High Performance Protocol Architecture

Walid S. Dabbous
INRIA Centre de Sophia Antipolis,
2004 Route des Lucioles, BP-93,
06902 Sophia Antipolis Cedex, FRANCE.
Tel: + 33 93 65 77 18, Fax: + 33 93 65 76 02
e-mail: Walid.Dabbous@inria.fr

Abstract

The development of high speed networking applications requires improvements to data communication protocols in order to efficiently provide the required services to the applications. In this survey paper, we will first give a rapid presentation of protocol optimization techniques and of some high speed transport protocols developed for specific application needs.

We will then present a new protocol architecture based on the "Application Level Framing" (ALF) and "Integrated Layer Processing" (ILP) concepts. ALF states that applications send data as a sequence of autonomous "frames", which will be at the same time the unit of transmission, the unit of control and the unit of processing. This allows for efficient design of application specific protocols. It also enables ILP i.e. the integration of multiple transmission control layers in a single loop, maximizing the efficiency of modern processors.

We will present both the architectural design aspects and the corresponding implementation problems. Practical experimentations with ALF and ILP will also be described and their benefits and limitations assessed. We will also present a new approach to network architecture namely the active networks, and discuss its pros and cons.

1 Introduction

The development of high speed networking applications such as audio and video conferencing, collaborative work, supercomputer visualization, transactional applications and WWW, requires efficient communication protocols. As networks proceed to higher speeds, the performance bottleneck is shifting from the bandwidth of the transmission media to the processing time necessary to execute higher layer protocols. In fact, the existing standard transport protocols (e.g. TCP) were defined in the seventies. At that time, communication lines had low bandwidth and poor quality, and complex protocol functions were necessary to compensate for the transmission errors. The emergence of high speed networks has changed the situation, and these protocols would not be designed in the same way today.

One the other hand, the transport protocols such as TCP or TP4 were designed to provide a so called "standard transport service" mainly by performing end to end error control for point to point connections: this includes detection and correction of packet losses, duplications, misordering and alterations. However, the application environment is changing. New applications (cf hereabove) with specific communication requirements are being considered. Depending on the application, these requirements may be one or more of the following: (1) high bit rates, (2) low jitter data transfer, (3) simultaneous exchange of multiple data streams with different "type of service" such as audio, video and textual data, (4) reliable multicast data transmission service, (5) low latency transfer for RPC based applications, variable error control, etc.

The above requirements imply the necessity to revise the data communication services and protocols in order to fulfill the specific application needs. In fact, applications may "classically" choose either the connection-less or the connection oriented transport services. In both cases, the application needs are expressed in terms of quality of service (QoS) parameters such as e.g. the transit delay or the maximum throughput. However, the applications should be involved in the choice of the control mechanisms and not only in selecting the parameters of a "standard" transport service. Placing most of the burden of network adaptation in the user equipment is in line with the "end to end argument", a key point of the Internet architecture. It contributes in keeping the network simple, and is very often the only way to scale up a complex system.

This should allow to build high performance communication modules. Performance here is defined as the efficient use of resources while still meeting the applications' requirements

according to Clark's definition in [Cla94]. We will detail later the issues that influence the performance of a communication system.

This paper is a survey of several techniques in the area of high performance protocols. However, it is focused on the Application Level Framing and Integrated Layer Processing (ALF/ILP) concepts and on the automatic generation of high performance communication systems. The rest of the paper is composed of five sections. In Section 2, we present the state of the art in high performance protocols: we start by studying the impact of the environment on performance and then we present different performance enhancement techniques ranging from protocol parameter tuning and adaptation algorithms support to the design of special purpose protocols. Section 3 discusses architectural design issues and shows why the layered model needs to be re-considered. We also present in this section, the ALF and ILP concepts as the foundations of a high performance protocol architecture. We discuss the possibility to achieve a performance gain by applying these concepts for protocol design and implementation. We also present some experimental results obtained with ALF/ILP, and we discuss their advantages and limitations. We also present a protocol compiler supporting these design rules. Section 4 presents the active networks approach to network architecture and discuss its advanatges and disadvantages. Section 5 concludes the paper.

2 High performance protocols

Early work on high performance protocols concentrated on the optimization of each layer separately. Concerning the transport protocols, several approaches have been studied such as the work on *enhanced environments* for protocol development, the *tuning* of standard general purpose protocols [Wat87, Col85] including the design of new transmission control algorithms [Jac88a], or the development of *specialized protocols* [PEI92, Cher86].

In fact, the protocol execution environment has a strong impact on the performance of the communication system. The environment overhead is mainly due to interrupt handling, timer management, buffer allocation and process scheduling. Work on enhanced execution environments concentrated on blocking on multiple events, low management overhead timers, good I/O buffer management and better resource scheduling [Cla94]. However, these issues are generally independent of the communication protocol and therefore will not be detailed in this section as

we focus only on protocol related issues.

2.1 Tuning of standard protocols

The general purpose transport protocols, such as TCP or TP4, support complex control mechanisms for connection management, error control and flow control on heterogeneous networks. The advantage is that these protocols may be used by a large number of applications requiring the standard reliable point to point transport service. The price to pay is a limitation of the protocol performance. A careful analysis of these protocols showed three types of issues to be resolved in order to enhance the performance:

- a bad choice of the acknowledgments strategy,
- a bad choice of the time-out values,
- limitations due to the lack of congestion control algorithms.

The *tuning* of these protocols consists of chosing the good values for the parameters and designing adaptive transmission control algorithms.

2.1.1 The acknowledgments

TCP uses positive cumulative acknowledgments indicating the number of the next expected octet. These transport level acknowledgments (ACKs) have a bad impact on performance and their number should be minimized. Clark proposed in [Cla82] to delay the emission of an ACK in certain cases with the hope of grouping several ACKs. Another modification of TCP was proposed in order to enhance the performance over links with high bandwidth×delay product: the use of selective ACKs. In the case of packet loss, the sender should only resend the lost packet and not all the unacknowledged packets [Jac88b], [Jac90]. Furthermore, a Negative Acknowledgment scheme (NACK) was proposed to be used in several protocols (e.g. NETBLT [Cla87b], XTP [PEI92]). A NACK has the semantics of an explicit retransmission request from the destination. This has the advantages of reducing the number of control packets (the NACKs) if there are no packet losses, and transferring the error detection problem to the receiver, thus allowing for more scalability in the case of a multicast transmission.

2.1.2 The timers

Transport protocols use several timers for connection management and error detection (TP4 uses 7 different timers). In addition to the timer management cost that we mentioned in the beginning of Section 2, a major problem is to find the "good" time-out values for the timers. This is particularly important for the retransmission timer. The time-out value should satisfy two incompatible goals: a rapid detection of packet losses and a reduction of the number of false alarms due to a delayed packet and not a "real" loss. The optimal value depends on the round trip time (RTT) between the source and the destination which, in turn, depends on several factors such as the network load, the routes taken by the packets and probably the packet size. This imposes that the time-out value be adapted to the RTT variation. The effective retransmission time-out in TCP is a function of the "smoothed" RTT. The smoothing factor filters the transient changes and its value determines the responsiveness of the algorithm to network changes. However, a difficulty may arise in the case of retransmission because of the lack of information whether the received ACK corresponds to the first or the second transmission of the packet. Therefore, the RTT sample corresponding to a retransmitted packet should not be taken into account when computing the smoothed RTT. In addition, the RTT variance should be taken into account to compute the retransmission time-out as described in [Jac88a]. Even with these tunings, it is still possible to have unnecessary retransmissions due e.g. to a packet blocked on the local network interface, a lost ACK or a bad RTT estimate. The main problem comes from the fact that timers are *local* means to estimate external events. Some researchers proposed to avoid the use of transport level timers for transmission control [Cla87a].

2.1.3 Congestion control

Transport level congestion control ensures that network resources are shared efficiently and network congestion is avoided. In fact, window based flow control algorithms tend to use large window values in order to "fill the pipe" in the case of networks with high bandwidth delay product. However, if several transport connections with large windows share a "low bandwidth" link, packet queuing delays may be observed in the router accessing the link, resulting in unuseful retransmissions. This positive feedback may cause a congestion if specific control algorithms are not applied. Van Jacobson proposed the "slow start" algorithm that has been implemented

in TCP since 1989. According to this algorithm, a transport connection starts by setting the window size W to 1. W is increased by 1 when an ACK is received in the exponential phase ($W < cwnd$). After the window reaches *cwnd* (the linear phase), it is increased by 1 each W ACKs. If the source estimates that a packet loss occurs (timeout expiry or the reception of three duplicate ACKs), the *cwnd* is set the $W/2$ and W is set to 1 and the algorithms restarts again. The implementation of this algorithm within the TCP version resulted in enhanced performance for the protocol.

2.2 Special purpose protocols

Some of the early work in the domain of enhanced transport *service* proposed the design of light weight special purpose protocols for specific application needs (e.g. NETBLT [Cla87a, Cla87b] for bulk data transfer and VMTP [Cher86] for transactional applications).

NETBLT or NETwork Block Transfer is a light weight transport protocol working on top of UDP/IP optimized for bulk data transfer over High speed LANs ([Cla87a], [Cla87b], [Lam87]). The main idea in NETBLT is to dissociate error and flow control (implicitly linked in all window based flow control protocols). A NETBLT client sends "blocks" in rate controlled packets without waiting for ACKs. When all the packets of a block are transmitted, the receiver requests the retransmission of lost or damaged packets. This receiver based error control removes the need for retransmission timers at the senders. Even if this protocol resulted in a reduced latency on high delay links, the lack of support in intermediate gateways limited this protocols to an experimental status. It was not supported over the Internet. The good performance for TCP after the support of slow start outdated the usefulness of NETBLT even on High Speed LANs.

VMTP or Versatile Message Transfer Protocol is another special purpose protocol that was designed for Intra-System communication (e.g. RPC for file pages access, etc) [Cher86]. The nice feature about this protocol was the support of multicast transmission at the transport level. In order to speed up the request/response time, no transport level connection is established. Stable addresses are used instead to provide a sort of semi permanent connection. Application level replies may be used to acknowledge the reception of the requests. This protocol was one of the first protocols in conformance with the Application Level Framing concept. However, the protocol could only be used in a local network environment, and even in this case the performance was not very good (4.5 Mbps over Ethernet) [Kan88].

XTP or Xpress Transfer Protocol was initially designed to be implemented on specialized hardware with execution efficiency as inherent part of the design process [Ches89]. Therefore, many syntactical and algorithmic choices of the protocol are oriented to facilitate high speed operation over low error rate networks. The hardware circuit implementation project was discarded in 1992, but the protocol survives and meetings of the "XTP-Forum" are held regularly, and XTP 4.0 revision was released in March 1995. In the previous releases, (3.6 and before) XTP was a "transfer" protocol integrating layers 3 and 4 processing [PEI92]. We concentrate the analysis of the XTP protocol on this transfer layer architecture. Integrating layers 3 and 4 reduces the packet processing overhead and enables several performance optimizations. Among these optimizations, (i) the possibility to calculate the checksum on the fly by using packet headers and trailers, (ii) a lightweight checksum algorithm that can be computed at high speed in software, (iii) fixed length fields for fast access to the header, (iv) fast connection establishment and release based on a "1-way handshake" mechanism and (v) fast de-multiplexing of the incoming packets by the exchange of local access keys. In addition, the XTP protocol provided several functional enhancement (e.g. the support of "pseudo" reliable multicast, rate control with the participation of intermediate routers, priority support and variable error control controlled by the sender and adapted to the application needs). However, there was no pre-defined XTP service. XTP provided a programmable toolkit (a set of mechanisms and not predefined strategies). The selection of the required functionalities is done by the application. The XTP protocol designers proposed new ideas (e.g. the layers integration) incompatible with the OSI "reference" model. The ideas represented a step toward integration. However, these ideas were discarded in the 4.0 version and XTP became a "normal" protocol.

The "special purpose protocols" approach has a major limitation: the diversity of the applications increases the complexity of the transport by the support of several protocols. Each application would then choose a protocol corresponding to its specific needs.

3 A new protocol architecture

The approaches presented in the previous section are not adequate to provide high performance communication systems for multimedia applications for two main reasons. The first one is that these protocols do not include the upper layers data processing overhead in the transmission

control loop. This overhead seems to be "the" effective bottleneck for these application (e.g. video coding or decoding for a video conferencing application, or data presentation for a data base replication). The second reason is that multimedia applications have specific communication needs that do not correspond to the standard general purpose or the proposed special purpose protocols.

The performance of workstations has increased with the advent of modern RISC architectures but not at the same pace as the network bandwidth during past years. Furthermore, access to primary memory is relatively costly compared to cache and registers and the discrepancy between the processor and memory performance is expected to get worse.

These observations lead some researchers [Cla90], [Gun91], [Abb93a] to reconsider the protocol architectural model and not only mechanisms within a specific layer or even the definition of a new protocol. The goal of the revision is to simplify the design of communication systems that take into account (i) the status of the network and the environment and (2) the applications needs. In this section, we discuss the issues concerning the design of this new protocol architecture.

3.1 "Layering considered harmful"

The OSI reference model adopted as a standard in 1980, was designed in the seventies. The layer organization reflected the hardware architecture of those days depicted in Figure 1. The artificial separation of concerns in entities called layers is partly due to the hardware architecture of the 70's. The service/protocol concept derives historically from the model presenting interfaces between different service providers. There would be a link, network, transport and session provider, perhaps all of which would have different potential vendors.

The physical and data link layer corresponded to mechanisms implemented by the modem and the driver. The network and transport layers were generally implemented in front-ends, and the session layer corresponded to the control of the dialogue via the channel between the mainframe (where the application hosted) and the front-end. Data presentation was integrated in the application (e.g. X.409 [Rec84b]).

With the advent of workstations (see Figure 2), the architecture of the implementations have changed, specially for the upper layers (transport and above). The transport layer is usually implemented in the kernel and accessible to the application through the socket interface. The

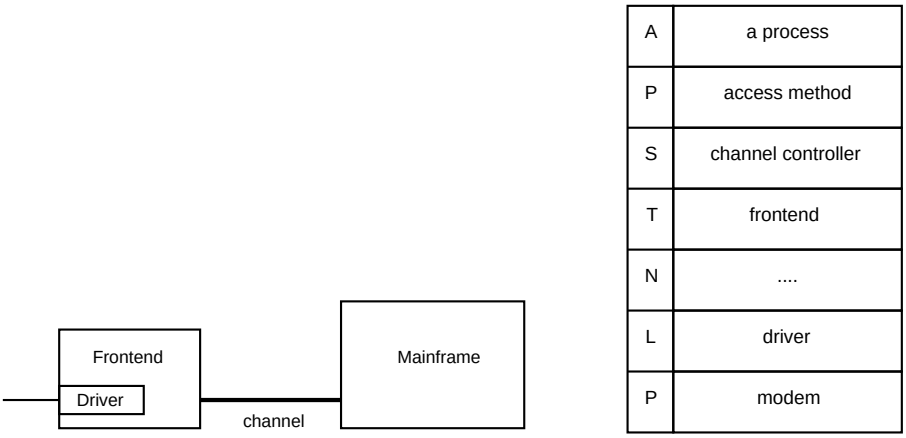

Figure 1: Hardware architecture of the 70s and the corresponding model

application is run at the user level and integrates specific parts of the presentation layer. The session layer presents, however, a problem. There is no need for such a layer in this architecture. In other words, the data transfer synchronization needs are better known by the application. It is therefore suitable to let the application control its synchronization needs rather than to delegate this control to the session layer. There is another problem posed by this layer in case it is implemented between the presentation and the transport: this would impose an asynchronous data exchange between these layers (which is known to degrade performance). This implies two data copies to/from the memory before data is transmitted on the network. These operations are relatively costly on RISC workstations. The session control should therefore be integrated within the application.

Even in a single vendor software implementation, one could accuse the layered model (TCP/IP or OSI) of causing inefficiency. In fact, the operations of multiplexing and segmentation both hide vital information that lower layers need to optimize their performance. In fact, the application should be able to exchange control information with the transport. This means that the transport should be aware of the application (just like VTMP replies served as ACKs for the requests). This allows to avoid bad decisions that may be taken by the protocol (e.g. send an ACK when there is a reply outstanding, or close the window or retransmit the packet after a packet loss during a video transfer). The two key issues for the performance enhancement are reducing the asynchrony and involving the application in the transmission control.

On the other hand, protocol processing can be divided into two parts, control functions

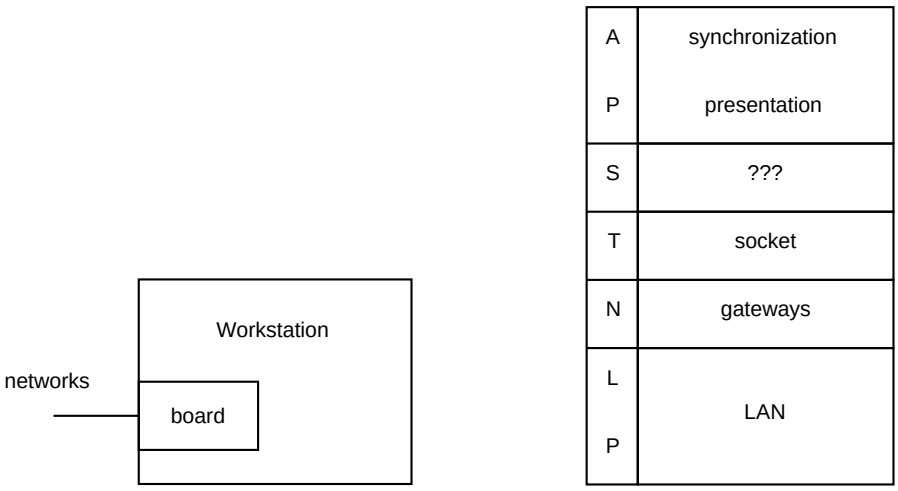

Figure 2: Architecture of the 80's-90's

and data manipulation functions. Example of data manipulation functions are presentation encoding, checksumming, encryption and compression. In the control part, there are functions for header and connection state processing. Jacobson et al. have demonstrated that the control part processing can match gigabit network performance for the most common size of PDUs with appropriate implementations [Cla89].

However, data manipulation functions present a bottleneck [Cla90], [Gun91]. They consist of two or three phases. First a read phase where data is loaded from memory to cache or registers, then a manipulation or "processing" phase followed by a write phase for some functions, e.g. presentation encoding. For very *simple functions*, e.g. checksumming or byte swap, the time to read and write to memory dominates the processing time. For other *processing oriented functions*, like encryption and some presentation encodings, the manipulation time dominates with current processor speeds. However, the situation is expected to change with the increase of processor performance: the memory access will be the major bottleneck rather than data processing.

The data manipulation functions are spread over different layers. In a naive protocol suite implementation, the layers are mapped into distinct software or hardware entities which can be seen as atomic entities. The functions of each layer are carried out completely before the protocol data unit is passed to the next layer. This means that the optimization of each layer has to be done separately. Such ordering constraints is in conflict with efficient implementation of data manipulation functions [Wak92].

In their famous SIGCOMM'90 paper [Cla90], Clark and Tennenhouse made the observation that if the performance of the communication system is to be enhanced the slowest part in the chain should be used in the best way. Many researchers agree that the data manipulation at the higher level is the bottleneck (presentation decoding, encryption, etc). However, transport level resequencing delays the processing of the PDUs received out of sequence until the lost or delayed PDU is received. An efficient utilization of the slowest part of the chain requires therefore that the application be able to process out of sequence packets.

3.2 ALF and ILP

Clark and Tennenhouse [Cla90] have proposed Application Level Framing (ALF) as a key architectural principle for the design of a new generation of protocols. ALF is in fact the natural result of advanced networking experiments, which showed the need for the following rules:

1. Efficient transmission can only be achieved if the unit of error and flow control is exactly equal to the unit of transmission. This rule is not satisfied for transport protocols over ATM networks. In fact, the control unit (the transport PDU) is not equal to the transmission unit (the ATM cell). Obvious penalties for violating this rule are unnecessary large retransmissions in case of errors and inefficient memory usage.

2. The key idea expressed in [Cla90] is that the unit of transmission should also be the unit of processing. Otherwise, large queues will build up in front of the receiving processes and eventually slow down the application.

3. We found, through experience with multimedia services, that adaptive applications are much easier to develop if the unit of processing is also the unit of control. Violating this rule means a separation between the application and the transmission control which may result in bad choices taken by the transport.

According to the ALF principle, applications send their data in autonomous "frames" (or Application Data Units (ADUs)) meaningful to the application. It is also desirable that the presentation and transport layers preserve the frame boundaries as they process the data. In fact, this is in line with the widespread view that multiplexing of application data streams should only be done once in the protocol suite. The sending and receiving application should define what data goes

in an ADU so that the ADUs can be processed out of order. The ADU will be considered as the unit of "data manipulation", which will simplify the processing. For example, an image server will send messages corresponding to well identified parts of the picture. When a receiver receives such a frame, it can immediately decompress the data and "paint" the corresponding pixels on the screen, thus minimizing response times. The ADU size should not exceed the minimum MTU (Maximum Transmission Unit) of all the subnetworks traversed by the ADU in order to avoid segmentation.

Integrated Layer Processing (ILP) is an engineering principle that has been suggested for addressing the increased cost of data manipulation functions on modern workstations. The main concept behind ILP is to minimize costly memory read/write operations by combining data manipulation oriented functions within one or two processing loops instead of performing them serially as is most often done today . It is expected that the cost reduction due to this optimization will result in better overall performance as it will reduce time consuming memory access. The interest of the ILP principle (and similar software pipelining principles such as lazy message evaluation and delayed evaluation) has been discussed in [Cla90], [Gun91], [Par93b], [O'M90], [Peh92], [Dab94a] and [Abb93a].

Previous work ([Gun91] [Dab94a] [Par93b]) has suggested that there is a performance benefit with ILP. The results show reduced processing time for one PDU, as much as a factor of five when six simple data manipulation functions were integrated. These reported results came from experiments that were isolated from the rest of the protocol stacks and where hand-coded assembler routines were used in order to control register allocation and cache behavior. Abbot and Peterson [Abb93b] use a language approach to integrate functions, but with less speedup. In [Par93b] only two functions are integrated, data copying and checksum calculation, but it is a real operational implementation of UDP. Experience with an implementation of the XTP protocol from the OSI95 project [Dab93] showed improved performance when ILP was used. This implementation is in user address space, which also demonstrates that such implementations can perform as well as kernel tuned implementations. One should go one step further and integrate several of the data manipulation functions in a complete, operational stack in order to understand the architectural implications and the achievable speedup.

ALF and ILP were proposed in 1990 and for the time being there are few manual implementations of a communication system based on both concepts integrated within applications

such as vic, ivs and wb. Several reasons make the design of a global framework for integrated implementation based on ALF/ILP concepts now more feasible and attractive:

- The increase of processor speeds and of the discrepancy between processor and memory performance is pushing toward the integration of data manipulation operations.
- Experience with an implementation of the XTP [Dab93] and TCP [Hogl94, Cas94a] protocols showed that user level software implementations can perform as well as kernel tuned TCP, while still keeping the flexibility of configurable software.
- Performance enhancements to the ASN.1 encoding facilitate the implementation of a presentation filter [Dab92].
- Multiple communication services including group communication and variable error control have been studied for a variety of applications (e.g. multimedia video conferences with shared workspace, mobile applications). It is very desirable to have a mechanism to customize communication systems to specific application needs.

Several experiments were made to evaluate the advantages and disadvantages of ALF and ILP. These experiments show that efficient implementation of distributed applications may benefit from new architectural considerations such as ALF and ILP. However, the ILP gain in a real implementation seems to be about 20 or 30 % [Brau95]. We may expect that the ILP gain will increase with the CPU speed, but for the moment it is not what one would expect. The most intersting results concerns ALF. Performance improvements were shown in some experiments [Diot95]. However, this was under high packet error rate condition. In fact, if there were no packet losses in the network, ALF complexity would impose a performance pernalty [Gag96].

On the other hand, ALF implies that communication systems are "tailored" for each application. This requires an automated approach for protocol code generation, which will allow to "easily" generate a large panoply of communication systems tailored to application needs. Therefore, a protocol compiler that generates efficient code is a step towards the support of new ALF based applications. A protocol compiler takes as input an abstract specification of a protocol and generates an implementation of that protocol. Protocol compilers usually produce inefficient code both in terms of speed and code size. However, [Cas96] describes a protocol compiler that uses a combination of two techniques, namely i) the use of a language compiler that generates from

the specification a unique tree-shaped automaton (rather than multiple independent automata), and ii) the use of optimization techniques applied at the automaton level, i.e. on the branches of the trees. The gain expected with this approach was evaluated on a real-life example, namely a working subset of the TCP protocol generated from a specification written in Esterel [Berr92]. The protocol code generated with this approach to that derived from the standard BSD TCP implementation. The optimized generated code executes up to 25 % fewer instructions than the BSD code for input packet processing while maintaining comparable code size (10 % bigger).

The compiler approach is attractive, but still faces portability limitations. This approach requires that the compiler generates portable code before it can be used in a general case.

4 Do we need a new network architecture?

In the previous sections we presented a survey of several approaches to enhanced communication performance, focusing on end-to-end protocol enhancements over an Internet-like network architecture. In this section, we discuss another approach to network architecture, namely the "active networks" approach in which the switches of the network perform customized computations on the messages flowing through them. This approach is motivated by both user applications, which perform user-driven computation at nodes within the network today, and the emergence of mobile code technologies that make dynamic network service innovation attainable. This approach will accelerate the pace of innovation by decoupling network services from the underlying hardware and allowing new services to be loaded into the infrastructure on demand [Ten96]. Active networks challenge the traditional Internet architecture based on providing the interoperability over the IP service. In fact, the computations performed within an "active network" can be dynamically varied; they can be user- and application-specific; and the user data is accessible to them. Network layer interoperability is based on an agreed program encoding and computation environement instead of a standardized packet format and a fixed encoding as with the classic IP service.

The main interest of this approach is not to impose a predefined network service, increasing the flexibility of the computation performed within the network. ALF states that the application should control the transmission. The "active networks" approach allows the application to control the network service. A combination of these approaches would facilitate the building of mobile

gateways (e.g. base stations for wireless networks). There is, however, a serious concern about the expected performance of the "active networks" approach. The increased flexibility at the routers has its cost, and may jeopardize the routing performance.

5 Conclusion

After a survey of several aproaches for enhanced protocols performance, this paper describes a high performance protocol architecture for efficient implementation of multimedia applications. This protocol architecture is based on ALF, which is a design principle allowing efficient processing of the application data units. Experiments with Application Level Framing (ALF) showed performance gain in the case of a heterogeneous Internet. The active network approach is promising for the design of programmable networks. However, the generalization of this approach may lead to performance limitations. The real challenge is therefore to build high performance active switches. In fact, the adaptive applications approach alone is not sufficient and a new network service should be supported by the network. Will the active network approach provide the support for that? – May be. Anyhow, there is a need to re-think what should be "in" the high speed networks of the future.

References

[Abb93a] Mark B. Abbott, Larry L. Peterson. "Increasing Network Throughput by Integrating Protocol Layers ", *IEEE/ACM Transactions on Networking*, Vol 1, No 4, October 1993.

[Abb93b] Mark B. Abbott, Larry L. Peterson. "A Language-Based Approach to Protocol Implementation", *IEEE/ACM Transactions on Networking*, Vol 1, No 1, February 1993.

[Berr92] G. Berry, G. Gonthier. "The Esterel Synchronous Programming Language: Design, Semantics, Implementation". *Journal of Science Of Computer Programming,* Vol. 19, Num. 2, pp. 87-152. 1992.

[Brau95] T. Braun and C. Diot. "Protocol Implementation using ILP", SIGCOMM '95, Boston, August 1995.

[Cas96] Claude Castelluccia, Walid Dabbous, Sean O'Malley. "Generating Efficient Protocol Code from an Abstract Specification". In *Proceedings of ACM SIGCOMM'96*, Stanford University, California, August 1996.

[Cas94a] Claude Castelluccia and Walid Dabbous. "Modular Communication Subsystem Implementation using a Synchronous approach", In *Proceedings of USENIX-94, Symposium on High Speed Networking*, Oakland, CA, August 1994.

[Cher86] D. R. Cheriton, "VMTP: a transport protocol for the next generation of communication systems", In *Proceedings ACM SIGCOMM '86,* Stowe, Vermont, August 1986, pp. 406-415.

[Ches89] G. Chesson, "XTP/PE Design Considerations", In *Protocols for High-Speed Networks,* H. Rudin, R. Williamson, Eds., Elsevier Science Publishers/North-Holland, May 1989.

[Cla94] David Clark. "The Art and Engineering of Protocol Performance", SIGCOMM '94 Tutorial, London, Août 1994, transparent 123, page 62.

[Cla90] David D. Clark and David L. Tennenhouse. "Architectural Considerations for a New Generation of Protocols", In *Proceedings ACM SIGCOMM '90,* September 24-27, 1990, Philadelphia, Pennsylvania, pp. 200-208.

[Cla89] David D. Clark, Van Jacobson, John Romkey, Howard Salwen. "An analysis of TCP processing overhead", *IEEE Communications Magazine*, June 1989, pp. 23-29.

[Cla87a] David Clark, Mark Lambert, Lixia Zhang. NETBLT: A Bulk Data Transfer Protocol. Network Information Center, *RFC-998*, SRI International, March, 1987.

[Cla87b] D. Clark, M. Lambert, L. Zhang. "NETBLT: a high throughput transport protocol", *CCR*, Volume 17, Number 5, 1987, pp. 353-359.

[Cla82] David D. Clark. Window and Acknowledgment Strategy in TCP. *RFC-813,* July 1982.

[Col85] R. Colella, R. Aronoff, K. Mills. "Performance Improvements for ISO Transport", *Computer Communication Review,* Vol. 15, No. 5, September 1985.

[Dab94a] Walid S. Dabbous. "High performance presentation and transport mechanisms for integrated communication subsystems", In *Proceedings of the 4th Internatinal IFIP Workshop on Protocols for High Speed Networks*, Vancouver, August 1994.

[Dab93] Walid Dabbous, Christian Huitema. *"XTP implementation under Unix"*, Research Report No 2102, Institut National de Recherche en Informatique et en Automatique, November 1993.

[Dab92] Walid Dabbous et al. *"Applicability of the session and the presentation layers for the support of high speed applications"*, Technical Report No 144, Institut National de Recherche en Informatique et en Automatique, October 1992.

[Diot95] C. Diot, I. Chrisment, A. Richards. "Application Level Framing and Automated Implementation", 6th IFIP International Conference on High Performance networking, Palma (Spain), September 1995.

[Gag96] F. Gagnon and C. Diot. "Impact of Out-of-Sequence Processing on Data Transmission Performance". INRIA Sophia Antipolis, July 1996.

[Gun91] P. Gunningberg, C. Partridge, T. Sirotkin, B. Victor. "Delayed evaluation of gigabit protocols", In *Proceedings of the* 2^{nd} *MultiG Workshop*, 1991.

[Hogl94] Anna Hoglander. *Experimental evaluation of TCP in user space.* INRIA internal report, request from `cdiot@sophia.inria.fr`.

[Jac90] V. Jacobson, R. Braden and L. Zhang. "TCP Extension for High-Speed Paths", *RFC 1185*, October 1990.

[Jac88a] Van Jacobson. Congestion avoidance and control. *Computer Communication Review,* Volume 18, No. 4, 1988.

[Jac88b] V. Jacobson, R. Braden. TCP Extensions for Long-Delay Paths. *RFC-1072,* October 1988.

[Kan88] Hemant Kanakia, David R. Cheriton. "The VMP Network Adapter Board: High-Performance Network Communication for Multiprocessors", In *Proceedings SIGCOMM '88,* Stanford, CA, 1988, pp. 175-187.

[Lam87] Mark Lambert. On Testing the NETBLT Protocol over Divers Networks. Network Information Center, *RFC-1030*, SRI International, November, 1987.

[O'M90] S. W. O'Malley and L. L. Peterson. "A Highly Layered Architecture for High-Speed Networks", In *Proceedings of the IFIP Workshop on Protocols for high speed networks II,* Palo Alto, CA, 1990, pp. 141-156.

[Par93b] C. Partridge and S. Pink. A Faster UDP. Submitted to *IEEE Transaction on Networking.*

[PEI92] *"XTP Protocol Definition, Revision 3.6",* PEI 92-10, January 1992, Protocol Engines Incorporated, Santa Barbara, CA 93101, USA.

[Peh92] Bjorn Pehrson, Per Gunningberg and Stephen Pink "Distributed Multimedia Applications on Gigabit Networks", *IEEE Network Magazine*, Vol 6, No 1, January 1992, pp. 26-35.

[Rec84b] Recommendation X.409, "Message handling systems: Presentation transfer syntax and notation", *Red Book, Volume VIII, Fascicule VIII.7, ITU, October 1984, pp. 62-93.*

[Ten96] D. L. Tennenhouse and D. J. Wetherall. "Towards an Active Network Architecture", Computer Communication Review, Vol. 26, No. 2, April 1996.

[Wak92] Ian Wakeman, Jon Crowcroft, Zheng Wang, and Dejan Sirovica, "Layering considered harmful", *IEEE Network*, January 1992, p. 7-16.

[Wat87] Richard W. Watson, Sandy A. Mamrak. "Gaining efficiency in transport services by appropriate design and implementation choices", *ACM transactions on computer systems,* Vol 5, No. 2, May 1987, pp 97-120.