7C－6

Improving End-to-End Throughput for FileTransfer Using Combined Layers

Onur ALTINTAS, Hitoshi AIDA, Tadao SAITO
The University of Tokyo

1 Introduction

Progress in the field of high speed networking made Gb/s transfer rates available which in turn, has lead to a debate in the research community on the suitability of existing protocols such as TCP/IP or the ISO/OSI suite over high speed networks [1]. The problem is that when computers are connected to high capacity links, the real bandwidth accessible by the system is choked by the bottlenecks of the network adapters, the ability of packet processing per second, and the overall hardware and software system structure. The present protocol architectures seem sufficient for today's networking requirements. However, demands of future networks require some novelty in the approach to current structures. In the rest of this paper, we show some limitations of the classiccal layering approach in comparison with the capacity improvement that can be available when things are handled according to the needs of the applications.

2 Layering versus Performance

Layering, as an architectural abstraction, is valuable in clarifying the roles of various protocols, but experience suggests that layered implementations are not perfect from the point of performance. At the lowest level, computer communication networks provide unreliable packet delivery. At the highest level, application programs often need to send large volumes of data. Network researchers have found general purpose solutions to the problems of reliable delivery , making it possible for experts to build a single instance of protocol software that all application programs use. Having a single general purpose protocol helps isolate application programs (usually too much) from the details of networking, and makes it possible to define a uniform interface.

However, in most of the current protocol designs, it is difficult to keep the application layer running in real-time due to data loss or reordering which prevents immediate processing. Today's applications do not deal with packet loss or reordering. Instead, lower layer protocols such as TCP shoulder the burden of following the data, and requesting retransmissions of the lost ones. With TCP, presentation conversions (if any, since some protocols simply avoid this) can occur only after data is reordered and recovered meaning that a lost packet stops the application.

3 Measurements on TCP/IP

In this section some experimental performance evaluation results of bulk data transfers over TCP/IP in the SunOS implementation of Interprocess Communication (IPC) using BSD sockets will be presented.

The performance measurements were conducted on a single Sun Sparc Station 10 with SunOS Release 4.1.3. All measurements were performed with a single user on the machine in order to prevent interference. The only load on the machines were the various system "housekeeping" daemons consuming little processing power and generating some background traffic occasionally causing insignificant fluctuations in the system behavior.

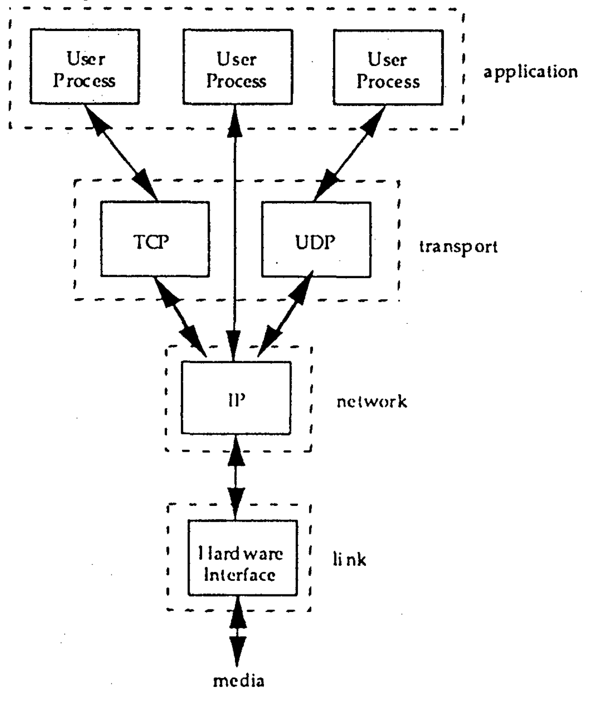

Figure 1. TCP/IP layering.

The test programs using the UNIX socket facilities to access TCP/IP transferred 4 MBytes of data from the server process to the client's user memory space.

Throughput as the Performance Measure

The main point of interest of performance in bulk data transfers is the throughput of the transmission under a certain set of system parameters. In our experiments, the throughput measurements are perfromed for two communicating processes running on a single Sparc Station 10 communicating through the software loopback. That is, instead of injecting packets into the Ethernet, they are simply forced to loop back to the receiving process. It is of no significance to the protocol whether packets traverse the Ethernet coming from a remote process or without any Ethernet journey delivered from another local process.

With end systems that are fast enough, even under TCP, transfer rates close to the theoretical limit of the Ethernet speed (1,155,063 bytes/sec) can be attained. Any significant performance improvements beyond the TCP rates are hard to observe under Ethernet speeds. This is one reason we used the software loopback to measure transfer rates. Throughput was calculated as the average delivery rate in bits per second.

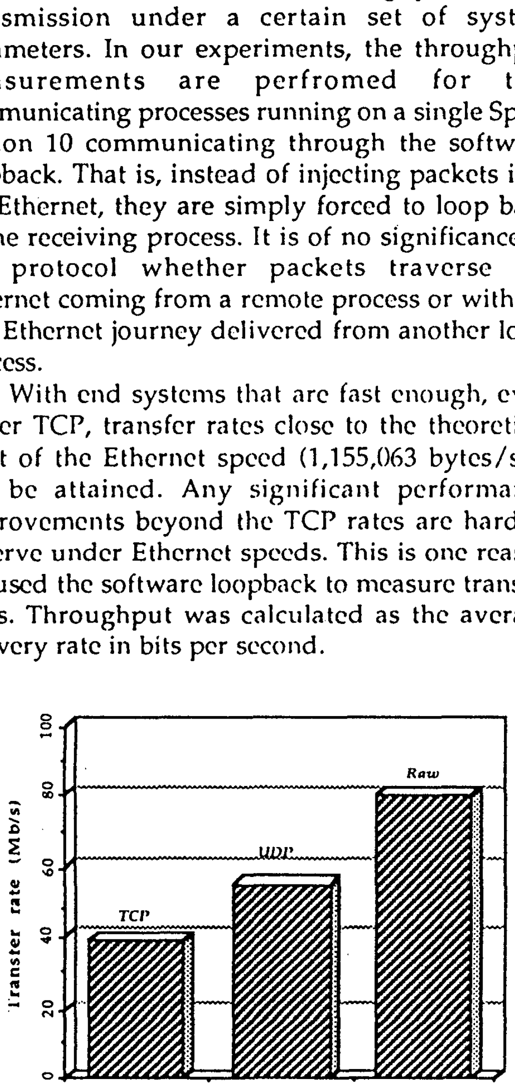

Figure 2. Comparison of data rates.

First, consider Figure 1. The three sets of measurements in Figure 2 are taken first, for two processes communicating through the reliable stream transfer protocol TCP; second, for the unreliable datagram service UDP; and third, for the direct access to the IP protocol (which is again unreliable in nature). Direct access to the IP layer is possible by the use of raw sockets. That is, the data accessed is "raw" indeed with only the IP and Ethernet headers. Also, note that the results for TCP shown in Figure 2 are the transfer rate values which are tuned precisely by changing socket buffer sizes of both processes.

4 Discussion

The raw capacity limited by the transport layer is huge enough to make one reconsider the pros and cons of layering. However, by saying this, we do not claim that the transport layer should be abolished completely. Transport layer might again have the responsibility of doing flow and congestion control, while the application layer doing error control according to its needs.

It is desirable to allow some manipulations to be performed even when there is misordering or loss. Taking action against data loss is dependent upon the needs of the application. With the classic approach of the transport protocol, delivery will be suspended. Another option is the application layer accepting imperfect data which works for real-time delivery of video and voice. Still another option is to have the application deal with lost data. In file transfers, for instance, the receiver can copy the data into the correct position (if some form of identifying the data units that application specifies is done) or simply exploits the random access nature of disks, without waiting for everything, in both cases.

In short, we believe that, although layering is a useful approach; with current architectures, too much emphasis is put on transparency, generalization and isolation and that with relatively smaller error rates of future networks, strict check for every application may become unnecessary. The ultimate way of handling things in a networking application had better depend on the context of the application data rather than providing the same sort of services to every application. At the time of this writing, protocol implementations considering the ideas explained above are under investigation.

References

[1] A.N. Tantawy, ed., *High Performance Networks: Frontiers and Experience,* Kluwer Academic Publishers, Massachusetts, 1994.

[2] D.D. Clark and D.L. Tennenhouse, "Architectural Considerations for a New Generation of Protocols", *Proc. ACM SIGCOMM '90,* Sept. 1990, pp. 200-208.

[3] J. Crowcroft, J. Wakeman, Z. Wang and D. Sirovica, "Is Layering Harmful?", *IEEE Network Magazine,* Jan. 1992, pp. 20-24.

[4] G. Conti, "RAP: A High Speed Protocol for Random Access Devices", *Int. Conf. on Communications, ICC '92,* June 1992, pp. 969-975.