Implementation of a B-ISDN/ATM connectionless data service in an Internet environment

Nail Kavak <nail.kavak@haninge.trab.se>
Kim Laraqui <kim.laraqui@haninge.trab.se>
Ala Nazari, <kim.laraqui@haninge.trab.se>
Patrik Ernberg <pernberg@sics.se>

Abstract

We specify and implement an efficient ATM data communications' platform using a broadband connectionless protocol and ATM Adaptation Layer 5. The implementation is performed on a Telia Research ATM experimental network. One important design policy is to keep things "simple", mainly in order to demonstrate that the introduction of ATM technology into LANs can be an evolutionary process that respects past, current and future investments in LAN technologies.

A consequence of this design policy is the projection, in little time, of the protocol specification onto well-known platforms (SUN Sparcs, Fore ATM Switches), using well-known protocol implementation approaches (bsd Unix sockets), in a well-known LAN environment (a TCP/IP-based LAN). This projection serves an early and realistic validation of our approach.

I. Data communication over ATM

One of the more important applications of B-ISDN will be LAN/MAN interconnection over ATM. ITU-TSS and ETSI is currently working on a solution in which the B-ISDN customer communicates via a connectionless BDS (Broadband Data Service) logical and/or physical overlay network [1] [2]. A complete specification is not expected to be available before late 1994.

II. First design decision - Indirect vs. direct

ITU-TSS draft recommendation I.364 [3] identifies two different BDS configuration types.

II.A. Type 1 - The "indirect method"

The "indirect method" requires that the BDS be installed outside of the B-ISDN. It could be viewed as a "leased-line" approach where a more or less fixed bandwidth is assigned to the virtual path between associated LAN routers or servers. For details on the indirect method, we refer to other sources [2][4][5].

II.B. Type 2 - The "direct method"

The "direct method" implements a connectionless service function by means of BDS servers (CLSs, Connectionless Servers, as they are referred to in I.364). These operate within the ATM network and are interconnected either permanently or semi-permanently. Each server performs routing by determining the next hop to which a cell must be delivered in order to reach its final destination such as a router or a B-ISDN terminal.

The main reason for choosing the direct method is economy of scale. With the direct method, LANs will require only one ATM connection to a BDS server. Given that they are (semi) permanent between servers, the total number of connections required in a BDS network by the end-users is substantially reduced [1][2].

A second reason is that LAN traffic is considered a good alternative when maximizing utilization of the resources available in ATM networks through the effects of statistical multiplexing [4].

A third reason is that we want to project the connectionless paradigms of LANs onto the connection-

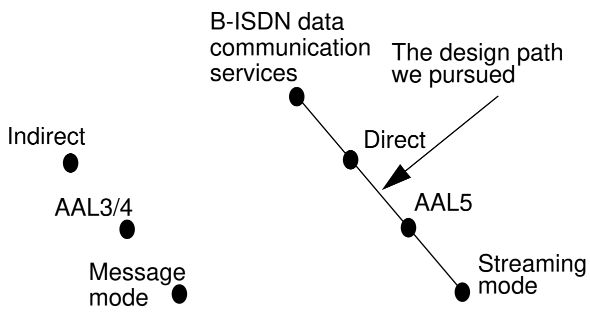

Fig.1

The design path we pursued for the specification and implementation of a data communication service on top of ATM in an Internet environment.

oriented paradigms of ATM, primarily to see what problems and solutions we come across in doing so. Designing connectionless access mechanisms for Gigabit networks is a real challenge that leads to the discovery of subtle issues inherent to bursty traffic typical of LANs.

III. Second design decision - AAL5 vs. AAL3/4

Two ATM Adaptation Layer (AAL) protocols are relevant in this study: AAL3/4 and AAL5. For further details, we refer to other sources [4][5][6]. In this paper, we assume readers have basic knowledge of ATM.

The main difference between the two protocols is visualized in Fig.2: AAL3/4 and AAL5 protocols differ in their packeting, fragmentation, and reassembly of user data to/from ATM cell payloads (or Segmentation and Reassembly, SAR, PDUs) and multiplexing.

III.A. AAL3/4

The CPCS (Common Part Convergence Sublayer) PDU receives user data and adds its own header and trailer (Fig.2). In message mode, the entire user data block is received before anything is passed down towards the ATM layer. In streaming mode, the user data block may be processed in chunks. The AAL3/4 CPCS header and trailer consist of various fields [7] which we will not describe in this report.

After a procedure in which the CPCS PDU is segmented into pieces of 44 octets these octets are combined with a two octet header and a two octet trailer to complete the AAL3/4 SAR PDU. Most importantly:

- The SAR PDU header contains a MID (Multiplexing ID) field that enables the sender and recipient to multiplex different user data sources and sinks over the same ATM connection.
- The SAR PDU trailer contains a 10-bit CRC code. This CRC is calculated over the entire SAR-PDU.

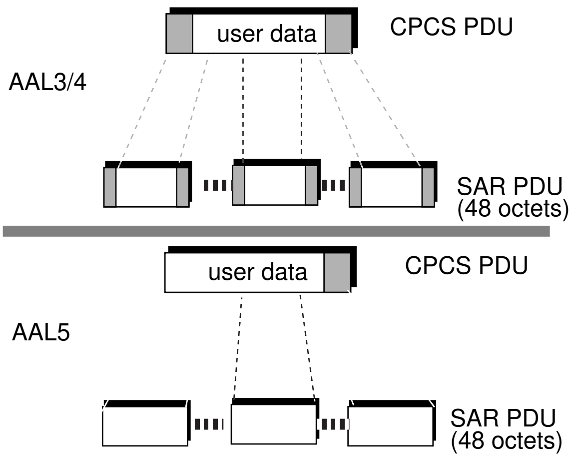

Fig.2

AAL3/4 vs. AAL5. Payloads are colored white, and headers/trailers are grey.

III.B. AAL5

AAL5 only adds a trailer to the user data (Fig.2). The trailer contains, among other things, a CRC field which is calculated over the entire CPCS PDU (CRC32).

However, the fragmentation of this PDU into ATM cell payloads (AAL5 SAR PDUs) is more straightforward than for AAL3/4, since the entire cell payload is used for the CPCS PDU. No additional CRC is provided into the ATM cell other than the one inherent to the ATM cell header. The decision to use AAL5 instead of AAL3/4 as the protocol on top of which to design our BDS, is based on several observations:

- The MID field of the AAL3/4 SAR PDU is no longer argued for as necessary because of the wide range of VPI/VCIs to employ as MIDs. Without the MID field one gains a payload of 10 bits/cell.
- AAL5 appears to be the AAL of choice among ATM equipment providers and several major operators.
- If we assume quality of service of ATM networks is very high, using AAL5 might just save us a lot of bandwidth and processing overhead for BDS.
- AAL5 is simple and does not involve too many PDU header and trailer shufflings.

IV. Third design decision - Streaming vs. Message mode

A drawback of AAL5 is its provision of the CRC in the trailer of the CPCS PDU (Fig.2). If a BDS server in the B-ISDN backbone wishes to check for validity of BDS data conveyed by AAL5 entities, it has to reassemble the entire CPCS PDU inside the BDS network.

The advantage of AAL3/4 is that the SAR PDUs themselves contain CRC fields for the entire SAR

PDU. This implies that a BDS server in the network can check individual cells for errors and upon detection of problems, discard all consecutive cells belonging to the CPCS PDU.

We decided not to reassemble the AAL5 CPCS PDU at all within the BDS overlay network. This means that everything in the BDS network is in streaming mode, even if BDS users operate in message mode. The underlying assumption is that the quality of service provided by ATM networks is good enough to leave most of the control functions to the higher layers on end-systems.

This solution will have to be supported by an intelligent tuning of system parameter such as maximum length of BDS data units, buffer sizes, and load splitting techniques. At present, our approach is indeed a “best effort” one.

V. Design of a BDS overlay network

V.A. Service

The BDS has the following service primitives:

- BDS_UNITDATA.request (Source, Destination, User Data, QOS).
- BDS_UNITDATA.indication (Source, Destination, User Data, QOS).

Source and Destination are E.164 addresses (64-bit ISDN numbers), and the maximum data size is most likely 9188 octets [3]. Candidate QOS elements are delay and loss.

Since we plan to extend the current solution with one in which AAL5 operates in message mode in the server itself, primarily to compare the two approaches, a combination of high delay and low loss could incite the network to use message mode instead of the default streaming mode.

V.B. Elements of BDS protocol operation

For cells to reach the BDS server it is assumed that the BDS ATM cells that enter the network are tagged VPI (Virtual Path Id.)=”BDS”. The ATM backbone (switches) must be configured so that all cells with VPI=”BDS” are forwarded to the server.

In Fig.3, we observe three different ATM cell types to be analyzed by the server. The BOM-cell (Beginning Of Message), indicates that the cell is the first one in a series of cells from a BDS packet. This cell is the only one in the server that has its payload analyzed. The analysis aims at identifying a VPI/VCI over which to route the cell, and is done by translating the E.164 address onto such a VPI/VCI pair.

Hence, our solution requires the E.164 address to be available in the first cell payload. The remaining cells are either COM (Continuation Of Message) or EOM (End Of Message).

Cell types are distinguished based on the PT (Payload Type or ATM-User-to-User, AUU) bit of the ATM cell header. PT=0 indicates BOM or COM,

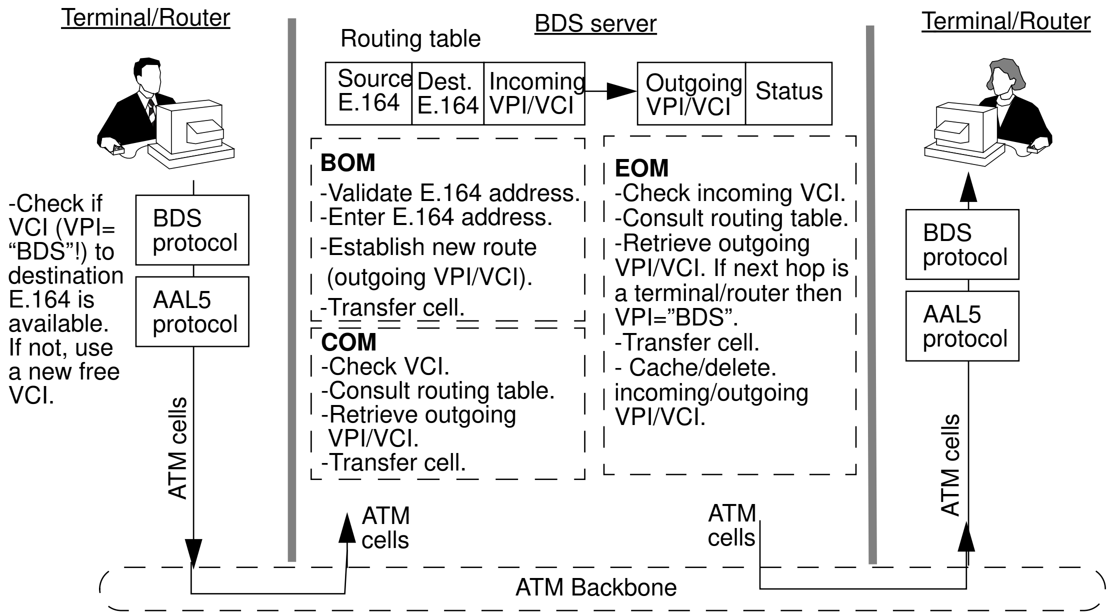

Fig. 3

The BDS server.

PT=1 is for EOM. For the server to distinguish between BOM and COM, it must rely on correct reception of EOM. If an EOM is lost, the server may mistake a BOM for a COM.

VI. The Telia Research BDS testbed

The testbed (Fig.4) consists of one ATM switch and 4 workstations connected to the switch by means of fibres. The switch is subdivided into several logical switching domains, allowing us to emulate more complex interconnection scenarios.

BDS servers are implemented as real overlay network entities, i.e. in workstations. A workstation may be configured as BDS terminal and/or BDS server, depending on the way its operating system kernel is set up and installed. Early 1994, it is expected that the Stockholm Gigabit Network (SGN) [8] will be available for large scale tests of, among other things, the BDS (Fig.4).

Other validation activities of the BDS involve verification [9] and simulations for which we are mainly interested in the application of chaos and fractal theory for the simulation of packet data networks [10].

VII. Implementation of a BDS

Inspired by the Internet community's incitement to research, implement, and use protocols before standardizing them [11], we have, throughout the research and design process, kept in mind the projection of specifications onto technology. Implementation activities force the designers to closely evaluate the specifications and discover many subtle issues.

A typical example is our decision to implement a streaming mode solution in order to avoid time and memory consuming stateful processing of AAL5 CPCS PDUs in the servers. A second example is the ongoing release of AAL5 products from numerous manufacturers which further convinced us to select that protocol as a basis for the BDS, particularly in the light of future trials over the SGN and other research sites. Another source of inspiration has been the recent studies on Internet protocols in Gigabit networks and in environments similar to ours [11][12].

The implementation of the BDS code has been performed on top of Fore's host interface [13] [14] code version 2.1.1. Our aim has been to use as much of the existing Fore code as possible. We have therefore utilized its segmentation and reassembly routines and data structures with slight modifications to accommodate for VP support. Such support is necessary since we tag BDS cells VPI="BDS" as explained earlier.

Other modifications involved the reassembly code to support BDS server functionality. Our main contributions have been to provide a thin protocol layer on top of the AAL5 code provided by Fore as well as a Raw socket interface and an IP interface [15] that allow access to the protocol. The Raw socket interface is intended mainly for testing and network management purposes.

VII.A. The core protocol - Terminal code

The terminal code essentially consists of an output routine (`BDS_output()`) and an input routine (`BDS_input()`) that respectively construct and parse a BDS PDU header.

The functionality of the output code is to add a BDS PDU header to the upper layer protocol. The header contains, among other things, E.164 format source and destination addresses as well as an upper level protocol identifier. The idea is to create a new bsd kernel [15] memory buffer (mbuf) with this information and then tag on the upper layer user data mbuf.

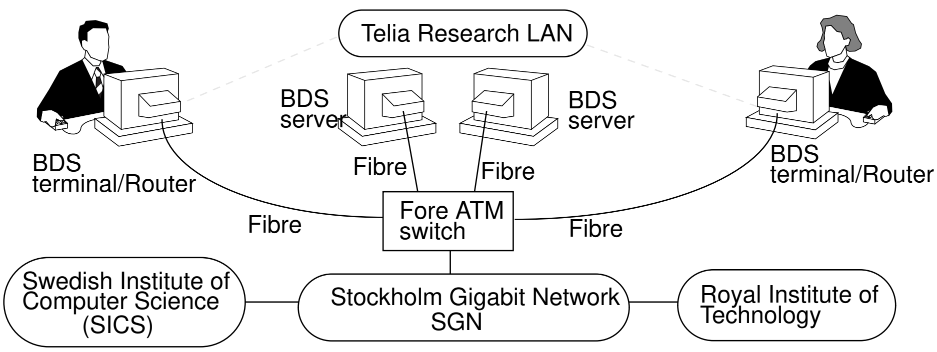

Fig. 4

The Telia Research testbed for studies on BDS overlay networks.

The output routine then calls the lower AAL5 protocol. Since this lower level protocol requires a VPI and VCI (Virtual Circuit Id.) pair on which to send, a table lookup has to be performed that returns the VPI/VCI pair corresponding to a given E.164 destination address.On the input side, the lower layer protocol delivers a mbuf chain with a BDS PDU header. The header is parsed by the input routine and then sent to the appropriate upper layer protocol depending on the protocol identifier field in the header.

We have chosen to run this code at the hardware interrupt level as only a marginal number of instructions have to be added to the original hardware interrupt level code to perform the required header parsing. At present, the two upper layer protocols that are available are IP and a Raw socket interface.

VII.B. The core protocol - Server code

Since the BDS server operates on a per cell basis, the code had to be introduced at the lowest level in the Fore driver. When a cell header has been read from the input fifo queue, the code checks if the VPI is one that has been designated to be part of the BDS network. If so, a table lookup has to be performed in order to find the appropriate outgoing VPI/VCI pair.

For efficiency reasons, entries in a table are maintained on a session level (using timeouts) rather than on a packet level.If the cell is part of the middle of a BDS packet (i.e. COM), this amounts to indexing a table. More complicated procedures are required if the cell is the first or the last part of a BDS packet. At this point, some table modification has to be performed based on the other arguments in the lookup function (a pointer to the cell contents as well as a flag indicating if the cell header has an end of message bit set (EOM)). See also Fig.3.

VII.C. Modifications to the Fore driver code

The original Fore reassembly data structure did not support VPIs and merely consists of an array of VCI entries. Each entry records what type of adaptation layer the VCI runs and what upper layer protocol it should be sent to. In the case of AAL5 VCIs, the incoming cells are reassembled in the entry until a complete AAL5 message has been delivered. At this point the entire AAL5 packet is sent to the next protocol layer.

In order to support VPIs, we modified the reassembly data structure to be a matrix, where each entry is indexed by a VPI/VCI pair. All VCI's that are part of the BDS VPI are marked as BDS entries. Terminal BDS cells will thereby be identified and collected in their respective entries. When a complete AAL5 packet has been reassembled in a BDS entry, the packet is sent to the BDS input function as described earlier.

VII.D. Raw Socket interface

A Raw socket interface was chosen to give direct access to the BDS protocol. The main reason for using the Raw socket interface was that a lot of code

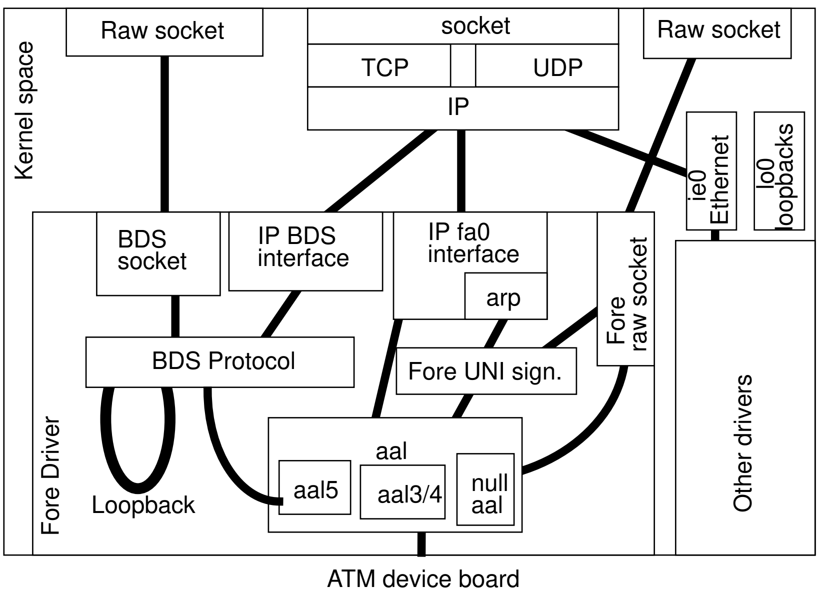

Fig. 5

Outline of software modules for the BDS implementation.

and data structures come for free, notably Protocol Control Block (PCB) handling routines and data structures.

The Raw socket interface works with the generic socket address (`sockaddr`) structure to describe source and destination addresses. Since we are working in a completely new address family based on E.164 addresses (ISDN numbers), a new socket address (`sockaddr_BDS`) that could be casted to the generic `sockaddr` structure was created. A protocol switch for this address family was also added. At present the only protocol available on this protocol switch is the BDS protocol (`BDS_PROTO_BDS`).

The socket flags have been defined such that messages are sent separately and the protocol presents the source address with each message. All the PCB handling is performed by already existing Unix bsd code and the generic `raw_usrreq()` (corresponding to BDS_UNITDATA.request) routine can be used without modifications.

The output routine (`raw_BDS_output()`) simply casts the generic source and destination `sockaddr` structures stored in the Raw PCB to `sockaddr_BDS` structures, and then calls a `BDS_output()` function with the appropriate arguments.

On the input side, the `BDS_input()` function currently assumes that all protocol identifiers in the BDS PDU header that are not IP are meant for the Raw socket interface. Thus, packets designated for this interface have their headers stripped off and the generic `raw_input()` routine is called that matches source and destination addresses with a corresponding PCB and socket identifier.

VII.E. IP interface

Introducing a new IP interface involves adding attach routines for the drivers of those interfaces, see Fig.5.

Since the Fore driver supports IP directly over the adaptation layer, a second IP interface had to be created for IP over BDS. In practice, a new `ifnet` structure is defined and attached using the `if_attach()` routine (`ifnet` structures define interfaces by specifying the interfaces' name, address, interface properties, routines, and statistics [15]).

This `ifnet` structure is very similar to the one defined for IP directly over the adaptation level, the differences being that another IP address as well as a new output routine are used.

The output routine has to make a table lookup in order to convert the destination IP address into the corresponding E.164 address. This table corresponds to the ARP table lookup that has to be done for IP over Ethernet. It then calls the `BDS_output()` routine with the protocol identifier argument set to IP.

On the input side, the `BDS_input()` function will recognize IP packets by the fact that the BDS PDU protocol identifier field is set to IP. These packets will then be queued on the IP input queue and an IP software interrupt is posted.

VII.F. User controlled parameters

Currently, the IP/ATM address resolution table can be modified from user space using an option to set sockets (`setsockopt`). Similarly, the VPI that is used for BDS traffic can be defined from user space using the option.

Further parameters that we plan to put under user control are, among others, LOSS_OF_ EOM, LOSS_OF_BOM, and LOSS_OF_COM to simulate losses of various cell types.

VIII. Testing the implementation

We have chosen to postpone serious performance tests [16] until the SGN is available for experiments as illustrated in Fig.4.

The first performance measurement was through the Internet PING application [17]. With the original Fore code, the results show a round-trip delay of 2 ms between two hosts. Our implementation yields the same result, with a ms resolution, despite the additional processing for VPIs, BDS, and table lookups. Further performance tests are under way, mainly with respect to throughput of the server.

Functional (internetworking) tests involve terminal-server-terminal tests, for which we run FTP and TELNET applications. By manipulating the workstations' and switch routing databases (Unix bsd /etc/hosts and others) we can further enforce our own LAN-routing on workstations so that one switch is logically subdivided into two. This means we can emulate a BDS network where two servers route BDS ATM cells between terminals (Fig.4).

All functional tests are encouraging and demonstrate the correctness of our specification under advantageous conditions.

The next step is of course to put the implementation under stress, e.g. to make sure that the ATM network starts to loose cells, as suggested earlier, to overload a server, etc.

IX. Discussions and directions for further work

Our specification is based on the view that the underlying ATM network provides a good enough quality of service to allow the BDS server to assume that the ATM cells generated from BDS terminals are essentially free from errors.

However, this minimalistic approach may face severe challenges. A typical example is when a BDS-generated ATM cell is lost, e.g. an EOM cell. At worst this results in the loss of consecutive BDS packets in the BDS network, which in turn may have serious repercussions on end systems. We are currently looking into different approaches to counter this problem.

A further problem is the validation of E.164 addresses. Since these are carried in the ATM cell payload, and we do not check for errors in payloads, this may be a serious argument against our solution and a valid argument for a solution in which BDS packets are reassembled by the BDS servers. For example, if a BOM is lost, the BDS Server will treat the COM that follows as a BOM. Also, if the BOM arrives in order, the E.164 addresses may still be corrupted.

One remedy could be to protect the E.164 addresses with a CRC32 in the BDS protocol header which fits into the BOM together with source and destination addresses. The CRC computation might however slow things down so much that streaming mode looses all its advantages as compared to packet mode. There is probably a balance to strike when choosing mode for the BDS server operation.

The main benefit of a real overlay BDS network is the ease with which direct BDS services can be introduced in ATM. However, this means that a port (in/out) on the switch to which the server is connected must be reserved for this "marginal" service. A physical overlay network also causes a duplication of BDS traffic volume from the point of view of the ATM switch. Ideally, the BDS server would be implemented within the switch itself; in fact, the BDS streaming mode server is a switch with the exception of the first cell for which it has to look into the payload. Finally, one important observation that we made when implementing the BDS, is that a real BDS overlay network will probably never operate in pure streaming mode. This is because the BDS server and terminals would find it too costly in terms of processing overhead, if at all possible, to enforce interrupts on a per cell basis.

One of the design issues we have to tackle is the size of buffers in the ATM backbone that are necessary to prevent congestion from bursty LAN traffic. Although a recent study [18] claims that a buffer size of 528 Kbyte per switch outport is sufficient, the real answer depends strongly on traffic characteristics, including packet size, burstiness, TCP window size, etc. Other intricate problems involve congestion and flow control, routing and addressing, multicasting, and management of BDS networks.

If we cannot come up with satisfactory solutions to these problems, we will have to reverse design choices of Fig.1. Ultimately, this may result in the use of dedicated ATM circuits between IP routers, in combination with a dynamic bandwidth reservation scheme and solid traffic shaping mechanisms at the edges of the ATM backbones.

X. References

[1] Nail Kavak, "Interconnecting 802 LANs through B-ISDN", INTERWORKING '93, S. Rao (editor), Bern, Switzerland, June 1993, ISBN 90 5199 1355.

[2] Peter Newman, "ATM LANs: The customer premises ATM network", 3rd B-ISDN Technical Workshop, Nice, France, April 1993.

[3] CCITT draft recommendation I.364, "Support of Broadband Connectionless Data Service on B-ISDN", August 1993.

[4] Martin de Prycker, "Asynchronous Transfer Mode - Solution for Broadband ISDN", Second edition, Ellis Horwood, 1993, ISBN 0-13-178542-7.

[5] Nail Kavak, Kim Laraqui, Ala Nazari, Patrik Ernberg, "Experiences and analyses of a connectionless broadband data server for provision of broadband data communication", To appear in the proceedings of INTERWORKING'94, May 1994, Sophia-Antipolis, France.

[6] Ala Nazari (editor), "Connectionless data services in B-ISDN", EURESCOM P302, Task 1.9, December 1993.

[7] CCITT recommendation I.363, "B-ISDN ATM Adaptation layer", February 1992.

[8] Björn Pehrson (editor), Proceedings of the 6th MultiG workshop, Stockholm May 1993, (KTH/ Electrum 204, S-16440 Kista, Sweden, Email: bjorn@it.kth.se).

[9] Gerard J. Holzmann, "Practical methods for the formal validation of SDL specifications", Computer Communications, Vol.15, March 1992.

[10] Ashok Erramilli, "Chaos, fractals and traffic", Proceedings of the 6th MultiG workshop, Stockholm May 1993, (refer to [8]).

[11] Craig Partridge, Stephen Pink, "An implementation of the revised Internet Stream Protocol (ST-II)", Internetworking: Research and experiences, Vol.3, No.1, March 1992.

[12] Olof Hagsand, Craig Partridge, Stephen Pink, "Making Sockets ready for multimedia", Proceedings of the 6th MultiG workshop, Stockholm May 1993, (refer to [8]).

[13] Edoardo Biagioni, Eric Cooper, Robert Sansom, "Designing a practical ATM-LAN", IEEE Network, March 1993.

[14] Fore Systems, Inc., R2.1 SBA-100 SunOS Device Driver and User's Manual, 1993.

[15] Samuel J. Leffler, Marshall Kirk McKusick, Michael J. Karels, John S. Quarterman, "The design and implementation of the 4.3 bsd UNIX OS". Addison-Wesley, 1989, ISBN 0-201-06196-1.

[16] Per Gunningberg, Mats Björkman, Erik Nordmark, Stephen Pink, Peter Sjödin, Jan-Erik Strömquist, "Application protocols and performance benchmarks", IEEE Communications Magazine, Vol.27, No.6, June 1989.

[17] Douglas E. Comer, "Internetworking with TCP/IP, Principles, protocols, and architecture", Prentice-Hall International, 1988, ISBN 0-13-470188-7.

[18] Johna Till Johnson, "ATM networking gear: Welcome to the real world", Data Communications, October 1993.

Acknowledgements

We would like to thank Bengt Ahlgren, SICS, for his insightful comments to the implementation work, and Gunnar Almgren, Telia Research, for his comments on drafts of this paper.

We would also like to credit the work of Stephen Pink, SICS, which formed an important source of inspiration.

Author information

Mr. Patrik Ernberg has been employed as a researcher at the Swedish Institute of Computer Science since 1989. His main research interests are in the specification, verification, and implementation of computer protocols. Address: Patrik Ernberg, SICS (Swedish Institute of Science), Box 1263, Kista, Sweden, Tel: +46.8.7521500, Fax: +46.8.7517230.

Mr. Nail Kavak is a researcher at the High-Performance Networking Group of Telia Research, Haninge, Sweden. He is currently a Ph.D. candidate at the department of Teleinformatics of the Royal Institute of Technology, Stockholm. His fields of research include communication architectures, high-speed network protocols and interworking. Address: Nail Kavak, Telia Research, S-13680 Haninge, Sweden, Tel: +46.8.7075565, Fax: +46.8.7075480.

Mr. Kim Laraqui is a researcher at the High-Performance Networking group of Telia Research, and a Ph.D. student at the department of Teleinformatics (Prof. Björn Pehrson) Royal Institute of Technology, Stockholm, Sweden. Currently, his research focuses on protocols and management systems for data communications over high-speed networks. Address: Kim Laraqui, Telia Research, S-13680 Haninge, Sweden, Tel: +46.8.7075568, Fax: +46.8.7075480.

Mr. Ala Nazari is a researcher at the High-Performance Networking group of Telia Research, and a part-time Ph.D. student at the department of Teleinformatics (Prof. Björn Pehrson) Royal Institute of Technology, Stockholm. Currently, his research focuses on protocols and routing for high-performance data communication networks. Address: Ala Nazari, Telia Research, S-13680 Haninge, Sweden, Tel: +46.8.7075569, Fax: +46.8.7075480.