A Communication Architecture for High-speed Networking

Zygmunt Haas
AT&T Bell Laboratories
Room 4F-501
Holmdel, NJ 07733

Abstract

The communication speed in wide-, metropolitan-, and local-area networking has increased over the past decade from Kbps lines to Gbps lines; i.e., six orders of magnitude, while the processing speed of commercial CPUs that can be employed as communications processor has changed only two to three orders of magnitude. This discrepancy in speed translates to "bottlenecks" in the communications process, because the software that supports some of the high-level functionality of the communication process is now several order of magnitude slower than the transmission media. Moreover, the overhead introduced by the operating system (OS) on the communication process strongly affects the application-to-application communication performance. As a result, new proposals for interface architecture have emerged that considerably reduce the overhead associated with the OS. With the alleviation of the OS bottleneck, the performance required from the communication system will be so high that a new protocol architecture that will support high-performance (high-throughput and low-delay) communication is needed. The purpose of this paper is to propose an alternative structure to the existing models of communication architecture. The architecture presented employs an approach quite different than that used in the existing layered models; i.e., the proposed architecture has a *horizontal* (parallel) structure, as opposed to the *vertical* (serial) structure of the layered models. The horizontal structure, coupled with small number of layers, results in elimination of unnecessary replication of functions, unnecessary processing of functions, and lowers the unnecessary interprocess communication burden of the protocol execution. (For example, unnecessary error detection for voice packets in integrated applications and unnecessary error detection in multiple layers for data applications are eliminated, the overhead associated with multi peer-to-peer connection of multi-layer architecture is substantially reduced, etc.) Furthermore, the proposed architecture lends itself more naturally toward full parallel implementation. Parallelism offers the potential of increased protocol processing rates, reduction in processing latency, and further reduction in processing overhead. And with the advances in VLSI, full parallel implementation becomes more and more feasible. Some additional features of the proposed scheme include ***selective functionality,*** which permits the protocol to easily adjust itself to the kind of performance demanded by the network-application requirements. The architecture presented supports the notion of communication networks that resemble an extension of a computer bus, and are capable of providing very high bandwidth, on the order of Gbps, directly to the user.

1 Introduction and Motivation

The communication speed in wide-, metropolitan-, and local-area networking has increased over the past decade from Kbps lines to Gbps lines; i.e., six orders of magnitude. The processing speed of commercial CPUs that can be employed as communications processors has changed only two to three orders of magnitude. This discrepancy in speed translates to "bottlenecks" in the communications process, because the software that supports some of the high-level functionality of the communication process is now several orders of magnitude slower than the transmission media. In other words, when the lines operated at 9.6Kbps, the multi-layer conventional protocols implemented in software were fast enough to match the speed of the lines. However, now, when the lines operate at 1.7Gbps, the mismatch in speed is so large that the advantage of high-speed lines is buried in the large processing overhead of high-level protocols, leading to large delay and low throughput. This change has shifted the bottleneck from the transmission to the (software) processing at the end points.

Another trend that has influenced the design of communication networks is the potential of hardware implementation (i.e., VLSI). It is much easier and cheaper today to implement large and fast communications hardware on a single silicon chip. However, this opportunity cannot be fully exploited with current protocols that were developed for software implementation. The purpose of this work is to investigate the various possibilities of improving the performance of communications protocols and interfaces, so that the slow-software-fast-transmission bottleneck can be alleviated, and to propose a new protocol architecture that is suitable for future, high-performance communication.

A basic question is whether or not the current (software-based) protocols can provide the required performance? As suggested in [1], TCP/IP ([2]) can, with some minor changes, provide up to few hundred Mbps throughput. Assuming that the machines will, indeed, require such a high bandwidth, there are still other major communication bottlenecks that prevent delivery of such large throughput. Thus, [1] concludes that it is the other bottlenecks that we're supposed to be concerned with, not the high-layer protocols. Furthermore, there is still the question of what applications require such a large throughput. (The problem is not only in identifying such applications, but also in predicting future application.)

Let us first provide an answer to the last question. The general trend of computing environments to move toward distributed and parallel processing systems is, and will be, strongly responsible for the demand for increased performance. For example, a parallel processing system implemented on the fine-grain level requires delays on the order of μseconds. In distributed processing system, large files may be required to be transferred between machines with very low latency. Thus, for the parallel and distributed processing environment, both the very low delay and large throughput are crucial. (Another bandwidth demanding application that can be identified today is video. Video applications will become more and more important with traffic integration in packet-switched networks.)

Unfortunately, indeed, the impact of the overhead introduced by the operating systems (OS) on the communication process strongly affects the application-to-application communication performance. The major sources of this overhead are (see also [3,4,5,6]):

- scheduling
- multiple data transfers from/to the user[1]
- overhead of entities management:
  - timers
  - buffers
  - connection states
- overhead associated with division of the protocol processing into processes (including interprocess communication)
- interrupts
- context switching

The reason for large OS overhead is the structure of the communication process in general, and the implementation of network interfaces, in particular. In other words, the CPU performs such an important role in the communication process, that, as a consequence, there are too many interrupts, too many context switches, and too large a scheduling overhead. Network interfaces were invented to offload the CPU from the communication process. Unfortunately, they do only a partial job; interfaces are still being built that interrupt the processor for each received packet, leading to multiple context switches and scheduler invocations. (Another solution is to structure the process in a different way; to eliminate the scheduler calls by the interrupts, and to resolve the scheduling of the process that completed the communication at the "regular" scheduler invocations.) Some new proposals for interface architecture ([4]) considerably reduce the overhead associated with the OS. These proposals structure the communication process much in the same way that Direct Memory Access (DMA) is implemented; the CPU initiates a communication event, but has little to do in the actual information exchange process. The considerable reduction in the OS overhead that results from the new structuring of the communication process, will probably have a crucial impact on the feasibility of providing multi-Mbps bandwidth directly to the user (see also Section 6 for our ideas on a new interface structure).

The communication goal in the parallel and distributed system environment is to provide communication, whose throughput is restricted only by the source capacity of the transmitter or the sink ability of the receiver. Also, the communication delay should be minimized. As stated, OS are today the bottleneck of the communication process. However, once the OS bottleneck are resolved, the performance required from the communication systems will be so high that a new approach will be needed to the architecture of communication protocols and to the network interface design to support high-performance (high-throughput and low-delay). This argumentation answers the basic question whether the current protocols are adequate for future high-speed networks; local improvements in the protocol design might be adequate for current OS-limited systems. This will not be the case for future systems.

Along these lines, we propose an architecture that is an alternative to the existing layered architectures. The novel feature of the proposed architecture is the reduction in the **vertical layering**; services that correspond to the definitions of layers 4 to 6 in the ISO/OSI RM[2] are com-

[1] It has been already emphasized several times in the literature (for example, [3,4]), that the number of data copies between buffers (i.e., the per octet overhead) has a crucial effect on the performance of a protocol.

[2] The references to the ISO/OSI Reference Model in this paper relate only to the definition of services in the various layers of the model, and not to the actual architecture of the ISO/OSI protocols.

bined into a single layer that is **horizontally** structured. This approach lends itself more naturally to parallel implementation. Moreover, the delay of a set of processes implemented in parallel is determined by the delay of the longest process, and not by the sum of all the process delays, as is the case in sequential implementation. In the same way, the total throughput need not to be limited by the lowest capacity process, but can be increased by concurrently performing the function on several devices. Thus a protocol structure that lends itself to parallel implementation has the potential to provide the high performance matched to the requirements of the new generation of improved OS.

2 The Challenge

The challenge is to propose a single protocol that successfully provides communication over diverse networks: data rates ranging from Kbps to Gbps and network diameters from LANs to WANs. Also, the diverse requirements of many different applications need to be supported: connection-oriented and connectionless service, stream-like traffic and bursty traffic, reliable transport and best-effort delivery (error-control and flow-control), different data sizes, different delay and throughput requirements, etc. The required throughput is on the order of hundreds of Mbps application-to-application (with a tendency toward Gbps), and the delay is on the order of hundreds of microseconds[3].

Let us discuss the above a little bit further. Assuming a very reliable subnet, the error-recovery procedures can be simplified so that when there are no errors very little overhead is incurred. However, this low overhead comes at the expense of having a large penalty in the event of an error. (On the other hand, in networks with a high bit error rate (BER), both of the error and no-error cases the recovery procedure should be minimized.) This is what *success-oriented*[4] protocols mean: minimize the overhead for successful delivery cases at the expense of larger penalty for unsuccessful delivery attempts.

The reason for insisting on optimizing the performance[5] over such an extensive range of the data-rate (Kbps to Gbps) is that in the future we expect to have an enormous variety of networks. In other words, one cannot expect that, with the introduction of multi-megabit-per-second networks, Ethernets will (at least immediately) disappear. Even today, the range of communication is very large; 300 bps modems are still being used.

Thus, in order to optimize the performance of the protocol for all the diverse networks/applications, the protocol must consist of a set of versatile protocols, that can be readily switched between. This is what we refer to in this work as ***selective functionality protocols***.

The work is organized in the following way: The next Section outlines possible approaches to improving protocol performance. Section 4 presents our horizontal protocol approach, while the ***selective-functionality*** feature is described in Section 5. A new approach to network interfaces is discussed briefly in Section 6. Section 7 shows a basic design example of the horizontally structured architecture. Section 8 concludes the work.

3 The Possible Approaches.

There are several possible solutions to overcome the slow-software-fast-transmission problem:

1. Improve the performance of current high-layer protocols implementations([1,7])
2. Design new high-layer protocols based on the current philosophy; i.e., software-based, layered structure ([8,9,5])
3. Hardware implementation of high-layer protocols ([10])
4. Reduction of high-layer protocols overhead (i.e., "lite" protocols)[6]
5. New design philosophy that differently structures the high-layer protocols ([6])
6. Migration of high-layer functionality to lower layers, in particular to the physical layer (for example, networks that perform some high-layer functions by trading the physical bandwidth; i.e., end-to-end flow control done at the physical layer in *Blazenet* [11]).

The approaches were arranged according to how much they diverge from the conventional protocol design philosophy. Of course, there can be any combination of the above five approaches.

[3]Of course, because of the propagation delay, such a low delay requirement has little advantage for a WAN, and is necessary only in LAN/WAN environment. However, because of the requirement that the protocol design be independent of the actual subnet being used, the stringent performance requirements need to apply to any communication.

[4]This term refers to protocols that exploit the characteristics of reliable networks. Such networks, typically composed of fiber links and digital switches, have a very low error rate, deliver packets in order, and rarely drop packets.

[5]Here "performance" refers to: throughput, delay, and transmission efficiency.

[6]We note that a "lite" (i.e., lightweight) protocol is created by adjusting the number of states and the amount of control information that is passed between the protocol states, in such a way, on one hand, to maximize the protocol functionality (to reduce the overhead of the software-based application layer) and, on the other hand, to minimize the overhead caused by the low-level implementation of the protocol states (for example, number of chips for hardware implementation or the number of page faults for software implementation).

Note that there exists yet another approach: Reduction of lower-layer functionality. (For example, [12] proposes to solve the link-by-link flow control problem by dropping excessive packets and correcting the packet loss at the higher-level (transport) through retransmissions.) We consider this approach, which is the opposite of the fifth approach, to be, in general, unable to solve the high-speed communication problem, since pushing the problems to higher layers introduces, in fact, larger delays (the implementation of higher layers is typically software-based, and thus slower). Also, in this specific case of the flow control example, the long timeout associated with detecting the dropped packets (which is done on the end-to-end basis), increases the overall delay. (We note, however, that pushing some of the functionality to higher layers simplifies the network design, since at the lower layers there is more traffic aggregation, and the total number of packets-per-second is larger, leading to more complex processing within the network. Thus, in some cases this approach may be appropriate.)

Every one of the six approaches outlined can potentially improve the protocol performance. However, major improvement in performance can be obtained by combining a few of the above approaches. For instance, even though changes in the current version of TCP/IP that reduce overhead might increase the protocol throughput to several hundred Mbps, hardware implementation of the improved version will signify the improvement even further. Consequently, we believe that the "correct" answer to the question of how to implement high-speed protocols, is an intelligent integration of several approaches.

In the next section, we consider a combination of the third, fourth, fifth, and (in a limited sense) the sixth approaches. The proposed architecture is based on three layers: the ***N**etwork **A**ccess **C**ontrol* (the NAC layer), the ***C**ommunication **I**nterface* (the CI layer), and the ***A**pplication* (the A layer). The NAC layer consists of all the services defined in and below the Network layer of the ISO/OSI RM. The CI layer consist of the services defined by the Transport, Session, and Presentation layers. The A layer corresponds to the conventional Application layer. (The reason for the proposed three-layer structure is that the services of the NAC are mostly hardware-based, while the services of the A layer are software-implemented. Thus the CI represents the boundary between the software and the hardware. Thus, in our view, the model of the communication protocol architecture can consist of the three layers, where the NAC is hardware-based, the A is software, and the CI is a mixture of software and hardware. It is our belief that to achieve high-speed communication, the structure of CI must lend itself easily (little overhead) toward parallel hardware implementation, and, as shown in the nest section, the proposed three-layer structure provides a basis for such a parallel implementation.

4 Horizontally oriented protocol for high-speed communication

We propose here an alternative structure to the existing communication architectures. The central observation is that protocols based on extensively-layered architectures possess an inherent disadvantage for high-speed communication. While the layering is beneficial for educational purposes, strict adherence to layering in implementation decreases the throughput and increases the communication delay. There are several reasons for this reduction in performance, among them (see also [6]) the replication of functions in different layers, performance of unnecessary functions, overhead of control messages, and the inability to parallelize protocol processing.

The architecture presented here employs an approach quite different than that used in the extensively-layered models; i.e., the proposed architecture has a **horizontal** structure, as opposed to the **vertical** structure of multi-layered architectures. We refer to our architecture as **H**orizontally **O**riented **P**rotocol **S**tructure, or **HOPS**.

The main idea behind HOPS is the division of the protocol into functions, instead of layers. The functions, in general, are mutually independent, in the sense that the execution of one function can be performed without knowing the results of the execution of another. (Thus intercommunication between the functions is substantially reduced.) For example, flow-control and decryption are independent functions. If the dependency between two (or more) functions is such that the execution of one depends on the result of another, the function can still be conditionally executed. For example, packet resequencing is to be executed only if error-control detects no errors. Thus, resequencing can be conditionally executed in parallel with error-control, and at the end a (binary) decision made whether to accept or ignore the resequencing results.

Because of the independence between the functions they can be executed in parallel, thus reducing the latency of the protocol and improving throughput.

Figure 1 shows the structure of HOPS. In this Figure, the correspondence in services between HOPS and the ISO/OSI RM is also shown. Thus the CI of HOPS implements in hardware the services defined by layers 4 to 6. The meaning of hardware implementation is not (necessarily) that HOPS is fully cast into silicon, but that specific hardware exists to perform the functions, rather than relying on the host software. Thus HOPS can be implemented as a collection of custom-designed hardware and general-purpose processors.

CI receives the raw information (unprocessed packets) from the *Network Access Control* (NAC) layer. The central layer in HOPS is the *Communication Interface*

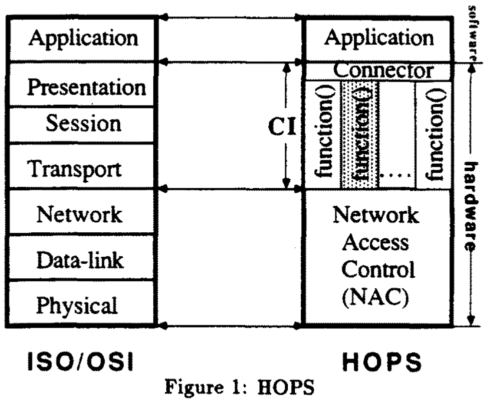

Figure 1: HOPS

(CI). CI is divided into parallel (and independent or conditionally-independent) functions. Before the results of the functions are passed to the *Application* layer, they are evaluated in the *Connector*. The *Connector* executes the conditional dependency among the functions, and passes the processed information to the *Application* layer.

HOPS is expected to lead to high-performance implementations because of several reasons. First, because of the horizontal structure of functions, the delay of a packet processing is determined by the slowest function, rather than by the sum of all delays. This is achieved in HOPS by the independency characteristic of functions. Thus a function need not to wait for a result of another function before its execution can begin. Second, the throughput can be easily improved by increasing the number of units of capacity-limited functions. Such increase is rather natural in paralleled-structured CI. Third, because of the compression of layers, much of the replication and overhead are eliminated (such as buffering on different layers, for example). Fourth, the horizontal structure lends itself to parallel implementation on separate (possibly customized) hardware. The parallel implementation by itself has the potential of lowering the processing delay and increasing the processing throughput. Also, the overhead associated with switching between the execution of functions is grossly eliminated, as is the communication messages between the processes. Finally, the *selective functionality* feature[7] can eliminate unnecessary function processing.

We also believe that in order to achieve the limit of performance, one should implement the HOPS-structured protocols in custom designed hardware.

It should be noted that HOPS, as well as other solutions

[7] See Section 5

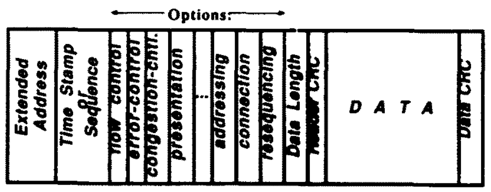

Figure 2: HOPS packet format

based on the fourth, fifth, and sixth approaches described in Section 3, are not compatible with the ISO/OSI model, and, in fact, violate the model boundaries.

5 HOPS as a Selective Functionality Protocol

Because HOPS is intended to support communication over diverse networks and for diverse applications, a single protocol cannot provide optimum performance. For example, retransmission policy depends on the quality of the network: selective retransmission is better for networks with large average BER, while go-back-n may be beneficial in very reliable environment. Moreover, the requirements for a protocol may change with time and space; for example increasing congestion may change retransmission policy, or the required error-control mechanism may differ from subnet to subnet. Consequently, what we propose is a "protocol with a *menu*," whereby a user will request some combination of functions that are needed to achieve some particular level of performance. For instance, the function *retransmission* may be designed to receive the following values: *selective, go-back-n, go-and-wait, none, any*, ... The network interface (NI) has to decide on the particular retransmission policy required. If the NI has some knowledge about the subnets the packet is going to travel on, then the NI can make an intelligent decision on the required policy. The NI can change its decision with time, if it learns that the conditions have changed or that its previous decision was incorrect.

The function values are communicated throughout the network by means of the special option field in the packet format as shown in Figure 2. The values are decoded separately, in parallel, and "on the fly" at the packet arrival time.

There is another, very important advantage of the selective functionality feature; possible compatibility with current protocols. It cannot be expected that there will be immediate transfer from the rooted architectures and protocols. Thus protocols like TP4/CLNP will continue to exist, and it is of paramount importance that the current and the new high-speed oriented protocols interwork.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase®
Smarter legal research.