload of an XTP packet, simplifying interworking with XTP and adaptation of existing XTP implementations to the new protocol.

The remaining protocol fields of a data frame are different to the protocol fields of an XTP packet. The first byte of a frame (Dis) is used to identify the frame type (by the field FrType), to request immediate acknowledgments from the receivers (by setting the flag I-Ack), and to indicate that a frame is the last frame in a burst (by setting the LastF flag). The LastF flag avoids unnecessary idle-frames. The field Tx-ID allows to identify a specific sender. This is of particular interest when frames of several transmitters are multiplexed and distributed over the same ATM multicast VC. In order to allow multiplexing of AAL5 frames, a specific multiplexing scheme must be applied that avoids mixing of cells from different frames. The field LWE contains the lower window edge of the transmitter buffer. This allows to indicate the lowest byte sequence number which might be retransmitted. The feature is useful for real-time and for k-reliable services. For a fully reliable service, and for cases where the status information of individual receivers is stored by the transmitter, receivers send acknowledgments with detailed status information. The format of an acknowledgment is shown in Figure 5. The lower window edge (LWE) is used for cumulative positive acknowledgments. The field Hi-Seq indicates the highest byte sequence number that was received. The UWE identifies the maximum byte sequence number a receiver is prepared to receive and may be used for flow control. The number of gaps contained in an acknowledgment frame is indicated by the field #gaps. Gaps are identified by the byte sequence numbers of their lower (BSN-L) and upper edge (BSN-H). The field Rx-ID allows to identify acknowledgments of individual receivers that are multiplexed onto the same VC. The field FrType allows to distinguish acknowledgment frames from data frames.

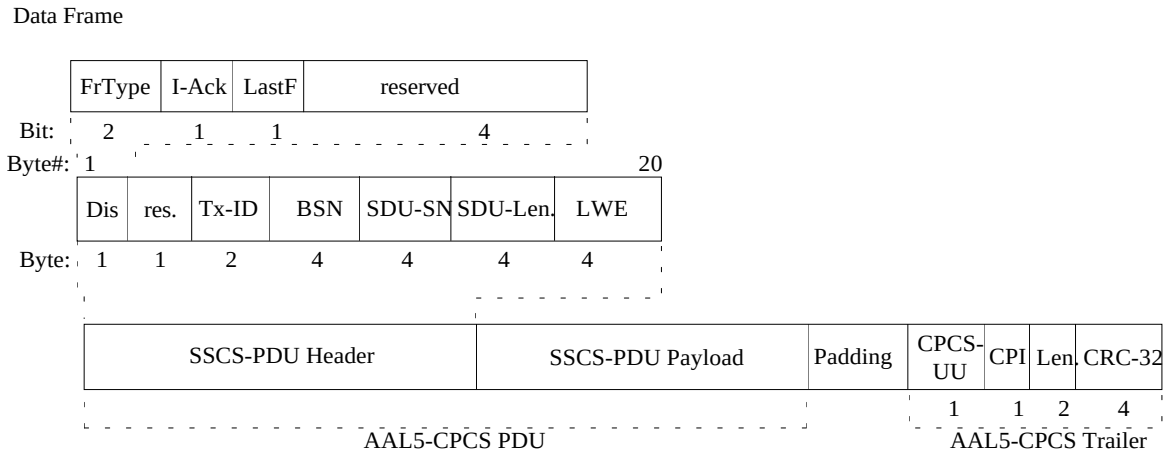

Figure 4: Data format of SSCS-PDU

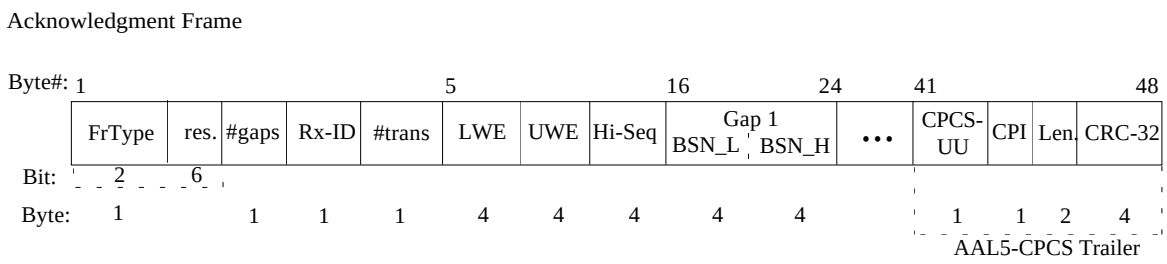

Figure 5: Format of acknowledgments

3 Implementation

Efficient protocols with error control mechanisms for intermediate and end systems are important, but not sufficient for the provision of high-performance multipoint multimedia services. Equally important are implementation architectures that allow for high-performance protocol processing. Therefore, the proposed framework also covers the integration of VLSI components dedicated to specific processing tasks into end systems and Group Communication Servers.

The architecture described in this section may be implemented as part of an intelligent network adapter or as coprocessor(s) similar to floating point units within a workstation. Parallelism is supported at various levels, e.g., fine-grained parallelism among protocol functions as well as coarse-grained parallelism among connections.

The Connection Processor (c.f. Figure 6) forms the basic building block. It consists of multiple components: a simple crossbar switch, processing units (e.g., simple Finite State Machines (FSMs), and microcontrollers), ALUs, and timer components. All components have identical interfaces to the switch. Such an interface consists of a data bus and two handshake lines. The crossbar switch supports multicast and two priority levels. The CP shown in Figure 6 is configured for a network adapter and, therefore, the internal switch has 4 connections to its environment (memory management units, network, host). To support protocol processing, global ALU and timer components may be integrated inside a CP.

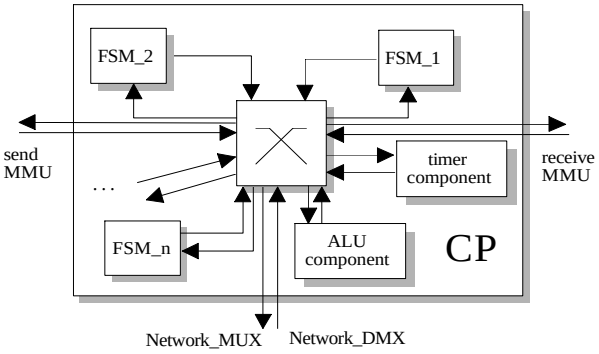

Figure 6: Connection processor with interconnected finite state machines

The protocol itself may be implemented using extended finite state machines (FSM_i, c.f. Figure 7). Therefore, the protocol description should consist of a set of FSMs communicating via signaling. Existing protocols can be subdivided into functions and, thus, be implemented on the presented architecture.

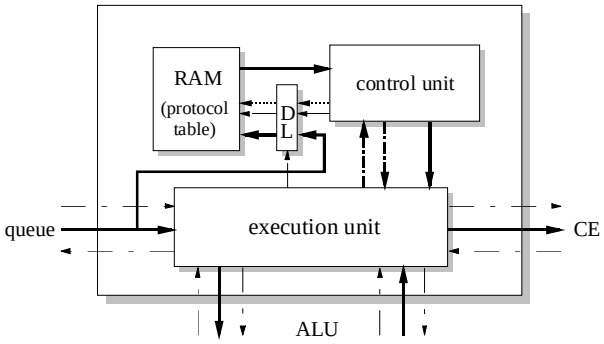

Figure 7: Internal architecture of a finite state machine

An FSM component consists of several modules: a queue to decouple the components, the finite state machine with its state transition table, and a local ALU to support time consuming protocol functions. Decoupling is needed in order to avoid deadlocks and to speed up overall execution time. The state transition table and, therefore, the protocol is implemented as a microprogram. Thus, although the structure of the FSM component is fixed, different protocol automata can be implemented via reprogramming using the same basic VLSI architecture. Without changing the basic implementation architecture, standard micro-controllers can also be used instead of FSMs for protocol processing.

Figure 8 shows a proposed implementation architecture of the GCS. Main focus of the design was to achieve a high degree of pipelining. The functionality of the GCS is distributed to a number of modules with FSMs and local memory. The send manger is responsible for scheduling. within a multipoint connection, scheduling between ordinary transmissions, retransmissions and acknowledgements is performed. Scheduling between different multicast connections allows to provide support for service quality guarantees. Hardware components are provided for cyclic redundancy check (CRC), buffer management, list and timer management. The ARQ manager generates acknowledgements and also provides multicast flow control information to the send manager.

If an application needs reliable data transfer, receiving of all data has to be controlled and enforced. Therefore, a list representing data that has to be acknowledged needs to be implemented. Every time data is sent or an acknowledgment is received, the list needs to be updated. Assuming a bit rate of 600 Mbps and 600 Byte per packet, every 8µs such an update has to be performed which is almost impossible for pure software solutions.

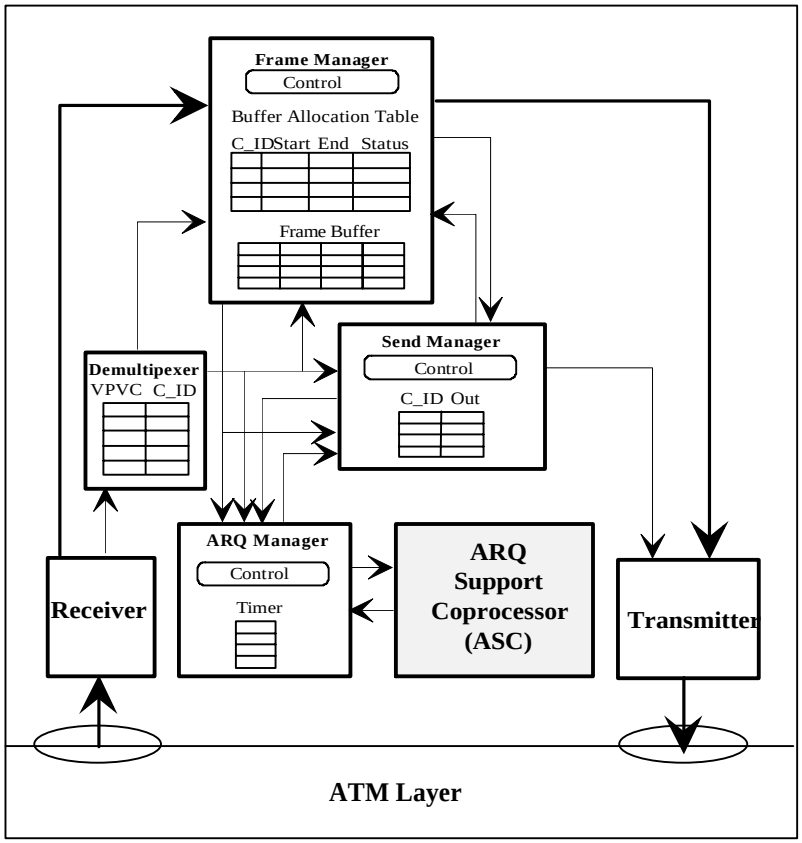

Figure 8: Architecture of the Group Communication Server

As acknowledgement processing for a large number of receivers is a potential bottleneck, dedicated hardware support is provided for the ARQ manager unit. The potentially time-consuming list functions for acknowledgement processing are performed by a specialized high-performance component, the so-called ARQ Support Coprocessor (ASC). The ASC is designed to support protocols with selective retransmissions based on byte sequence numbers. The component provides for a certain set of powerful list manipulation functions (c.f. table 1 for examples).

The ALU supporting this task consists of memory to store the required information, registers, ALUs for standard op-

erations, control units, and specialized comparators. The comparator units can, e.g., perform the operation $a \le b \le c$, $b \le c \le d$ in a single clock cycle ($a,b,c,d \in$ int). This operation is applied to search in a list. The developed ASC is fully microprogrammable and may also be used as a list coprocessor for conventional implementation architectures (e.g., network adapters, workstations). Example microcode operations of the ASC are listed in table 2. The operations of the ALUs, the central control unit and the comparators are always executed in parallel in one clock cycle.

The complete architecture with all its components is described using the hardware description language VHDL [23] to provide simulation and synthesis based on the same language. Preliminary synthesis results using a 0.7µm CMOS standard cell library show promising results. E.g., the above mentioned comparisons $a \le b \le c$, $b \le c \le d$ need 45ns on this coprocessor, while the simple comparison $a \le b \le c$ requires 72.6ns on an Alpha processor assuming one instruction per cycle and no context switching (11 instructions, 6.6ns cycle time, incl. load/store). The control logic of the processor needs 28800 gates or 10945 standard cells, respectively. Additional 64k RAM is needed to support the handling of 600 gaps in the transmitted data stream. The size of the control logic is 49.02mm^2, 130 pins are needed in total.

4 Summary

Within this paper, the concept of Group Communication Servers with VLSI support for heterogeneous high-performance networks was introduced. This concept has the potential to fulfill the requirements of upcoming distributed applications. It was shown that these servers offer superior scaling properties and offer reliable multicasting for heterogeneous and homogeneous internetworks. To provide the necessary processing power for the aggregated bandwidth, a special modular VLSI implementation architecture was introduced. Key features of this architecture are high performance and flexibility due to programmable components.

Currently, the implementation of additional components for FEC and memory management are under development. In addition to the standard-cell CMOS design, several components will be also implemented on FPGAs inserted into workstations. 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.

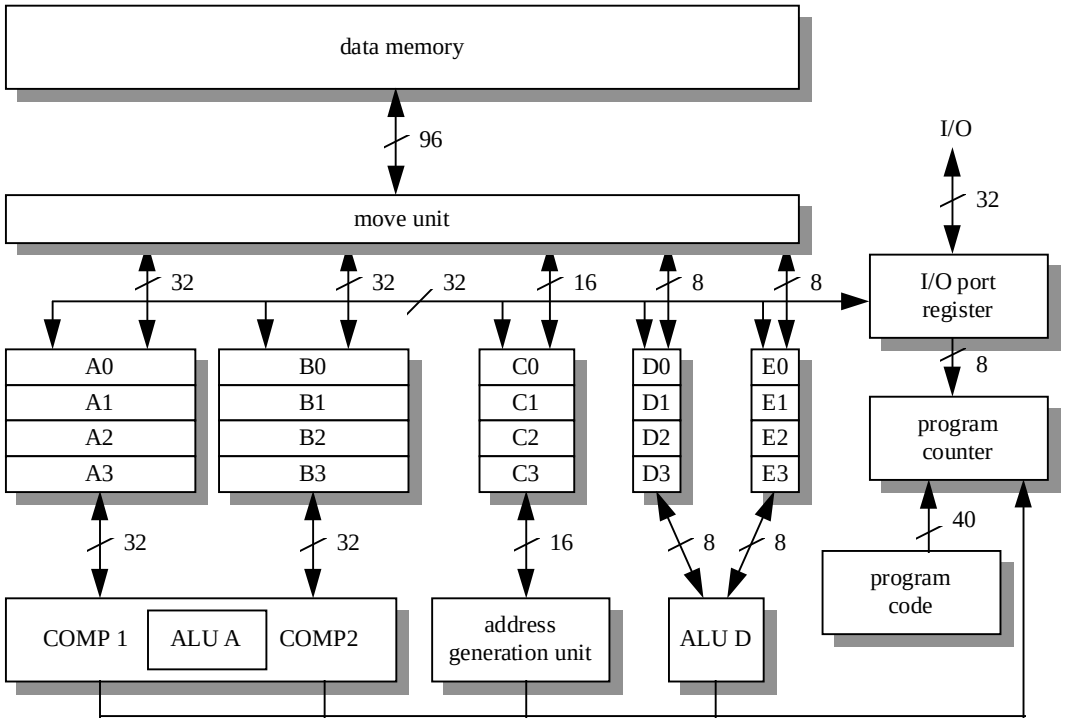

Figure 9: Structure of the ARQ Support Coprocessor (ASC)

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
`add_mcg`	`mc_con_id`, `rec_id`		adds a new connection *rec_id* to an existing multicast group *mc_con_id*
`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
set_gap	rec_id, seq_no, length		inserts new entry (*seq_no, seq_no + length*); overlapping entries are automatically joined or deleted, respectively
del_gap	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 automatically divided into two new entries
get_gap	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

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 1: Example operations 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
MOVE I/O, D	move data from the I/O-bus into the register D; D ∈ {Ai, Bi; 0 ≤ i ≤ 3}
TBBC Ri.n, ra	test bit n of register Ri and branch to relative address ra if clear; R ∈ {Di, Ei; 0 ≤ i ≤ 3}, 0 ≤ n ≤ 7

CBMOD Ri, Rj, Rk, Rl, ra1, ra2, ra3	compare Ri ≤ Rj ≤ Rk and Rj ≤ Rk ≤ Rl modulo 2^{32} and branch to:
	result = 00 then PC := PC + 1; result = 01 then PC := PC + ra1;
	result = 10 then PC := PC + ra2; result = 11 then PC := PC + ra3;
	PC: program counter; Ri, Rj, Rk, Rl ∈ {Ai, Bi; 0 ≤ i ≤ 3}

AADD S, D	S + D -> D; S, D ∈ {Ai, Bi; 0 ≤ i ≤ 3}

Table 2: Microcode examples of the retransmission ALU

References

[1] 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

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

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

[4] 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

[5] 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

[6] 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

[7] Braun, T.; Schiller, J.; Zitterbart, M.; *A Highly Modular VLSI Implementation Architecture for Parallel Transport Protocols;* IFIP 4th International Workshop on Protocols for High-Speed Networks, Vancouver, Canada, August 10-12, 1994

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

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

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

[11] 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

[12] Ohta, H., Kitami, T.; *A Cell Loss Recovery Method Using FEC in ATM Networks*; IEEE Journal on Selected Areas in Communications, Vol. 9, No. 9, December 1991, pp.1471-1483

[13] 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

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

[15] 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

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

[17] 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

[18] Carle, G.; *Adaptation Layer and Group Communication Server for Reliable Multipoint Services in ATM Networks;* in: Steinmetz, R. (ed.): Multimedia: Advanced Teleservices and High-Speed Communication Architectures, Springer, 1994, pp. 124-138

[19] Santoso, H.; Fdida, S.; *Transport Layer Multicast: An Enhancement for XTP Bucket Error Control*, in: Danthine, A.; Spaniol, O. (eds.): High Performance Networking, IV, IFIP, North-Holland, 1993

[20] Dempsey, B.; Liebherr, J.; Weaver, A.; *A New Error Control Scheme for Packetized Voice over High-Speed Local Area Networks*, Proceedings of 18th Conference on Local Computer Networks, September 19-22, 1993, Minneapolis, Minnesota, U.S.A, pp. 91-100

[21] The XTP Forum; *XTP Protocol Definition Proposed Revision 4.0*, 1994

[22] Braun, T.; *A Parallel Transport Subsystem for Cell-Based High-Speed Networks;* Ph.D. Thesis (in Ger-

man), University of Karlsruhe, Germany, VDI-Verlag, Düsseldorf, 1993

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