Raj Jain
William R. Hawe

Raj Jain is now at
Washington University in Saint Louis
Jain@cse.wustl.edu
http://www.cse.wustl.edu/~jain/

Performance Analysis and Modeling of Digital's Networking Architecture

Digital has some of the highest performing networking products in the industry today. Transfer rates of 3.2 megabits per second and higher have been measured on an Ethernet. These high speeds result from careful performance analyses and planning at all stages of the development cycle. A set of case studies illustrates these analyses. These studies include performance modeling for adapter placement in the physical layer; buffering in the data link layer; path splitting in the network layer; cross-channel piggybacking, timeout and congestion algorithms in the transport layer; and file transfer and terminal communications in the application layer. Completing the paper are studies on network traffic measurements and workload characterization.

Performance analysis is an integral part of the architectural design and implementation of networks at Digital Equipment Corporation. This deliberate strategy has helped to make us the industry leader in networking products. Some of these products have the highest performance available today. Task-to-task transfer rates of more than 3.2 megabits (Mb) per second have been measured on an Ethernet local area network (LAN) connecting two MicroVAX II systems.[1]

This paper describes a number of case studies that illustrate the analyses done to improve the performance of Digital's network products. These analyses are ongoing; they are planned for every stage in the life cycles of products. The design life cycle of a product consists of the following stages: conceptualization, prototyping, marketing research, development, sale, and field support. Each stage takes place in a different organization within Digital. A research organization usually conceives an idea for a new product. An advanced development team then develops the architectural specification and builds a prototype to demonstrate the feasibility of the idea. In turn, the marketing organization decides if the product can be sold and how competitive it will be. If they decide that the idea should become a product, the development organization will perform that task.

Each of those organizations has a team of performance analysts who ensure that the best alternatives are chosen at each stage. The sales organization also measures product performance and develops capacity planning and performance-tuning tools. The field support organizations monitor performance at customer sites and feed the information back to the development organizations. They then improve the product through revisions, field changes, and updated models.

To conduct performance studies, we use analytical modeling, simulation, and the taking of appropriate measurements. Which of those techniques to use depends upon the product development stage and the time available to do the study. Queuing theory, including operational analysis, is extensively used in analytical modeling.[2,3] Simulation models are usually developed to solve specific problems[4,5,6] or often they are solved via queuing network solvers. Measurements of operational characteristics are taken of the system as well as the workload, using both software and hardware monitors. Traffic measurements are taken on Digital's own networks, as well as on those at customers' sites.[1,7] Tools for capacity planning, monitoring, and modeling are also used by the teams doing these performance analyses.[8,9] Sometimes we have to

develop new performance metrics[10] or statistical computation algorithms.[11]

This paper presents the diversity of the performance analysis techniques used to ensure that our networking products operate at high efficiencies. Many performance studies of our products have been published; we do not intend to reproduce them here. We have selected a representative group of unpublished case studies to illustrate the diversity of our approach to performance improvement. One typical problem from each of the key layers of the networking architecture will be discussed. A discussion of workload characterization and traffic analysis will close the paper.

Physical Layer Performance

We conducted many performance studies within Digital to help set the parameters of the 10 Mb-per-second Ethernet LAN. This is the same Ethernet that, with certain modifications, we proposed for standardization and was later adopted as the IEEE 802.3 standard. The two most interesting problems in the physical layer design are clock synchronization (phase lock loop versus counter) and the placement of adapters on the Ethernet cable. We describe the latter problem and the proposed solution below.

At each adapter, some fraction of the incoming signal is reflected back along the cable. If adapters are placed in close proximity, their reflections may reinforce each other and interfere with the signal.

The adapter designers had specified that a total noise level of 25 percent of the true signal level was an acceptable limit. Since half of this noise normally comes from other noise sources (sparks, radiation, etc.), the reflected voltage must be less than 12.5 percent of the signal level.

The cumulative reflection is actually strongest at the transmitter itself because of the attenuation of the signal and its reflection as they propagate through the cable. Since the transmitter is not adversely affected by the reflection, however, adapters placed next to the transmitter are the most sensitive to reflection problems. Therefore, those adapters were the best candidates for analyzing problems caused by reflections.

It is essential to maintain some minimum separation between adapters. To assist network installers, the Ethernet cable is marked at 2.5-meter intervals; the specifications state that the adapters should be placed only at those marks. That spacing was determined from a model that simulated many different random placements of a given number of adapters on the cable and determined the worst-case reflection. The simulation model showed that the worst case occurs when approximately 100 adapters are placed on the marked cable. With 100 nodes, the reflected voltage exceeded 10 percent of the true signal in only 24 of the 10,000 configurations that were simulated. In fact, the maximum reflection observed for any placement was 12.1 percent, well below the 25 percent noise allowance.

It is easy to see why the 100-adapters case performs worse than other cases with both more and fewer adapters. With the cable marked at 2.5-meter intervals, a single Ethernet segment (500 meters) can accommodate up to 200 adapters. When the number of adapters is small, their reflections will be too small to cause any problem. On the other hand, if the number of adapters is close to the maximum of 200, the reflections from neighboring adapters will tend to cancel each other out.

The cable marking alone is no guarantee against experiencing reflection problems. Given this or any other marking guideline, it is still possible to position adapters so that the reflections reinforce. This happens if the adapters are placed $\lambda/2$ apart, λ being the wavelength at which transmissions are taking place. For example, for a 10-MHz signal traveling at a speed of 234 meters per microsecond, λ is 23.4 meters, the speed divided by the frequency. Hence, if the adapters are placed approximately 11.7 meters apart, their reflections will reinforce.

Data Link Layer Performance

A number of our studies about the performance of the data link layer in Ethernet have already been published.[12,13] M. Marathe compared five back-off algorithms and concluded that none was significantly better than the binary exponential back-off algorithm.[12] This simulation-based analysis also showed that the number of retries should be increased from the original 8 to 16.

Response times at the user level have also been studied. Such studies show that a 10 Mb-per-second Ethernet can support up to several thousand timesharing users.[14] A capacity planning tool was developed to study the system-level performance for any given configuration.[6] The performance

studies of extended LANs are discussed in this issue of the *Digital Technical Journal*.[15,16]

The case study described here uses a very simple analytical model to assist in designing an Ethernet adapter.

Three common approaches used for interfacing machines to a LAN are depicted in Figure 1. Case A represents the approach used in the Digital UNIBUS Ethernet Adapter (DEUNA) product. Case B represents the approach used in the Ethernet adapters made by 3COM Corporation for UNIBUS and Q-bus systems. Case C represents the approach used in adapters like the Digital Q-bus Ethernet Adapter (DEQNA) product. Each approach has certain advantages and disadvantages.

In Case A the packets received are first buffered on the adapter, then moved via direct memory access to buffers in the host's memory. Packets to be transmitted follow the same path, except in reverse. The throughput limit of the device is limited by how quickly it can move packets between the adapter and the host's buffers.

In Case B the packets received are also buffered on the adapter. In this case, however, the packet buffer memory is dual ported, with the host side being mapped into the address space of the host. This scheme allows the host to examine packets in the buffers on the adapter, without using any backplane bus bandwidth to receive them. However, to receive the packets, the host must copy them to buffers elsewhere in memory with a programmed move.

In Case C the packets flow in real-time into buffers in the host memory. The backplane is isolated from the channel with a first-in, first-out (FIFO) control point. This approach reduces the overhead on the host, as well as that on the adapter processors. In this case, however, excessive DMA-request latencies may cause overflow in the FIFO control; hence a packet may be lost when received. When packets are transmitted, these latencies may cause an underflow in the FIFO control; hence a packet may be aborted.

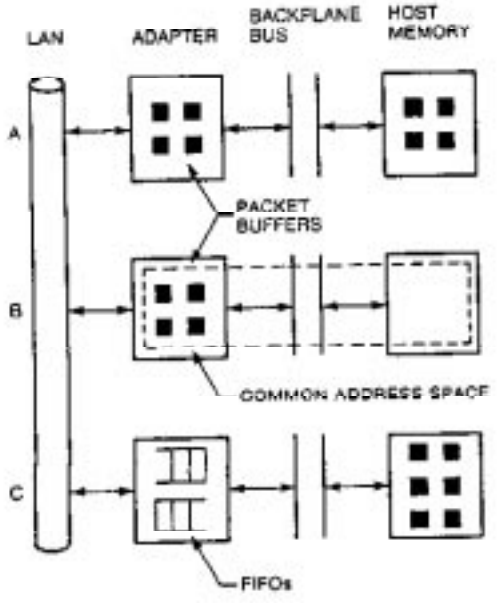


Figure 1 Three Ways of Organizing Buffers

The performance of the DEUNA adapter is sensitive to two factors: the number of receive buffers in the adapter, and the number of words to be transferred per UNIBUS capture. The number of receive buffers chosen affects the packet loss rate in the adapter. The number of words transferred affects the response to disks and other devices on the UNIBUS system. Transferring too many words per UNIBUS capture may cause a disk to experience "data lates," indicating that the disk could not get the bus for data transfer within the required time.

We wanted to know if the number of buffers chosen by the designers of the DEUNA adapter would cause these problems to occur.

We used a simple analytical model to determine the packet loss rate and a simulation model to determine the words per UNIBUS capture. The simulations showed that the DEUNA adapter should transfer only one word per UNIBUS capture. The analytical model is described here.

The packet-arrival process is assumed to follow a "bulk Poisson" distribution in which bursts of packets arrive at a rate of λ bursts per second. The number of packets per burst is assumed to be geometrically distributed with a mean B. The burst size is thus described by the formula

$$Pr(k \text{ packets in burst}) = (1-1/B) \times B^{(k-1)}, k \geq 1$$

For mathematical convenience we assume that all service times are exponentially distributed: this assumes that packet lengths are exponentially distributed, with a mean of L words per packet. The UNIBUS bandwidth is approximately 800,000 words per second. A fraction of that bandwidth, μ words per second, is available for

the transfers between the DEUNA buffer and memory. The rest of the bandwidth is used up by transfers between the disk and memory, and by other devices on the UNIBUS system.

If the DEUNA buffer has a capacity for not more than N packets, then any packet arriving at a full buffer will be lost. Let us first calculate the probability $P(n)$, $0 \leq n \leq N$, that there are n packets (including the one, if any, in service) queued at the DEUNA adapter. The distribution of the number of packets in the queue has a relatively simple form:

$$P(n) = (1 - \rho)/(1 - \rho \times \alpha^N) \qquad \text{for } n = 0$$

and

$$P(n) = P(0) \times (\rho/B) \times \alpha^{(n-1)} \qquad \text{for } 1 \leq n \leq N$$

in which

$$\rho = \lambda \times B \times L/\mu$$

and

$$\alpha = 1 - (1 - \rho)/B$$

If $B = 1$, the distribution of the arrival process reduces to an ordinary Poisson distribution, and $P(n)$ reduces to the classical solution of a space-limited M/M/1 queue.

The packet loss probability is

$$P(\text{loss}) = P(N) \times (B - 1 + \rho)/\rho$$

Here, $P(N)$ is the probability that the DEUNA adapter is totally full. Notice that the loss probability exceeds $P(N)$ because of burstiness.

Figure 2 shows the loss probability as a function of the number of buffers. This case assumes an arrival rate of 300 packets per second and a UNIBUS bandwidth of 22,000 words per second (40 percent of the UNIBUS width) available for transfers between the DEUNA adapter and memory. Curves for other arrival rates and available bandwidths can be similarly plotted. The curves show clearly that the designers' choice of 13 receive buffers will result in a loss rate of less than one percent, even with a UNIBUS system that is relatively heavily loaded.

Network Layer Performance

The concept of path splitting was introduced in Phase IV of the Digital Network Architecture (DNA). In earlier DNA versions, the routers maintained only one path to each destination even if several paths of equal cost existed. The following case study illustrates how simple analytical models were used to demonstrate that path splitting can significantly improve a network's performance.

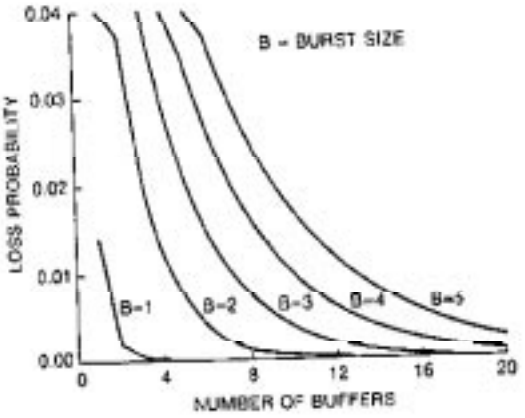


Figure 2 Loss Probability with Burst Traffic

Assume there are M packet sources in a network and that the ith source has a rate of $L \times \lambda_i$ packets per second, for $1 \leq i \leq M$ and some real constant L. (We increase or decrease all traffic by varying L). Assume further that there are N paths in the network and that the jth path has a speed of μ_j packets per second, for $1 \leq j \leq N$. The stochastic behavior of packet arrival and transmission is otherwise arbitrary. Assume that the set of paths usable by source i is $S_i \subseteq \{1, 2, \ldots, N\}$. Now we compare two strategies:

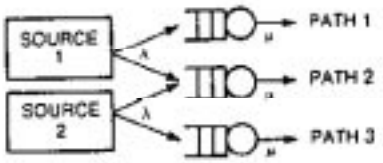
1. No Splitting — For each source i, select a path $j \in S_i$ with probability $P_{ij} > 0$. In this case all source i packets are sent on path j.
2. Equiprobable Splitting — For each packet from source i, select a path $j \in S_i$ with probability $1/|S_i|$. In this case the packet is sent on path j and successive paths are chosen independently.

For a large enough overall load factor L, the mean waiting time per packet under equiprobable splitting will be much less than it would be under no splitting. This can be proven by showing that, with no splitting, there exists a possible set of path assignments in which at least one path will saturate before any path saturates under

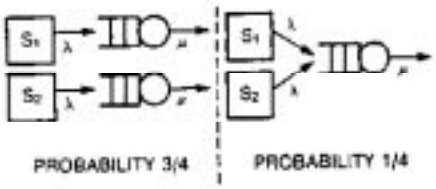
equiprobable splitting. Since the mean waiting time on a saturated path is infinite, the average waiting time of all sources over all paths will include an infinite term and therefore will also be infinite.

The performance impact of path splitting can be seen from the following example. Assume the simple configuration of two senders and three lines shown in Figure 3. Sender 1 has access to paths 1 and 2; sender 2 to paths 2 and 3. Each sender selects either accessible path with equal probability. Without path splitting, both sources might select the same path (path 2) with probability 1/4, or select separate paths with probability 3/4. The mean waiting time (assuming M/M/1 servers) is

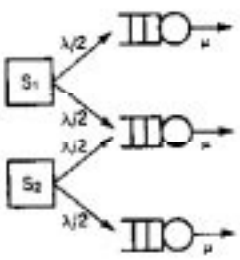
$$W_n = 3/4 \times (1/(\mu - \lambda)) + 1/4 \times (1/(\mu - 2\times\lambda))$$



(a) Two Sources and Three Paths

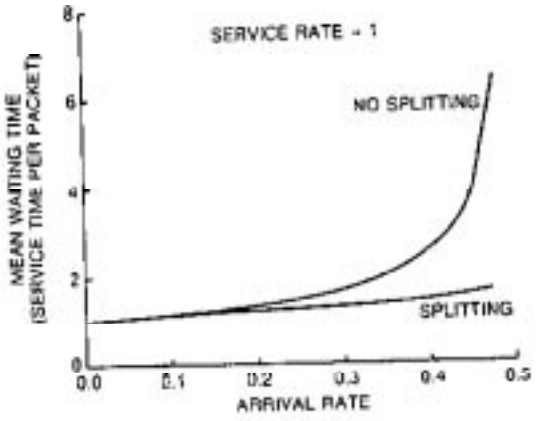


(b) No Splitting



(c) Equiprobable Splitting

Figure 3 Two Senders Transmitting on Three Lines



NOTE: ALL TIMES ARE NORMALIZED BY THE PACKET SERVICE TIME

Figure 4 Mean Waiting Time

With equiprobable path splitting, the input rate to paths 1 and 3 is $\lambda/2$ and the rate to path 2 is λ. The probability of a packet following path 1, 2, or 3 is 1/4, 1/2, and 1/4 respectively. The mean waiting time is

$$W_s = (1/4 + 1/4) \times (1/(\mu - \lambda/2)) + 1/2 \times (1/(\mu - \lambda))$$

The values of the mean waiting time both with and without splitting, W_s and W_n respectively, are illustrated in Figure 4. Observe that saturation occurs much earlier when there is no splitting.

Another advantage of path splitting is that it makes traffic less bursty. Bursty traffic presents a serious problem in performance control, both in average performance and in predictability. With bursts, the mean waiting time can greatly increase; in fact, if there is an average B packets per burst, then the mean waiting time will be about B times that value predicted by an identically loaded M/M/1 queue.[17] The waiting time variance will similarly increase, since the first and last packets in a burst will experience markedly different waiting times. The overall performance is very difficult to control or guarantee in such a situation.

Path splitting has a major advantage in this situation because it breaks up the bursts, sending each packet over a different path. The performance of a network with bursty traffic is thus appreciably improved. In fact, if there are more

paths usable by a source than there are packets per burst, then burstiness will have little effect on either the mean or the variance of waiting time. On the other hand, only two or three alternative equiprobable paths are enough to decrease the bursty-packet waiting time from one-half to two-thirds for the first hop. The packet bursts will tend to spread apart as they propagate, so that the improvement in subsequent hops will be somewhat less.

Transport Layer Performance

Several studies have been published on the performance of the transport layer in the DNA structure.[18,19,20,21] One of the published studies is on timeout algorithms. We found that under sustained loss, all adaptive timeout algorithms either diverge or converge to values lower than the actual round-trip delay.[18] If an algorithm converges to a low value, it may cause frequent unnecessary retransmissions, sometimes leading to network congestion. Therefore, divergence is preferable in the sense that the retransmissions are delayed.

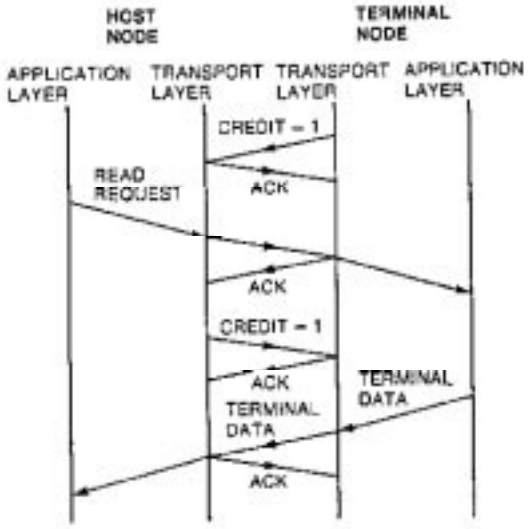


Figure 5 Eight Transport Level Packets

One key lesson we learned from the timeout algorithm research was that a timeout is also an indicator of congestion in the network. Therefore, not only should the source retransmit the packet on a timeout, but it should also take action to reduce future input into the network. There is a timeout-based congestion control policy called CUTE (congestion control using timeouts at the end-to-end layer) that manages these actions.[19]

Among the new features of DNA Phase IV are cross-channel piggybacking, acknowledgment withholding, and larger flow-control windows. These features were introduced as the result of a study that concluded that straightforward terminal communication over a DECnet network would be slow. This conclusion lead eventually to the development of a new local area transport protocol, called LAT, for terminal communications. These enhancements were also added to the DNA transport protocol. This study is described below.

In the DNA structure, each transport connection has two subchannels, one for the user, and one for control. The user subchannel carries user data and their acknowledgments, called acks. The control subchannel is used for flow-control packets and their acks. Protocol verification can be easily achieved if the two subchannels are independent so that information on one channel is not sent on the other. In studying terminal communications over a LAN, we discovered that each terminal read took eight transport protocol data units (TPDUs), as shown in Figure 5. Each unit consists of two application level packets: a read request, and a data response. Each packet requires a link service packet from the respective receiver; this service packet permits the sender to send one packet. The remaining four units are transport level acks for these four packets.

Given the CPU time required per packet, we computed that communication for remote terminals takes four times as much CPU time as that for local terminals. Therefore, our goal was to improve performance by a factor of four. We proceeded in three ways to solve this problem. First, we modified application programs to utilize larger flow-control windows; second, we searched for ways to reduce the number of packets per I/O operation; third, we tried to reduce the CPU time required per packet. The first goal was achieved by multibuffering, discussed later in the section "Application Layer Performance."

The second goal was achieved by

- Cross-channel piggybacking — This technique allows transport control acks to be piggybacked on normal data packets or acks.
- Delayed acks — The receiver can delay an ack for a small interval. This delay increases the probability of the ack being piggybacked on the next data packet.
- Ack withholding — The receiver does not acknowledge every packet, particularly if expecting more packets from the source. The source can explicitly tell the destination to withhold sending an ack by setting a "No Ack Required" bit in a packet.
- No flow control — This option allows flow control to be disabled for those applications operating in request-response mode and thus having a flow-control mechanism at the application level.
- Multiple credits per link service packet — Credits are not sent as soon as each buffer becomes available. Unless the outstanding credits are very low, a link service packet is sent only when a reasonable number of buffers becomes available.

To achieve the third goal, reducing the CPU time per packet, we used a hardware monitor to measure the time spent in various routines. We found that in a single-hop loopback experiment, only one third of the CPU time at the source was attributable to DECnet protocol routines. The remainder was associated with the driver for the line adapter; operating system functions, such as buffer handling and scheduling; and miscellaneous overheads associated with periodic events, such as timers, status updates. Of the time spent in the DECnet protocol, 30 percent was spent in counter updates and statistics collection. Similarly, 21 percent of the time spent in the link driver was used in a two-instruction loop that implemented a small delay. The net result of modifying these routines and implementing the architectural changes mentioned above is that we achieved our target of improving the performance by a factor of four.

Application Layer Performance

The three key network applications are file transfer, mail, and remote terminal communications. Earlier, we discussed some of the terminal communication performance issues. The new LAT protocol has been designed to provide efficient terminal communication. This protocol and its performance are described in this issue of the *Digital Technical Journal*.[22] In this section, we will describe some performance issues in file transfer.

File transfer in DNA takes place via a network object called a file access listener (FAL), which in turn uses an application level protocol called the disk access protocol (DAP). Measurements of an initial version of FAL revealed that the remote file transfer took an excessively long elapsed time. A subsequent analysis showed that the single-block "send-and-wait" protocol used by FAL was responsible for that excessive time. The local FAL waited for the remote write operation to finish before sending the next block. Thus the advantage of larger flow-control windows offered by the transport protocols were ignored by the application software. The suggested remedies were to allow multiblocking and multibuffering.

Multibuffering consists of allowing several buffer writes to proceed simultaneously, an action similar to the window mechanism used at the transport layer. Multibuffering allows parallel operations at the source and destination nodes and at the link, thus considerably reducing the elapsed time and enhancing throughput. Experiments have shown that there is considerable gain in throughput as the buffering level increases from one to two. Further increases do result in better performance, but the amount of gain is smaller.

Multiblocking consists of sending more than one block per FAL write, which decreases CPU time and the disk rotational latency (the time spent in waiting for the disk to come under the heads at the start of each write). As with multibuffering, the elapsed time is considerably reduced and the throughput is enhanced.

Workload Characterization and Traffic Analysis

The results of a performance analysis study depend very heavily on the workload used. To keep up with continuously changing load characteristics, we regularly conduct system and network workload measurements. Workload characterization studies enable us not only to use the correct workload for our analysis but also to implement our products more efficiently.

A study of the system usage behavior at six different universities showed that a significant portion (about 30 percent) of the user's time is spent in editing.[23] This conclusion led us to use our text editor (EDT) as the key user level benchmark for network performance studies. The study results also led to the transport layer performance improvements discussed earlier.

A study of network traffic at M.I.T. showed that the packets exhibit a "source locality."[7] That is, given a packet going from A to B, the probability is very high that the next packet on the link will be going either from A to B or from B to A. These observations helped us improve our packet-forwarding algorithm in bridges. The forwarding decision is cached for use with the packets arriving next. A two-entry cache has been found to produce a hit rate of 60 percent, resulting in significant savings in table lookup.

The principal cause of source locality is the increasing size of data objects being transported over computer networks. The sizes of data objects have grown faster than packet sizes have. Packet sizes have generally been limited by the buffer sizes and by the need to be compatible with older versions of network protocols. Transfer of a graphic screen could involve data transfers of around two million bits. This increase in information size means that most communications involve a train of packets, not just one packet. The commonly used Poisson arrival model is a special case of the train model.[7]

The two major components of a networking workload are the packet size distribution and the interarrival time distribution. J. Shoch and J. Hupp made the classic measurements of these components for Ethernet traffic.[24] Their tests have been repeated many times at many places, including Digital. The bimodal nature of the packet size distribution and the bursty nature of the arrivals are now well accepted facts; we will not elaborate further on them.

The utilization of networks is generally very low. Measurements of Ethernet traffic at one of our software engineering facilities with 50 to 60 active VAX nodes during normal working hours showed that the maximum utilization during any 15-minute period was only 4 percent. Although higher momentary peaks are certainly possible, the key observation, confirmed by other studies as well, is that the network utilization is normally very low. While comparing two alternatives, say H and L in Figure 6, some analysts would choose alternative H, which performs better than L under heavy load but worse under light load. Our view of this choice is quite different. We feel that, while high performance at heavy load is important, it should not be obtained at the cost of significantly lower performance at normal, light load levels. Therefore, the choice between L and H would also depend upon the performance of H at low loads.

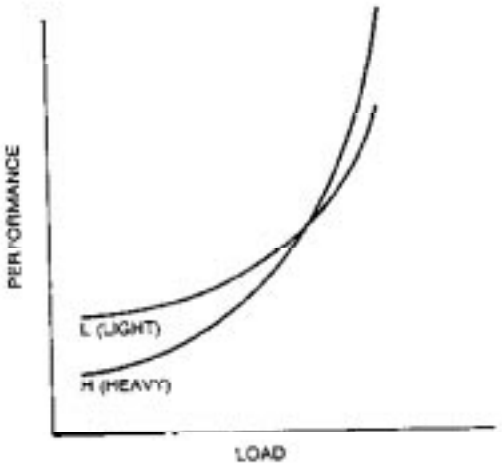


Figure 6 Preferred Alternatives

Traffic monitoring is also used to study the performance of networking architectures. Table 1 shows a breakdown of the DECnet traffic during the normal working hours at the same engineering facility. All values represent the average of several 15-minute sampling intervals. The maximum and minimum values observed during the monitoring period give an idea of the large variability. The DECnet traffic typically accounts for 86 percent of the total packets at this facility. The routing overhead is very low (5 percent). The protocol overhead comes mostly from the end communication layer (ECL), which provides error control (acks), flow control, sequencing, and connection management. Forty-four percent of DECnet packets and 39 percent of DECnet bytes are user-transmitted data. Thus the ECL overhead is approximately one packet per user packet, which is low considering that most ECL connections are of short duration (one file transfer of a few blocks). Furthermore, the results of this study confirm that we actually did reduce the transport packets per application level packet by 50 percent.

Table 1 DECnet Packet Statistics

	Average	Maximum	Minimum
DECnet packets (percent of total packets)	86	99	60
DECnet bytes (percent of total bytes)	68	99	32
Routing packets			
Percent of DECnet packets	4	52	1
Percent of DECnet bytes	5	66	2
Transport (ECL) packets			
Percent of DECnet packets	96	99	48
Percent of DECnet bytes	95	98	34
ECL data packets			
Percent of DECnet packets	44	51	3
Percent of DECnet bytes	60	81	4
User transmitted data			
Percent of DECnet bytes	39	68	2
Percent of ECL data bytes	65	84	50
IntraEthernet ECL data packets			
Percent of DECnet packets	79	91	34
Percent of DECnet bytes	79	93	24

The table also shows that, typically, 80 percent of all packets and bytes are used in intranetwork communication. That is, only 20 percent of the observed traffic originated from or was destined for a node not in the facility.

Summary

Performance analysis is an integral part of the design and implementation of network architectures at Digital. Analytical, simulation, and measurement techniques are used at every stage of a network product's life cycle. This conscious effort has made Digital the industry leader in networking.

Over the past decade, the link speeds have increased by two orders of magnitude; however, the performance at the user application level has not increased in proportion, mainly because of high protocol processing overhead. The key to producing high performance networks in the future, therefore, lies in reducing the processor overhead.

We have described a number of case studies that have resulted in higher performance for the Digital Networking Architecture. This performance increase has come about by reducing the number of packets, simplifying the packet processing, and implementing protocols efficiently.

Acknowledgments

The studies reported in this paper were done over a period of time by many different people, some of whom are no longer with Digital. We would also like to acknowledge Dah-Ming Chiu (DEUNA buffers) and Stan Amway (traffic measurements). We would like to express our gratitude to them and other analysts for allowing us to include their results in this paper.

References

1. W. Hawe, "Technology Implications in LAN Workload Characterization," *Proceedings of the 1985 International Workshop on Workload Characterization of Computer Systems and Computer Networks* (October 1985): 111–130.
2. K. Ramakrishnan and W. Hawe, "Performance of an Extended LAN for Image Applications," *Proceedings of the Fifth International Phoenix Conference on Computer Communications* (March 1986): 314–320.
3. M. Marathe and S. Kumar, "Analytical Models for an Ethernet-like Local Area Network Link," *Proceedings of SIGMETRICS'81* (1981): 205–215.

4. I. Chlamtac and R. Jain, "Building a Simulation Model for Efficient Design and Performance Analysis of Local Area Networks," *Simulation* (February 1984): 55–66.

5. R. Jain, "Using Simulation to Design a Computer Network Congestion Control Protocol," *Proceedings of the Sixteenth Annual Modeling and Simulation Conference* (April 1985): 987–993.

6. J. Spirn, J. Chien, and W. Hawe, "Bursty Traffic Local Area Network Modeling," *IEEE Journal on Selected Areas in Communications*, vol. SAC-2, no. 1 (January 1984): 250–258.

7. R. Jain and S. Routhier, "Packet Trains: Measurements and a New Model for Computer Network Traffic," *IEEE Journal on Special Areas in Communications* (Forthcoming, September 1986).

8. N. La Pelle, M. Segar, and M. Sylor, "The Evolution of Network Management Products," *Digital Technical Journal* (September 1986, this issue): 117–128.

9. M. Sylor, "The NMCC/DECnet Monitor Design," *Digital Technical Journal* (September 1986, this issue): 129–141.

10. R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems," Digital Equipment Corporation Internal Research Report, TR-301 (1984).

11. R. Jain and I. Chlamtac, "The P^2 Algorithm for Dynamic Calculation of Quantiles and Histograms without Storing Observations," *Communications of the ACM*, vol. 28, no. 10 (October 1985): 1076–1085.

12. M. Marathe, "Design Analysis of a Local Area Network," *Proceedings of the Computer Networking Symposium* (December 1980): 67–81.

13. W. Hawe and M. Marathe, "Performance of a Simulated Programming Environment," *Proceedings of Electro'82* (1982): 21/4/1–8.

14. M. Marathe and W. Hawe, "Predicting Ethernet Capacity—A Case Study," *Proceedings of the Computer Performance Evaluation Users Group Eighteenth Meeting (CPEUG'82)* (October 1982): 375–389.

15. W. Hawe, M. Kempf, and A. Kirby "The Extended Local Area Network Architecture and LANBridge 100," *Digital Technical Journal* (September 1986, this issue): 54–72.

16. W. Hawe, A. Kirby, and B. Stewart, "Transparent Interconnection of Local Networks with Bridges," *Journal of Telecommunications Networks*, vol. 2, no. 2 (September 1984): 117–130.

17. J. Spirn, "Network Modeling with Bursty Traffic and Finite Buffer Space," *Proceedings of the ACM Computer Network Performance Symposium* (April 1982): 21–28.

18. R. Jain, "Divergence of Timeout Algorithms for Packet Retransmissions," *Proceedings of the Fifth International Phoenix Conference on Computer Communications* (March 1986): 174–179.

19. R. Jain, "A Timeout Based Congestion Control Scheme for Window Flow Controlled Networks," *IEEE Journal on Special Areas in Communications* (Forthcoming, October 1986).

20. K. Ramakrishnan, "Analysis of a Dynamic Window Congestion Control Protocol in Heterogeneous Environments Including Satellite Links," *Proceedings of the Computer Networking Symposium* (Forthcoming, November 1986).

21. D. Chiu, "Simple Models of Packet Arrival Control," Digital Equipment Corporation Internal Technical Report, TR-326 (1984).

22. B. Mann, C. Strutt, and M. Kempf, "Terminal Servers on Ethernet Local Area Networks," *Digital Technical Journal* (September 1986, this issue): 73–87.

23. R. Jain and R. Turner, "Workload Characterization Using Image Accounting," *Proceedings of the Computer Performance Evaluation Users Group Eighteenth Meeting (CPEUG'82)* (October 1982): 111–120.

24. J. Shoch and J. Hupp, "Performance of the Ethernet Local Network," *Communications of the ACM*, vol. 23, no. 12 (December 1980): 711–721.