A Flexible Protocol Architecture for Multimedia Communication in ATM-Based Networks[*]

Markus Hofmann, Claudia Schmidt
Institute of Telematics, University of Karlsruhe, Zirkel 2, 76128 Karlsruhe, Germany
Phone: +49 721 608 [3982, 3414], Fax: + 49 721 388097
E-Mail: [hofmann, schmidt]@telematik.informatik.uni-karlsruhe.de

Abstract: *Emerging applications are characterized by highly different service requirements. Most importantly, they require specialized end-to-end quality of service and multicast support from the communication system. The BERKOM-II projects comprise application support as well as an enhanced communication system with multicast and QoS support. In the first section of this paper, the BERKOM-II projects are shortly introduced, and the Multimedia Transport System (MMT) is described in some detail. Section 2 gives an overview over QoS support and group communication of the XTP-Lite protocol. Finally, section 3 describes the implementation architecture.*

1 Introduction

The BERKOM (Berliner Kommunikationssystem) project is one of the most prominent Broadband ISDN trial projects worldwide. Several computer manufacturers and research institutes, such as Digital, Hewlett Packard, IBM, Siemens, and GMD (Gesellschaft für Mathematik und Datenverarbeitung), participate in the BERKOM project. The current phase of the project, which is named BERKOM-II, comprises three main areas of work (Figure 1):

- The *Multimedia Collaboration Service (MMC)* [1] supports joint working in a distributed environment. It allows users to share applications and to participate in audiovisual conferences.

- The *Multimedia Mail Service (MMM)* [2] facilitates exchange of multimedia documents including audio and video annotations.

- The *Multimedia Transport System (MMT)* [3] provides the communication platform for the above applications. It is based on ST-II, the Internet Stream Protocol Version 2, as network layer protocol [4], and a part of XTP (Xpress Transfer Protocol) [5], the so-called XTP-Lite, as transport layer protocol.

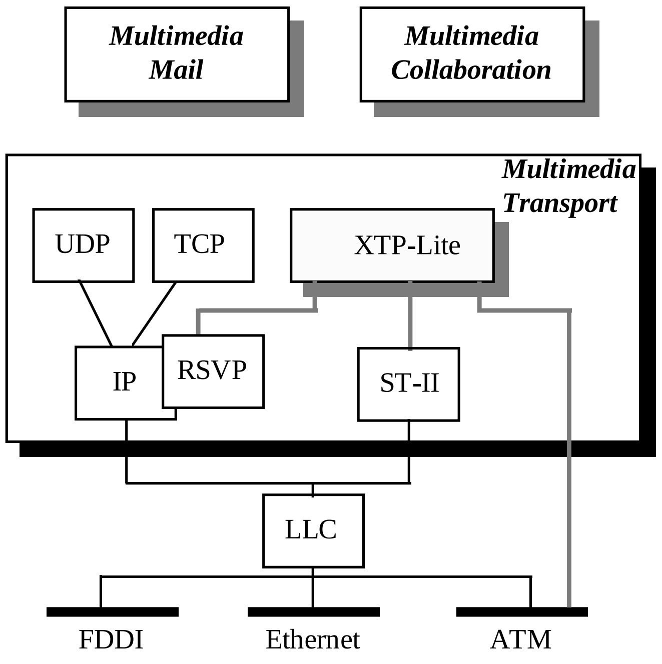

Figure 1: Protocol Architecture of MMT

The BERKOM-II Multimedia Transport System (MMT) is designed with an emphasis on communication between distributed multimedia applications. These applications simultaneously handle several highly diverse data streams, such as audio, video, and conventional data communication. Each of these data streams has different requirements concerning the Quality of Service

[*] In *Proceedings of 18th Biennial Symposium on Communications*, Kingston, ON, Canada, June 1996.

(QoS) provided by the communication system. The requested service can be described and negotiated at the MMT service interface by several QoS parameters such as throughput, maximum TSDU size, delay, and several reliability classes. Furthermore, a guarantee class indicates how the QoS parameters should be guaranteed. During connection establishment phase, the requested QoS parameters are negotiated between sender and one or more receivers. The negotiated QoS parameters described in the resulting *service contract* are guaranteed as long as the applications obey their traffic characterization.

Basic QoS guarantees are provided by resource reservation on network level. Currently, the MMT protocol stack uses the connection-oriented protocol ST-II for this purpose [4]. Other protocols, such as RSVP with IP [6], can be used instead. ST-II is based on so-called *streams*. Streams are routing trees from one sender to one or more receivers with associated quality of service. ST-II is an unreliable protocol that achieves efficiency by avoiding segmentation and reassembly. SCMP, the Stream Control Message Protocol, is a reliable control protocol for connection establishment and maintenance. As part of connection establishment, SCMP negotiates QoS parameters of a stream. For this purpose, a data structure, the so-called *FlowSpec* containing all QoS parameters, is distributed to all involved systems.

The connection-oriented transport layer protocol, XTP-Lite, enhances the basic service of ST-II in order to provide service flexibility. XTP-Lite supports unreliable and reliable duplex point-to-point connections as well as simplex multipoint connections. Furthermore, a variety of protocol functions suited for highly diverse communication requirements can be selected based on the service specification. For connection establishment, a 2-way-handshake or alternatively an implicit connection establishment can be performed. The activation of several mechanisms for error detection and correction is based on the required quality of service. Rate control supports the continuos transfer of multimedia data, and finally, special protocol mechanisms such as selective repeat are used to achieve an efficient transmission of data.

Generally, the MMT stack is able to operate over several networks. However, only networks that allow resource reservation, such as ATM, are able to guarantee a certain quality of service. Currently, the MMT protocol stack works directly on top of ATM. Private ATM networks of the project partners are interconnected with the public German Telekom pilot. First tests between Berlin and Karlsruhe have shown significant improvements with respect to the quality of communication compared to traditional data transfer via Internet.

2 QoS and Multicast Support

The MMT protocol architecture incorporates support for Quality of Service management and multicast communication. Additional functionality has been added to realize these tasks.

2.1 QoS management in XTP-Lite

Guaranteed quality of service from application to application is based on several QoS management tasks inside the communication system [7]. Apart from the enhanced service interface with several QoS parameters, XTP-Lite provides *QoS mapping* of service parameters on appropriate parameters of underlying protocols. The BERKOM-II project defines similar parameters on XTP-Lite and ST-II level, and thus, these parameters can be mapped in a straightforward manner. Mapping of ST-II parameters on ATM signaling parameters [8] is a more complicated task [9]. The XTP-Lite protocol was designed to work over several protocols (e.g., ST-II, ATM, RSVP/IP), and therefore alternative mapping functions were developed to map XTP-Lite parameters on parameters of each of these protocols.

Normally, in XTP-Lite *QoS negotiation* is part of the connection establishment procedure, but additionally, parameters can be modified and re-negotiated during data transfer. *QoS maintenance* is achieved by selecting appropriate protocol functions and parameters. Additionally, QoS parameters have to be monitored continuously during data transfer. If a QoS violation is detected, protocol functions and parameters are adapted accordingly, and in the case of a major violation, the application is informed. For this purpose, a *QoS monitor* was designed that operates with XTP-Lite as well as with other protocols [10], [11].

2.1.1 The XTP-Lite Service Interface

XTP-Lite provides an enhanced service interface in order to specify and negotiate a certain quality of service between the application and the MMT protocols. Several QoS parameters are applied at the service interface and allow for a flexible selection of services. In detail, the QoS parameters

- throughput: number of TSDUs (transport data units) per second,
- delay: end-to-end delay of one TSDU in milliseconds
- maximum TSDU size in bytes, and
- reliability

can be specified separately for each direction of a duplex connection.

The first three parameters form the so-called performance parameters. Each of them is specified by an interval ranging from a desired value (*value*) to a hard bound (*limit*). During QoS negotiation, the desired value is reduced based on the available resources, but if it reaches the specified limit, negotiation has failed and the connection is rejected.

For the reliability, five separate classes have been defined ranging from an unreliable service with or without error indication to a totally reliable service. The classes are defined as follows:

- type 0: ignore corrupted and lost TSDUs
- type 1: ignore corrupted TSDUs, indicate lost TSDUs
- type 2: indicate corrupted and lost TSDUs
- type 3: ignore corrupted TSDUs, correct lost TSDUs
- type 4: correct corrupted and lost TSDUs

All negotiated QoS parameters are guaranteed by the MMT protocols for the lifetime of a connection according to a so-called *guarantee class* selected by the user at connection establishment time. The guarantee class distinguishes between best effort and guaranteed service. In the best effort class, the QoS parameters must be guaranteed for an average traffic, while short QoS violations are allowed. In contrast, the guaranteed class allows no QoS violations, even under worst conditions.

2.1.2 Mapping XTP-Lite parameters on ST-II parameters

In order to support QoS parameters specific to the transport service interface, QoS mapping is required. XTP-Lite has to translate QoS parameters visible at its own service interface into corresponding parameters of the underlying layer. In this section, the mapping of XTP-Lite parameters on ST-II parameters is discussed. The derived parameters are then filled into the ST-II FlowSpec. Network layer and transport layer parameters of MMT are very similar, as shown in Table 1. Both layers use a limit and a desired value for SDU size, throughput, and delay, and a guarantee class to describe the desired level of guarantee. Furthermore XTP-Lite provides several reliability classes. The main difference between QoS parameters at XTP-Lite and ST-II level is the data unit size they refer to. Network layer parameters relate to NSDUs (network service data units) instead of TSDUs (transport service data units). Since the transport layer is able to segment data units, the TSDU size generally differs from the NSDU size. Thus, the transport layer has to map TSDU-based values specified by applications on NSDU-based values of the ST-II FlowSpec.

QoS parameters	*XTP-Lite*	*ST-II*
guarantee classes	best effort, guaranteed	best effort, guaranteed
reliability classes	class 0-4	-
maximum service data unit size	TSDU size	NSDU size
throughput	TSDUs / s	NSDUs / s
delay (in ms)	per TSDU	per NSDU

Table 1: XTP-Lite and ST-II QoS parameters

Obviously, there is no mapping required for the selected guarantee class since XTP-Lite and ST-II use the same classes. Furthermore, the reliability class need not be mapped because reliability is only supported by protocol mechanisms of XTP-Lite. The remaining three parameters refer to the TSDU size on one hand and the NSDU size on the other hand. If no segmentation and reassembly are necessary in the XTP-Lite layer, mapping of these parameters can be done straightforward. However, if a TSDU is segmented in XTP-Lite and transferred over the network in several NSDUs, mapping becomes more complicated.

Considering throughput, the number of required NSDUs must be derived in the first step as shown in Formula 1. The value *number_NSDUs* describes how many segments are needed to transmit one TSDU of the length *TSDU_size* considering a maximum length of NSDUs (*NSDU_size*) given by ST-II. Moreover, the protocol overhead of XTP-Lite (*XTP_header_trailer_size*) included in each NSDU is taken into account in the formula.

$$number\_NSDUs = \left\lceil \frac{TSDU\_size}{NSDU\_size - XTP\_header\_trailer\_size} \right\rceil$$

Formula 1: Number of NSDUs

In the second step, the throughput requested from ST-II can be derived as shown in Formula 2. The throughput at ST-II level is measured in NSDUs per second, and thus must be *number_NSDUs* times higher than the throughput at the XTP-Lite service interface measured in TSDUs per second.

$$NSDU\_throughput = TSDU\_throughput * number\_NSDUs$$

Formula 2: Throughput in NSDUs per second

The delay parameter of the ST-II level is calculated in a similar fashion. Furthermore, mapping functions for XTP-Lite over ATM, or alternatively, XTP-Lite over ST-II over ATM have been derived.

2.1.3 QoS monitoring in XTP-Lite

In order to measure the QoS parameters achieved by the protocol stack and to detect QoS violations of the service contract, we have developed and implemented a monitor for high performance protocols [11], that is currently under test with the MMT protocol stack. In order to keep up with the high data rates of high performance protocols, it is placed into an autonomous entity that communicates asynchronously with the monitored XTP-Lite protocol. Logically, the monitor functions can be divided into two parts. A monitoring entity located at the sender is able to detect whether the application obeys the traffic specification. Additionally, a second entity at the receiver is able to detect QoS violations of the communication system by continuously measuring end-to-end QoS parameters. The monitor operates event driven, i.e., XTP-Lite collects relevant information, reads the system clock (which can be done in one CPU cycle on modern workstations), associates collected monitoring events with a time stamp and informs the monitor entity. Subsequently, this monitor computes QoS parameters and checks for QoS violations. Thus, time critical system interrupts are avoided and high performance can be achieved. Currently, we are able to monitor the QoS parameters throughput, delay, jitter, and several error-related parameters of simplex data streams. Each of the monitored QoS parameters is described by a so-called QoS vector including minimum and maximum limits, a lower and upper threshold, an average value, an average interval (defining the number of consecutive data units over which the average is computed), a type of service class (statistical, deterministic, best effort), and a fractional bound. Reactions of the monitor can be configured by a monitoring policy. Possible reactions are periodical report, report on demand, immediate report of QoS violations, or report of warnings.

The QoS monitor comprises three entities, that are able to work independently. They are grouped around a QoS-MIB where all QoS-related information is stored. For each monitored data stream, the MIB includes the QoS limits of the service contract as well as measured values. The core of the QoS monitor is a monitor entity, that computes QoS parameters from protocol information, updates QoS parameters in the QoS-MIB, and detects QoS violations. Furthermore, it performs all communication with the protocol entity. The two other entities are a presentation entity and an SNMP agent. The presentation entity graphically presents QoS parameters stored in the QoS-MIB. It is activated and solely controlled by a human user. The SNMP agent allows an integration of the monitor into traditional SNMP management.

2.2 Multicast support

In addition to traditional peer-to-peer communication, XTP-Lite also supports reliable and unreliable multicast communication. It provides a simplex data flow from one transmitter to an arbitrary number of receivers. The management of communication groups is brought into line with the BERKOM specific application semantics. The conference system of MMC, for example, realizes conference management at application level. The so-called *Conference Manager (CM)* maintains a list of all users participating in the conference. Hence, the transmitting MMT user is able to give the identity of all multicast receivers at connection establishment. XTP-Lite also supports dynamic group membership. It is possible to add or to delete an arbitrary number of receivers during the lifetime of a connection. However, the admission of receivers is always sender-initiated. Therefore, group management in XTP-Lite is called *sender-based*.

Extended protocol mechanisms are essential to provide efficient data transfer towards several communication participants [12]. Rate and flow control, for example, have been modified to support an arbitrary number of receivers. The multicast transmitter gathers control information from the receiver set and evaluates it according to the application specific group semantic. If an all-reliable service is required, for example, the values for the error control algorithm are evaluated such that the worst case values for retransmissions are taken from the set of received control packets. This means that the multicast sender retransmits all of the lost data reported in the set of control packets from the receivers. It is also possible to include just a subset of returned messages in traffic control. The so-called *active group* is made up from receivers whose returned information is used by the transmitter to control an association.

There are different ways for a receiver to leave a multicast connection. When it is no longer interested in receiving any data, the receiver immediately disconnects from the association. The transmitter can also force a certain receiver to immediately drop out of the association without performing any retransmissions. Once all data has been sent, the transmitter initiates a graceful release ensuring correct delivery of all data to all receivers.

The traffic specification is negotiated during connection establishment. Conflicts caused by different capabilities of several receivers are resolved according to the given group semantics. XTP-Lite also supports QoS renegotiation during an in-progress multicast association. At any time during the multicast connection, the transmitter can initiate a renegotiation of the traffic specification. Those receivers that do not accept this change drop out the multicast group.

3 Implementation Architecture

One objective of the BERKOM-II project is the provision of a transport system suitable for a variety of network environments. Therefore, it must be possible to rapidly switch between different network delivery services, such as ST-II, IP and ATM (Figure 1). The implementation should also be available on different hardware platforms running different operating systems. The support for such a wide range of system environments is facilitated by a modular, flexible implementation architecture shown in Figure 2. The XTP-Lite implementation is based on several system specific modules surrounding the protocol core. The protocol core consists of the protocol state machine and several modules for management tasks.

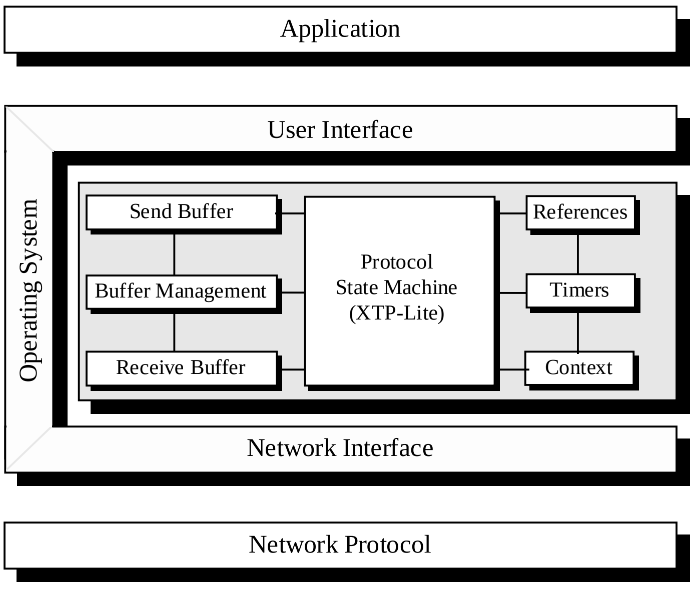

Figure 2: Implementation Architecture of XTP-Lite

The separation of user, operating system, and network specific functions into external modules facilitates integration of the protocol core into different system and network environments. However, it is not necessary to change the implementation itself. An unmodified XTP-Lite implementation has been run successfully on top of

ST-II as well as on top of IP just by modifying the network interface module. In the same manner, the protocol can be adapted to different operating systems. Specific modules have been implemented for Digital UNIX, Ultrix and Linux. Several modules have been implemented to support different user interfaces, like BSD Sockets API or X/Open Transport Interface API (XTI). The performance of the XTP-Lite implementation has been improved significantly by using shared memory objects for communication between the application and the protocol core. The application programmer links the MMT library providing shared memory objects to its code. User data to be transferred is written directly into the shared memory. The application signals this event via sockets to the protocol daemon. As one parameter of this call, the application passes the address of the shared memory object to the protocol daemon. Therefore, it is not necessary to copy user data from the application's address space into the address space of the daemon. Write and read access to the shared memory is controlled by semaphores. Another important factor is the design and the structure of send and receive buffers. Performance measurements have shown advantages for a combined buffer management based on hash tables and ordinary lists.

4 Recent Activities

An API has been implemented to directly access the ATM interface on Digital ALPHA workstations running Digital UNIX. Based on this API, a network interface module for direct access to ATM is being realized. This permits to run XTP-Lite directly on top of ATM. Extended measurements and performance evaluations will be performed to investigate the behavior of XTP-Lite in cell-based high speed networks.

Acknowledgments

The authors would like to thank Michael Fuchs, Jochen Klein and Rene Michel for implementing the main part of XTP-Lite and Kay Uwe Fischer for valuable discussions on QoS mapping.

References

[1] BERKOM Working Group: *The BERKOM-II MultiMedia Mail System (MMM), Version 3.0,* BERKOM Working Document, 1993.

[2] BERKOM Working Group: *The BERKOM-II MultiMedia Collaboration System (MMM), Version 3.3*, BERKOM Working Document, May 1995.

[3] BERKOM Working Group: *The BERKOM-II MultiMedia Transport System (MMT), Version 5.0,* BERKOM Working Document, May 1994.

[4] L. Delgrossi, L. Berger: *Internet Stream Protocol Version 2 (ST2), Protocol Specification - Version ST2+,* RFC 1819, August 1995.

[5] XTP Forum: *Xpress Transport Protocol Specification, Version 4.0,* XTP Forum, Santa Barbara, March 1995.

[6] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin: *Resource ReSerVation Protocol (RSVP), Version 1 Functional Specification,* Internet Draft, March 1996.

[7] C. Schmidt, M. Zitterbart: *Towards Integrated QoS Management*; Proceedings of the 5th IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems, Cheju Island, Korea, August 1995.

[8] ATM Forum: UNI (User-Network Interface Specification), Version 3.1, 1994.

[9] M. Hofmann, C. Schmidt, H. Wiltfang*: Mapping the MMT FlowSpec on ATM Quality-of-Service Parameters*, BERKOM Working Document, November 1994.

[10] K. U. Fischer: *Monitoring of QoS Parameters in XTP-Lite,* Thesis at Institute of Telematics, University of Karlsruhe, September 1995 (German).

[11] C. Schmidt, R. Bless: *QoS Monitoring in High Performance Environments,* Proceedings of the Fourth International IFIP Workshop on Quality of Service - IWQoS'96, Paris, 6 - 8 March 1996.

[12] M. Hofmann*: A Generic Concept for Large-Scale Multicast*, Proceedings of International Zurich Seminar on Digital Communication, Springer Verlag, February 1996.