Introduction to Network Performance

digital

Order number EK-NETWK-TM-001.

This document was prepared using Interleaf technical publishing software (TPS).

CONTENTS

PREFACE

This book aims to introduce the basic concepts and terminology of network performance. It can help you find answers to such questions as:

- What constitutes good performance?
- How can performance and the understanding of it contribute to a business or company?
- How much importance should be given to performance?
- How is performance measured and predicted?
- What factors affect performance?
- How can performance be improved?

The book aims to help you set reasonable performance expectations before purchasing, implementing, or reconfiguring a network. It provides a base from which you can intelligently formulate performance requirements and evaluate performance results. It should help you see where performance problems lie, or where they are about to develop, and how to respond to them.

Where performance issues or problems become too difficult to deal with — and they often may — you probably will seek technical assistance. When you do, this book helps you ask the right questions.

The book does not pretend to have the ultimate say on what network performance is all about. Many of the concepts and principles formulated today will probably be outdated not long from now. So, this book is Digital's way of sharing with you its understanding of what network performance is *today*.

The book consists of three parts:

Part 1 (Chapters 1 - 3) is an overview of network performance. It introduces the terms and concepts.

Part 2 (Chapters 4 - 7) describes the major performance measures — the criteria for evaluating network performance, such as response time and throughput. Each of chapters 4 through 7 defines a performance measure and explains:

1. What you should expect.

2. How you can measure or evaluate it.

3. How it can be misunderstood or misrepresented.

4. What makes it better or worse.

5. How it can be improved.

Part 3 includes the appendixes, a list of references, and a glossary. The appendixes provide technical information relevant to certain chapters of the book.

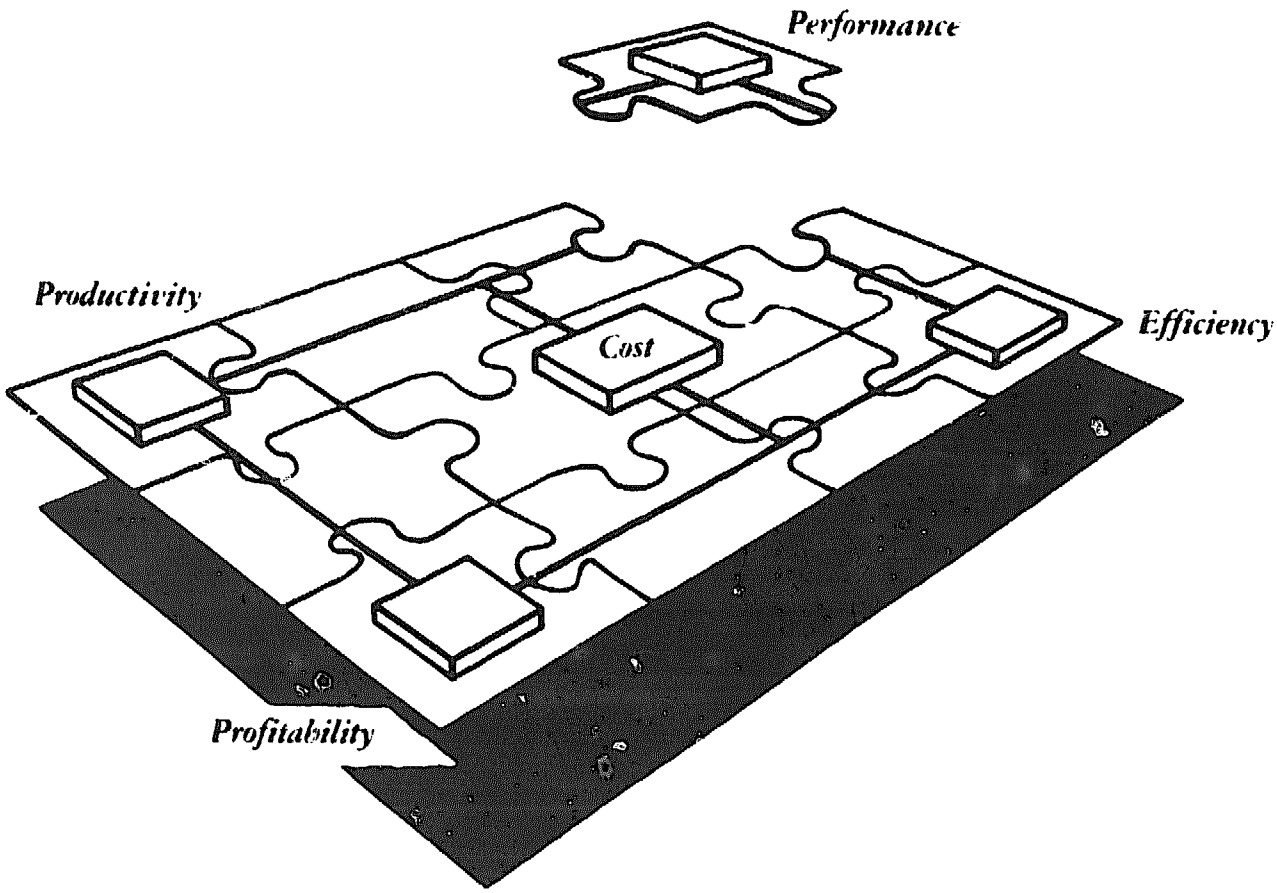
Any bracketed number in text (for example, "refer to [3]") specifies an item in the **References** for the chapter being read.

Digital recommends the following publications to enhance the understanding of networking in general and Digital networking products in particular:

- *Introduction to Local Area Networks*
- *Digital's Networks: An Architecture with a Future*
- *Overview of Digital Networking Products*

PART 1
Overview of Network Performance

1 WHERE PERFORMANCE FITS IN



Today, computer networks are growing rapidly in number and size, penetrating all types of business, government, and industrial enterprises. As more companies install networks, and as networks grow in size, network performance takes on greater importance. This chapter discusses why the demand for networks has increased so sharply in recent years and where performance fits into the network picture.

THE GROWTH OF NETWORKING

A network is a group of computers that are connected to each other by physical media (such as cables, telephone lines, or satellites) to share information and resources. A **local area network** (LAN) is a network that spans a limited geographical area, such as a building or cluster of buildings, and uses new technologies of data transfer such as high-speed buses. A **wide area network** (WAN) is a network that spans a large geographical area, such as several cities or countries.

Today, large networks have become commonplace. The average size of local area networks (LANs) is now growing at a rate of 150% per year. The quick spread of office automation, which includes microcomputers, word processing systems, and personal computer networks, has been a major factor. As of this printing, the number of office desks being connected to networks is growing at a rate of about 40 to 45% annually. Some larger companies are implementing networks to connect 20,000 or more workstations into office automation systems.

One reason for the growing interest in networking is the success of those organizations and companies already using networks. A recent survey found that 70% of the companies that have integrated their automation, telecommunications, and data processing with networking have enjoyed "substantial or very substantial productivity improvement." [3]

One large business reports that, with the use of Digital's ALL-IN-1 office-automation networking system, managers have experienced a 23% productivity gain, while the secretarial and administration staff has experienced a 53% gain.

Needs Dictate the Growth of Networks

Networks grow out of the needs of the enterprises that use them. The most prominent of these needs are:

- To become more competitive by operating faster and more efficiently.
- To tighten management control (large, decentralized organizations use networks to integrate operations in an intelligent manner).
- To collect, store, and quickly access the growing mass of data.

Another major factor that has stimulated the growth of networks is cost. With advances in the computer and communications technology, costs for computers and networks have decreased considerably while, at the same time, their versatility and capacity have increased.

The most valuable element in the network is the data itself (not the hardware, computers, or communications lines). For many enterprises, information is a vital competitive

weapon, and the network is used to collect that information in one or more databases. The network's value is mainly its ability to provide shared access to that data. That is where business realizes true productivity gains.

Networks Improve Employee Productivity

Local area networks (LANs) benefit employees by bringing the disparate parts of their organization closer together. LANs provide a way for employees within a building or complex to communicate and to share information and resources, especially expensive devices such as mass storage devices and laser printers. Without shared access to these devices, they would be seldom used; a large investment would have limited returns. LANs are cost-effective: their communications capability costs are commensurate with the low cost of the computers being connected.

Wide area networks (WANs) allow employees to share resources over a greater area. In large companies, employees in offices distributed across several cities or countries and in different time zones, can share information with the ease and immediacy that would result from their being in adjacent rooms. In such widely-dispersed enterprises, WANs help to tighten operations and to speed up the transfer of critical information.

THE IMPORTANCE OF NETWORK PERFORMANCE

Network Performance is a measure of how well the network performs the tasks required of it. Network performance is a "superset" of system performance. With system performance, concern is limited to the performance of a single system. With computer networks, the concern expands to the performance of a group of computer systems.

As the number of computers participating in a network increases, user satisfaction becomes a greater issue. How should you allocate network resources to make sure all users are satisfied? This is just one question that understanding network performance can answer.

Attention to performance becomes more critical as networks grow in size and complexity. In the past, when networks were much smaller, it was easier to isolate faults or to determine how to improve the level of network performance. With today's larger networks, computer systems that are geographically dispersed are difficult to access for fault isolation. Even within a local area, fault isolation may be difficult, because you must control numerous computer systems. Thus, automated fault-isolation and performance monitoring are now necessary.

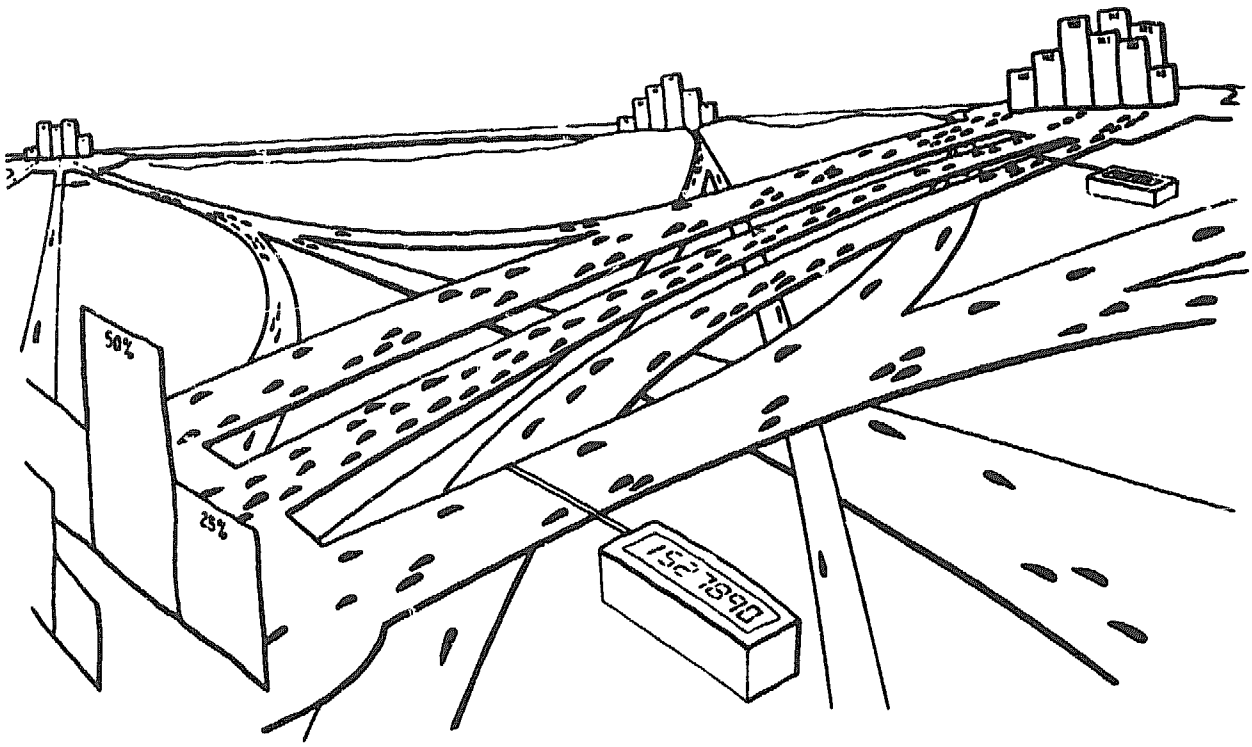
You assemble a network to solve a problem. By knowing the needs of your situation, you can ensure that the network performs to solve that problem in a satisfactory way. By

setting reasonable performance goals, you can better choose network components that match capabilities to needs. You can avoid overspending for faster computers and communications lines when less expensive equipment will suffice.

Understanding network performance serves many useful purposes. It aids in the following activities:

- Selecting and configuring network products
- Designing the overall network
- Managing a network
- Evaluating a network
- Troubleshooting network problems
- Planning for future needs

2 NETWORK PERFORMANCE



Drivers see the performance of a highway system mostly in terms of how quickly they can get to their destination. Managers of the highway department see the performance of the highway system in terms of how large a traffic flow they can maintain on each highway without having excessive congestion and without exceeding the highway budget.

A computer network is like a "data highway." The network user's and manager's concerns are similar to those of their counterparts in a highway system. This chapter introduces basic concepts and issues that occur in managing network performance.

PUTTING PERFORMANCE IN PERSPECTIVE

A network's *effectiveness* is the degree to which it produces the results or responses expected of it. A network's *efficiency* is the degree to which it is able to produce these results with minimum expense or waste of resources.

To evaluate the performance of a network, you should select one or more **performance criteria**, characteristics that you can measure in quantitative terms, such as response time and throughput. The performance criteria selected depend on the applications and requirements of network users.

This book discusses only those criteria that are easily quantifiable. Those that are not — such as ease of use, security, adaptability to environmental changes, and compatibility for multivendors — fall into the category of functional aspects.

When selecting, assembling, or tuning a network, you should consider cost, versatility, reliability, availability, maintainability, and evolvability. The versatility of a network is the diversity of functions that it provides. Reliability concerns how long it can perform suitably without failure. Availability pertains to how often the network is available for use. (Note that a network with high reliability does not necessarily have high availability, and vice versa. For example, a network may seldom fail, but whenever it does it may be unavailable for a long time.) The maintainability of a network pertains to how easy it is to control and monitor network activity and to correct network problems. Evolvability refers to the ease with which you can expand the network.

You cannot weigh performance in a vacuum, without regard for these other factors that contribute to the value of a network. For example, if you emphasize performance, you may need to pay more for network components of higher capacity and for the tuning required to raise the level of performance to meet your needs. You may also have to sacrifice some of the versatility of the network so that you can concentrate resources on the most essential tasks.

If you overemphasize performance and underestimate other factors, the level of performance may be irrelevant, no matter how high it is. For example, what good is high performance if the network is often inoperable because of component failure or maintenance needs?

How easy is it to add or remove devices? In some networks, you must stop operations before adding or removing certain devices (other devices may have to be "reprogrammed" to take into account the changed environment). In Digital's DECnet, changes are simple to make. To add a node, you simply plug it in and the network has the intelligence to teach itself where that new node is. (A **node** is any intelligent device that communicates with other devices in the network.)

Even when performance is a low priority, it must be predictable and stable (within reason), regardless of the traffic load, error rates, or other fluctuating conditions.

THE USER'S VIEW AND CONCERNS

For users, network performance is a measure of how effectively their applications work over the network and how quickly the network performs network-related tasks, such as setting up a link between the source and destination.

The performance users expect depends on the type of applications they use. With interactive applications, such as transaction processing, credit card verification, videotex, and remote system management through virtual terminal connections, users care the most about network responsiveness. All users of the network shown in Figure 2-1 depend on quick responses to their requests. Parts-order clerks do not want to keep customers waiting on the phone. Stock personnel need to be able to find what is in inventory at any time. They depend on up-to-date information.

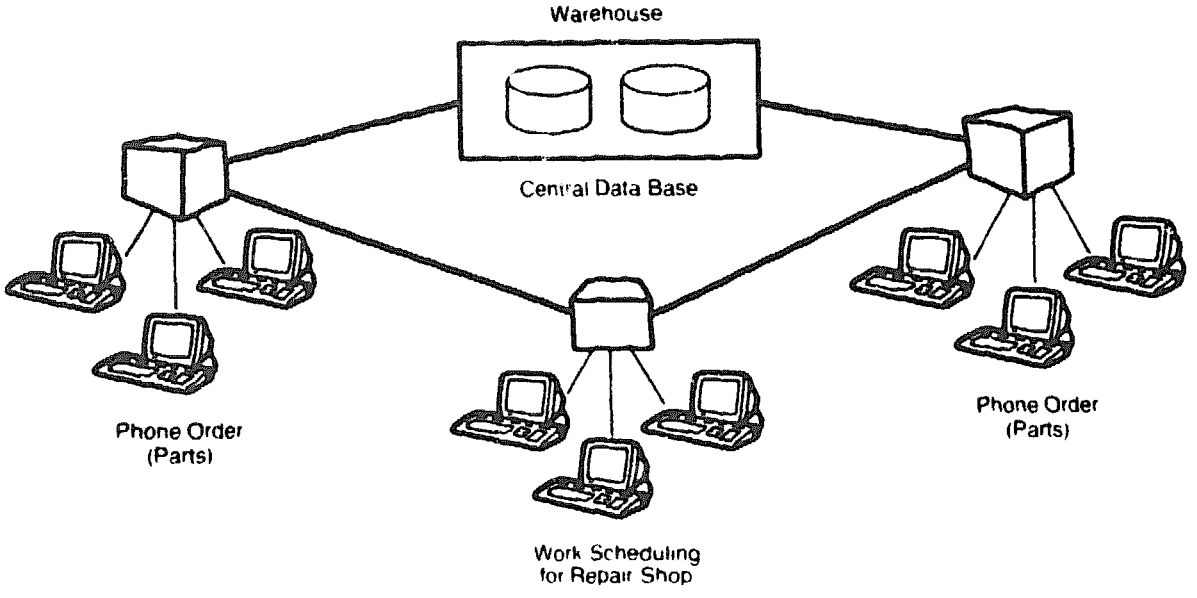


Figure 2-1 Warehouse Inventory Control

For interactive applications, perhaps the most important measure of network performance is response time. **Response time** is the period that elapses between a user request and the response at the user's terminal. The response can be, for example, the requested information or the system prompt.

For applications that involve bulk transfers of data, such as file transfers and virtual terminal applications, users are concerned with throughput. **Throughput** is the amount

of data that can be transferred from one location to another in a given period of time. Figure 2-2 shows a network used for process control within a factory. The main computer at the top stores the database and controls the processes. Great amounts of information "funnel" from the lower levels to the main computer, and thus throughput requirements increase at points nearest the main computer. Note that response time is also important in this application. For example, the data collection systems must operate in real-time to be able to keep up with the high rate at which data is supplied to them from the process level.

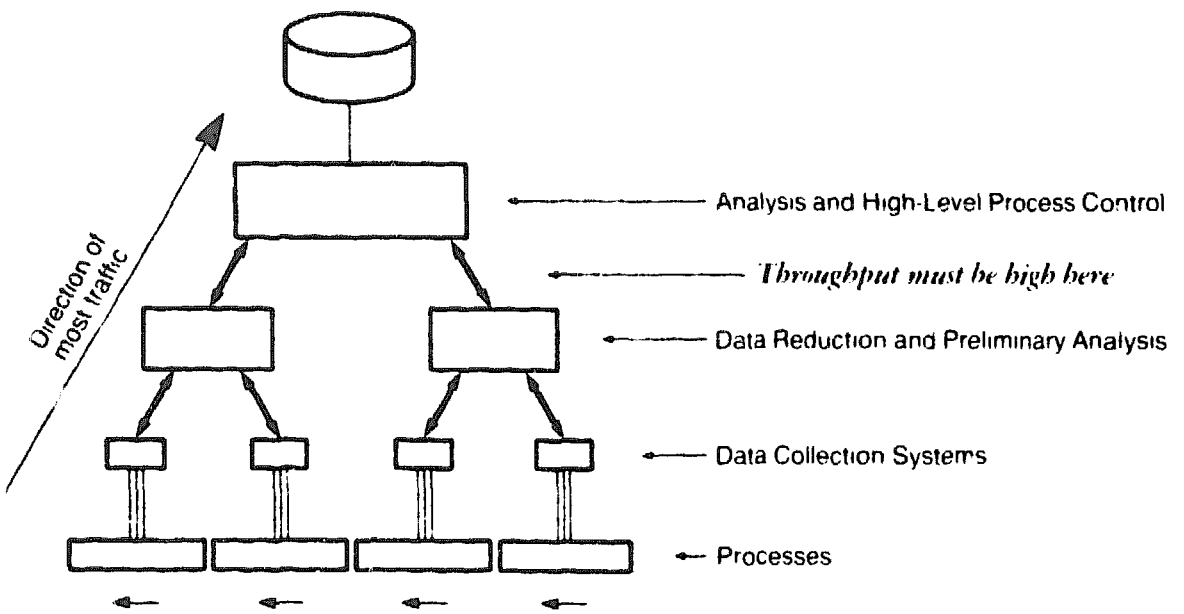


Figure 2-2 Process Control with High Throughput Demands

THE MANAGER'S VIEW AND CONCERNS

As a network manager, you should be concerned with the technicalities "under the hood" of the network. You oversee the operation of the network and monitor its performance to ensure that it runs smoothly. You configure and tune the network to maintain performance levels acceptable to users. You reconfigure network resources to reflect changes in user needs and traffic patterns.

As a system manager, you are responsible for the operation of individual nodes making up a network. Although you are not responsible for the operation of the network, you can contribute greatly to its performance by keeping user nodes internally tuned.

As a network or system manager, you must closely monitor the network resources to make sure they are working smoothly. **Resources** are any parts of the network where traffic is processed, stored, channeled, or printed, such as central processor units (CPUs), storage disks, the communications lines, and line printers. An application requires one or more network resources at each stage of execution, and if it does not have them all, delays occur until the necessary resources are available.

To evaluate the performance of a network in terms of efficiency, you should look at internal delays and resource utilization measures. **Internal delays** are imposed by the components and internal functions of the network. For example, the communications lines between nodes impose delays that depend on the speed of each line and on queuing delays. When a device cannot process or forward data fast enough, subsequent incoming data joins a **queue** where it waits its turn to be served. The **queuing delay** is the length of time that a unit of data has to wait before it is served.

Resource utilization describes the percentage of time that the resource is busy. It indicates how much use a resource is getting. Utilization measures help not only to evaluate current needs but to project future needs should network usage expand.

COMMON AND COMPETING CONCERNS

The effectiveness of a network depends on its internal efficiency. Just as you get poor mileage from an automobile if you do not have the engine tuned, so you get poor results from a network if you do not have it configured and tuned correctly. In addition, the costs for equipment, operations, maintenance, and performance tuning must be reasonable.

The goals of effectiveness and efficiency can be at odds with each other. Consider a highway system again. A highway is more effective for users when its capacity always exceeds the traffic flow demanded of it. This helps ensure that users can drive without traffic-jam delays. In contrast, highway department managers want the highway to be used efficiently. They cannot afford to build highways as large as users would prefer. So, the wishes of users (effectiveness) and of managers (efficiency) often have to be compromised to achieve the best overall results.

UNDERSTANDING NETWORK PERFORMANCE

In highway systems, the roads are analogous to network links (the connections between nodes). Towns or cities are analogous to the nodes of a network. The round-trip time for traveling to a given city and back is analogous to response time. The number of cars that travel over a particular road (or packets of data over the network) in a given time measures the throughput.

Networks such as DECnet transfer data in groups of bits called **packets**. A packet usually includes both user data (passengers or goods) and control data (a bill of lading). Some packets just contain control data. The network uses them to control communications. Control data includes the destination address, source address, and error-control information. This control data is part of the **protocol overhead** incurred in communications. Protocol overhead plays an important part in network performance.

Protocols are rules that govern the format and control procedures of transmitted data. Protocols are necessary for data communications in much the same way that driving regulations are necessary for safe and orderly transportation.

Improperly designed, protocols can hamper communications and performance. For example, if too much control data or protocol processing is required for each packet, a relatively small amount of user data is transferred in a unit of time. The ratio of user data to control data is one way to measure the efficiency of a network protocol. A low ratio means the network does more work with less benefit to the user in terms of throughput.

Communications protocols can serve many purposes. As the number or complexity of services required by users increases, the protocol overhead can also increase. This increase can come at the expense of performance. Thus, a balance has to be posed between the number of services (versatility) and performance. Proper design of network software minimizes the effect on performance.

Some situations dictate greater overhead and less performance. For example, in applications requiring error-free transmission, such as in banking and electronic funds transfers, greater overhead is the expense paid for better error-control procedures.

Appendix A describes some of the common networking technologies and their effects on performance.

MANAGING NETWORKS

Managing a network for optimun performance always involves tradeoffs. External tradeoffs include such things as performance against cost or versatility. Internal tradeoffs involve the performance characteristics themselves, such as throughput versus response time.

With internal tradeoffs, the improvement of one aspect of performance does not necessarily improve another, nor does it necessarily improve the total performance of the network. For instance, by increasing the throughput capacity of a component, you might cause an increase in traffic that overloads another part of the network.

To attain optimum performance for all aspects of a complex network, you must keep a clear picture of what is going on. The use of monitors, such as Digital's Network Management Control Center (NMCC), helps give you such a picture.

Network Behavior

One of the most important variables used to study network performance is the workload or traffic intensity. By experimenting with different levels of traffic, you can find the optimal operating point for the network.

Networks behave in different ways as the workload increases. Response times can grow or become irregular, while throughput can reach a maximum or, in extreme cases, even decline. Following is an important rule about network performance:

As the workload on a network increases, it inevitably reaches a point where performance begins to degrade. When performance degrades, it should do so smoothly so that performance problems do not suddenly occur and predictability is preserved.

The performance of some networks does not actually degrade in the sense mentioned above, but rather "blocks" users from gaining access when demands are excessive. For example, in a telephone network, you can be blocked from accessing the network when it is congested (your attempt to make a call is met with a busy signal). However, once you are granted access to the network (that is, you successfully make a call) service does not degrade during the call, regardless of changes in the network's workload. Response times are guaranteed.

Managing Network Resources

A large part of managing a network lies in managing sound network resources. You should be concerned with all resources where processing takes place, even if the delays they contribute are negligible.

A resource that is under-utilized may perform extremely well, but it might be too powerful or expensive for the demands placed upon it. For example, if you install a device that can handle 10,000 messages per second, but messages are sent at a rate of only 500 a second, then that device is too powerful. A less powerful device could be used just as effectively (however, keep in mind that some high-power devices on the market

are actually less expensive than their low-power counterparts). Chapter 7 covers resource utilization in more detail.

With any resource, it is a good idea to plan for some spare capacity to handle peak loads or to serve as an alternate route if a part of the network fails. When a resource reaches its maximum capacity and cannot handle any more traffic or requests, queuing delays increase considerably.

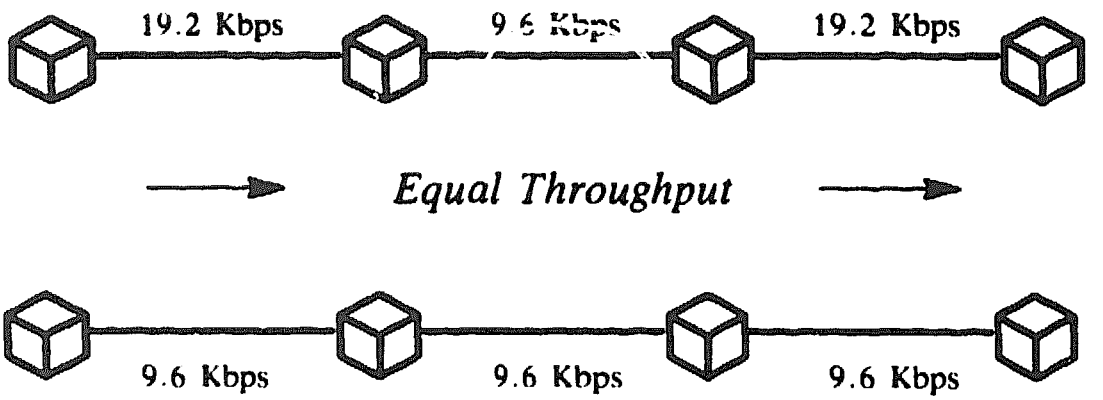


Figure 2-3 Effect of Slowest Link

Managing Bottlenecks

A resource that reaches saturation can become a network bottleneck. A **bottleneck** is a point where traffic is delayed or blocked. Bottlenecks can significantly restrict network performance. As shown in Figure 2-3, the slowest link on a path in a network limits the total performance of that path. (This applies especially to the most heavily used links.)

Bottlenecks can have a "ripple" effect on the network: one bottleneck can cause other bottlenecks. For instance, a bottleneck at one node on a path can force other nodes "upstream" to delay or slow down their processing. They have to buffer (temporarily store) data until the first bottleneck is removed. Meanwhile, these nodes become bottlenecks for other nodes upstream from them.

The main factors behind bottlenecks are the traffic load on the network and the capacities of network components. When network traffic is light, the workstations or nodes are the most likely bottlenecks. Network performance can be as high as the capabilities of the workstations or nodes. When network traffic is heavy, network transmission media are the most likely bottlenecks. Some causes of bottlenecks are:

- **An interface between two communications channels of different capacities.** For example, with an interface between a high-speed line and a low-speed line, data transmitted from the high-speed line through the interface to the low-speed line backs up when the throughput exceeds the capacity of the low-speed line.

- **An application where input is small and the associated output is large.** For example, in a transaction system where input consists of short requests and the output consists of many screen displays of data, the processor of those requests can become a bottleneck if it does not handle the incoming requests quickly enough.

- **A processing unit whose priority is so low that it is constantly interrupted.** When a task or unit is constantly interrupted, traffic can back up while waiting to be processed or transmitted.

- **Mismatch of facilities, resources, or requirements relative to the demands.** For example, if the number of buffers is insufficient for a task, the task might be interrupted or the incoming data might be delayed until buffers are available.

- **Poor topological design.** Topology design should be suited for the traffic flow in the network. For example, the star topology shown in Figure 2-4 is best for applications where several secondary nodes must communicate mostly with a central node. If this topology is used in a situation where the nodes at the "points" of the star talk mostly with one another, then all messages between the points must go through that central node to be routed to their destinations. This can overburden the central node.

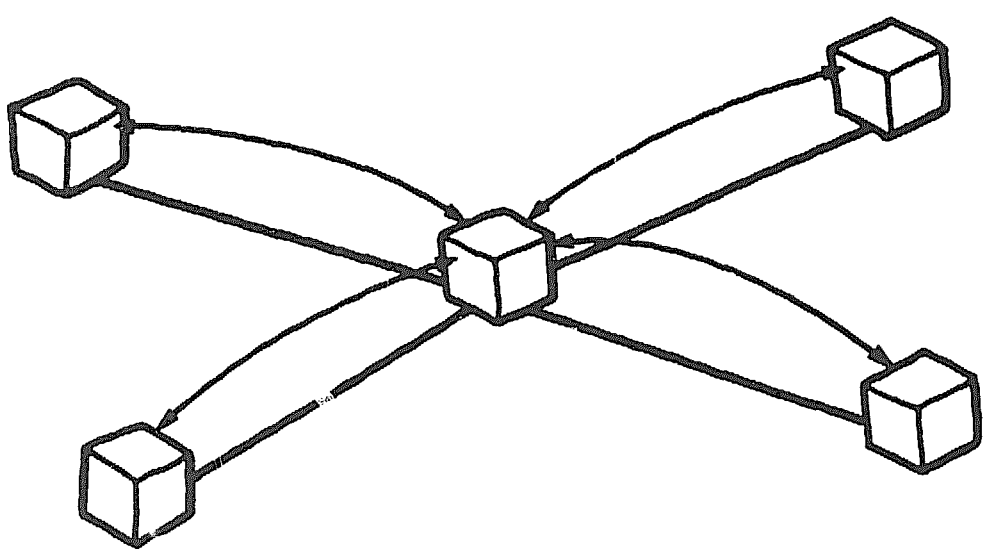


Figure 2-4 A Bottleneck Caused by Misapplication of Star Topology

Adjusting Network Software Parameters

In networks, there are variables or **parameters** that you can adjust to achieve optimum performance. They may include the sizes of certain quotas, the sizes of packets and buffers, the length of certain timers or timeout values, and others.

Pipeline Quota

On VAX/VMS systems, a system manager can specify a parameter called the **pipeline quota** to determine how many messages the transmitter can send without a positive acknowledgment (ACK) from the receiving end of the link. The receiver returns an ACK to indicate receipt of the preceding message without error and that it is ready for another block. The transmitter waits for an ACK before sending the next message.

A larger quota usually ensures higher throughput, since the transmitter can send more data per ACK. (This depends on the channel being relatively error-free. If errors are numerous and the quota is large, then each negative acknowledgment from the receiving end requires a large number of retransmissons. This would degrade throughput.)

Message and Buffer Sizes

When an application has large amounts of data to send, it may be better to send larger messages or blocks of data to reduce protocol processing overhead. If you had to mail a large number of goods, you would probably want to pack them together in a few large containers, rather than packaging each item separately in many small containers. Your overhead (the cost of the containers and postage, and the time required to address the containers) would be less this way.

However, sometimes smaller messages are better, such as when you want to improve the perceived response time at a terminal. Also, shorter messages require smaller buffers and thus use less CPU memory. Large messages and buffers can consume the CPU's memory too rapidly, causing other processes to wait for available buffer space.

Choosing buffer sizes depends a lot on the average message size. If buffers are too big for average messages, buffer space is wasted. If the buffer size is too small, then larger messages require segmentation to fit the buffer size. Unnecessary segmentation can waste CPU time and degrade network performance. Following are guidelines for choosing buffer sizes:

- **Large buffers improve throughput for bulk data transfers**, such as file transfers. However, sufficient memory must be available.
- **High-speed lines perform better with large buffers** because fewer messages must be processed by the communications processors, which tend to be the bottleneck.
- **Small buffers are preferred where lines have high error rates.** Larger units of data have a higher probability of being affected by error conditions, and they require

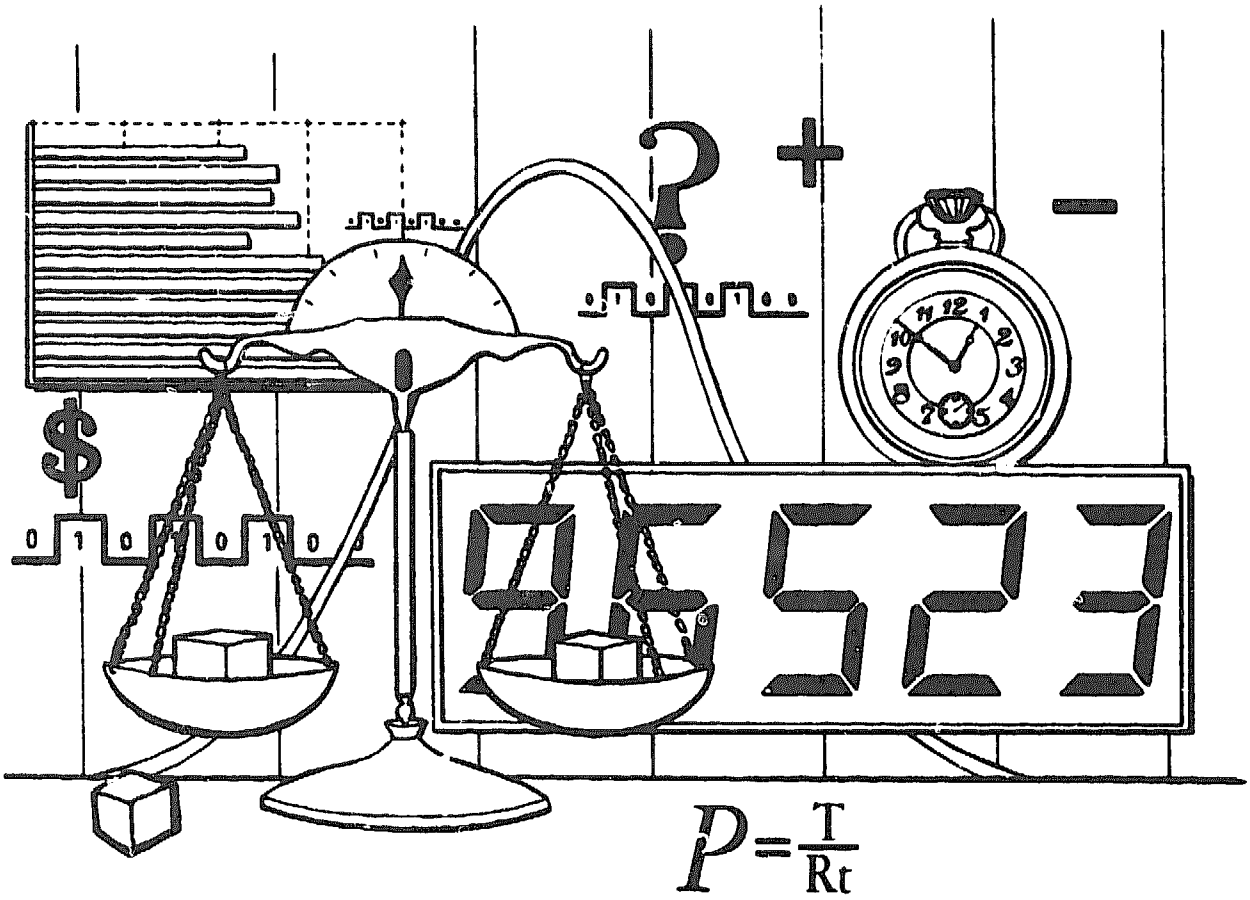
longer retransmission delays. (Note that on Ethernet, very large buffers are possible because the channel is relatively error-free.)

- **For interactive traffic consisting of small messages, or virtual terminal traffic with character I/O, the smallest buffer size is best.** Smaller buffers waste less memory and bring better response times.

Monitoring the Network

Managers of busy cities or highway systems analyze traffic problems and look for ways to improve the situation. They use **monitors** to analyze and measure the performance of the system. Similarly, if you are a network manager, you should monitor the network to observe its behavior, assess its utilization, and locate bottlenecks. In this way, you can avert serious problems.

3 EVALUATING NETWORK PERFORMANCE



Before evaluating the performance of a network, you should be aware of the obstacles that can prevent you from seeing it clearly and objectively. This chapter describes some of these obstacles, discusses ways to remove them, and attempts to answer some of the difficult questions about evaluating network performance. How do you evaluate network performance? What is good performance? What level of performance should you expect? Which expectations are reasonable and which unreasonable?

OBSTACLES TO OBJECTIVITY

If you are a network manager, you must try to unify the many, often contradictory, performance expectations of network users. Because you can seldom satisfy all user needs completely, you have to set priorities and find the right compromises.

Taking a Limited View – Intentionally

When shopping for a vehicle, you focus on the qualities most important to you. If you need a vehicle for hauling heavy equipment, your main concern is the power and load capacity of the vehicle. So, you will probably buy a truck. You intentionally overlook the truck's lower gas mileage as compared to a compact car or its slower acceleration rate as compared to a sports car.

Likewise, if your primary requirement for a network is to transfer large files from one point to another, you want a network that has a high throughput capacity. You are less concerned about the response times for interactive terminal activity. If you need to use the network for small transactions via interactive terminals, then quick response time becomes the main issue. You need instantaneous responses for small messages rather than high throughput for bulk transfers.

Taking a Limited View – Unintentionally

Be careful not to limit your view of performance unintentionally. For example, you may read that a certain network has terrific throughput. What might be unsaid is that only a small part of that throughput is user information and the rest is protocol overhead, the result of a cumbersome, poorly-designed communications protocol. (The experience is similar to buying a box of merchandise and upon opening it, finding that a large part of its contents is only packaging material.)

Another example of taking a limited view is testing or judging a network under an insufficient range of workload values. Figure 3–1 shows that the performance of a network is good at the load levels between points A and B. But if loads beyond point B are not tested, you would not know of the severe drop in performance just beyond point B. A network's performance must be studied under all possible workload conditions.

Settling for Too Little or Getting Too Much

When someone says that a transmission medium can provide links between systems at the speed of 5 megabits per second, that does not mean that the average throughput of applications using those links will be 5 megabits. (A **megabit** is one million bits. Mbps means megabits per second.) The actual throughput over a link is less than the hardware

ratings (such as the link or line speed). The limiting factors include the protocol overhead, the hardware devices that control communications over the link, and the applications themselves. You could get less throughput than anticipated.

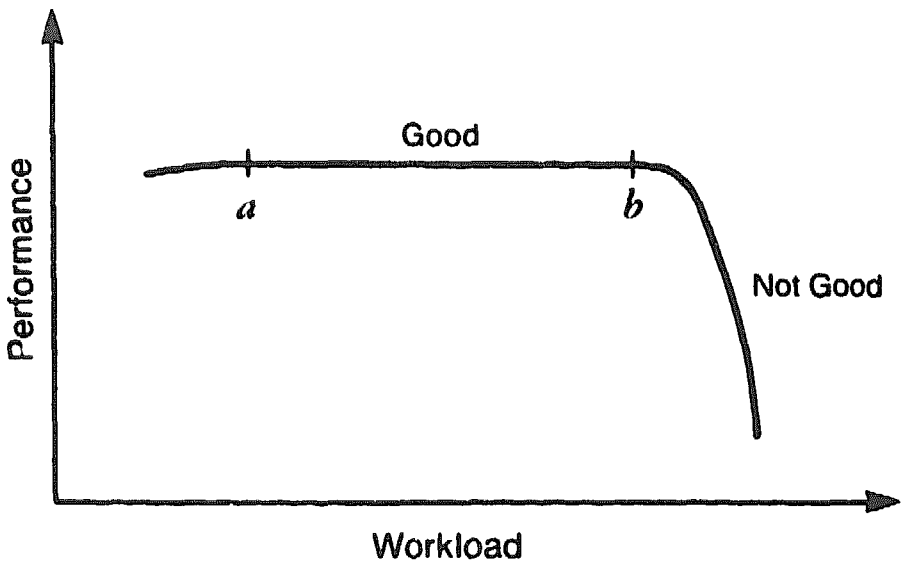


Figure 3-1 Do Not Fail to Get the Whole Picture

It is also possible to buy or design a network that has more performance capacity than you need or will ever need (however, network users usually find more uses for a network than originally planned, and so network usage often grows beyond expectations).

Perhaps the most realistic performance goal is that user applications perform *reasonably* well at a minimal cost, rather than optimally at *any* cost. The local post office does not use racing cars to deliver mail within the neighborhood; it just wants reliable vehicles that deliver the mail in a reasonable amount of time. The same applies to the delivery of electronic mail over a network.

Expecting Too Much

Unrealistic expectations usually arise from a misunderstanding of the details of networking or of the applications. Often the capabilities of certain components are overestimated, or the workload and traffic are underestimated.

If you are experiencing poor performance, the fault may not lie in the network. Often poor performance is the result of poor application design, one that fails to make the best use of the network. A network's performance can always be tuned and improved to some degree, but the effort may not be worth its cost if the cause of the problem is not the network itself.

OBJECTIVE EVALUATION

Performance evaluation determines how well the network does its job. The criteria must be those that users care most about. You use the expected values for these measures to judge the measured results. The final outcome of a performance evaluation is a mathematical model that specifies the relationship between inputs, parameters, and performance measures.

You need a method for evaluating network performance that is exacting. This section discusses yardsticks and standard, quantifiable measures that allow you to do that.

Performance Yardsticks

Evaluating performance means using benchmarks as yardsticks. A **benchmark** is a reference workload that serves as the basis for performance comparisons. It consists of a group of jobs or tasks that attempt to represent a typical workload for the component or network under evaluation.

Devising a benchmark for single systems is relatively simple. For networks, however, it can be complicated. Two or more networks can differ from each other in many ways, such as in size or in their transmission media. Thus, it may be unfair to compare networks of different types or under different conditions. When comparing networks or considering any performance results provided by vendors, consider (1) the type of network, and (2) the conditions under which results were obtained.

1. *Consider the Type and Characteristics of the Network*

When comparing networks, consider the types of networks, the uses of the networks, the technologies used by the networks, and the types of data transmitted by the networks (see Table 3-1). Also, consider the reconfiguration rules for the networks. For example, to add a node to the network, do you have to take the entire network down?

2. *Consider the Conditions Under Which the Network Is Evaluated*

The most important conditions to consider for evaluating network performance are the workload characteristics. Workloads must be equivalent for each network or component evaluated; otherwise, differences between various performance measures reflect only the differences in the workloads.

Table 3-1 Network Types and Characteristics

Characteristic	Possibilities
Type:	Local Area Network (LAN) Wide Area Network (WAN) Packet Switching Data Network (PSDN)
Organizational Structure:	Centralized, Terminal Network Distributed–Host Processing Network
Topology:	Ring Star Bus Other
Uses:	Virtual Terminal (SET HOST) File Transfer Electronic Mail Resource Sharing Transaction Processing Manufacturing–Control Processing A Combination of Applications
Technologies:	Dedicated Lines Standard Telephone (Circuit) Switching Private Branch Exchange (PBX)
Transmission Media:	Twisted–Pair Copper Coaxial Cable Baseband Broadband Fiber Optic Satellite Microwave
Data Type:	Voice Digital Data Video Graphics

In many real–time applications, the load varies considerably from one time to another. In a savings banking system, the workload may flood at noon each day because of the rush of customers who come at lunchtime. Thus, the performance may be good under the average workload during the day, but poor during the peak hour when performance is more crucial.

Other workload considerations include the nature of the traffic over the network:

- An application involves traffic among various nodes. What is the pattern of an application's traffic (which nodes are involved)? Are all nodes and lines capable of handling the traffic? How does the traffic vary? What other functions are the nodes performing?
- What are the traffic statistics (peak and average rates, stream or transaction traffic) at each node? (See Appendix B, Graphing Performance Data.)
- How well is the application distributed throughout the network?
- What size are the units of data or messages (message or packet size, for example)?
- Does incoming data have to be processed immediately? If not, how long a delay is acceptable?

Performance Measures

Measures such as response time, throughput, line utilization, and CPU utilization are useful as criteria for evaluating network performance. They set the framework for an evaluation; so you must define them clearly. There are two types of measures:

- **External measures** – those measures that indicate the productivity of the network. They include response time and throughput.
- **Internal measures** – those measures that indicate the internal efficiency of network components. These measures include CPU and line utilization, as well as network utilization, and they help you identify problems that degrade the results of external measures.

Good internal measures do not necessarily imply that the external measures will be good. There may be conditions external to the network itself that detract from the external performance. Also, the external measures of one network can be better than those of another, yet its internal measures can be worse. Thus, one network might be more effective than another with respect to a particular application, yet it might be using its components less efficiently.

SETTING REQUIREMENTS

Consider the situation where you have to set a single requirement applying to several applications. Suppose you have three applications (A, B, and C) for which the desired mean response times are 1, 3, and 4 seconds, respectively. If cost is not a problem, then

you can have the network designed to provide a mean response time of 1 second for all three applications combined. But what if such a design is not affordable? Should you use the mean of these three, 2.6, which significantly slights the most critical application (A)?

To offset this problem, you can calculate the **weighted mean.** You weight the applications according to their degree of criticalness, giving the highest weight to the most critical application. The weight you give is an arbitrary value, but the relationship between the various weights is significant. By giving relatively high weights to the most critical applications, you ensure that their needs are met. For example, to favor application A, you could give it a weight of 10, while giving B and C weights of 2 and 1, respectively. To find the weighted mean, you:

1. Multiply the required mean response time of each application by its weight.

2. Take the sum of these products.

3. Divide this sum by the sum of the weights.

So the weighted mean for applications A, B, and C is:

(1 x 10) + (3 x 2) + (4 x 1) divided by 13 = 1.54 seconds

Notice the weighted mean is less than the unweighted mean (2.6), and thus it favors the needs of application A.

MEASURING PERFORMANCE

To determine how well your network is performing, you must test and measure it. The tests are based on the conditions or benchmarks agreeable to all parties concerned. You can measure performance in several ways:

- **By direct measurement.** This involves hardware or software tools and is usually the most accurate method.

- **By use of models.** When direct measurement is not possible, such as when the network is in the planning stage, or when the workload is not available, analysts can construct mathematical models. These models allow the analysts to observe the performance of the network under prescribed conditions.

- **By simulation.** This also involves mathematics. By simulation, analysts construct a hypothetical situation (using one or more computer programs, for example) and measure the results.

For more information on models and simulation, see [1] and [9].

ANALYZING PERFORMANCE DATA

Performance analysis determines why the results came out the way they did and is done for the purpose of improving performance. You observe components to learn how they influence the performance of the network.

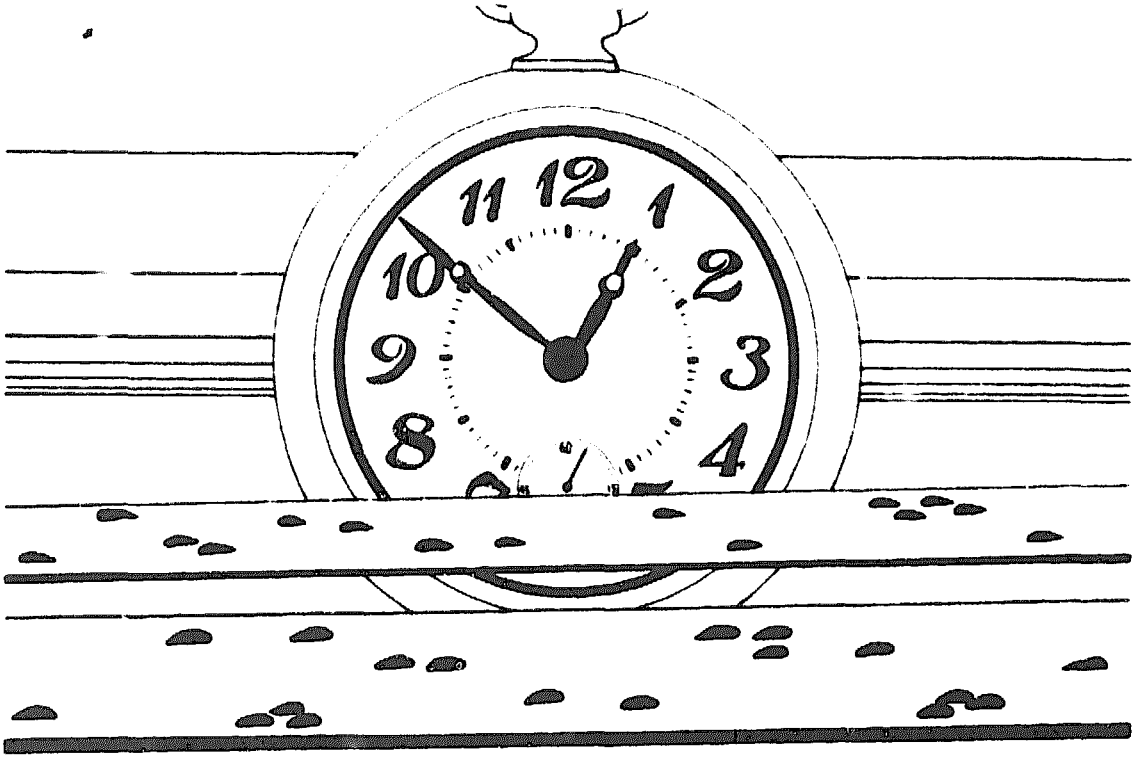
Performance analyses often take the form of **sensitivity studies** or **bottleneck analyses.** The analyses try to determine which components have the greatest affect on the performance of the network.

It is often useful to represent a network as a series of queues and servers (components that process the data in the queues). By observing queue lengths, server utilizations, response times, and throughputs, you can tell (1) for which component an improvement will have most impact on the network's performance, and (2) for which component a degradation will least impact the network's performance. This is important because improving or tuning performance is often a matter of give-and-take and of deciding how much to adjust a parameter value to reach the best performance level.

Appendix B explains some of the basic statistical tools you can use to make sense out of performance data.

PART 2
Network Performance Measures

4 RESPONSE TIME



Response time is the the first, and occasionally the most conspicuous, sign of performance that users notice. When response times are very long, users often pin the blame on the network. When response times are short, users often forget that the network is out there. Response times often depend largely on factors unrelated to the network itself, yet users often consider response time as the most critical index of the performance of a network. *Response time usually says more about an application and the systems that process the application than about the performance of the network.* This chapter discusses the elements that make up response time and the factors that influence it. It also discusses response time requirements and ways to measure and improve response times.

AN OVERVIEW OF RESPONSE TIME

Response time is an indication of a system's ability to take a user's request, interpret and process the data, and then return a response to the user.

On a single, stand-alone computer, response time is the delay between the moment you enter some form of input and the moment you receive the results. If a computer is connected in a network, such as for accessing a remote database in an inquiry/response application, the response time consists also of the time required for transferring the request to the destination and returning the response.

Response Times in Terms of the Application

In a savings bank, a customer wants to know his current balance. The clerk types a request at a terminal to display the current balance. The request is sent to a computer located several miles away at the main office that stores all customer information. The concern of the user (the clerk, in this case) is how long he has to wait for the information. Response time typically applies to real-time, interactive applications such as this.

In a chemical plant, sensors monitor the flow of fluids through a series of pipes and holding tanks. When a tank is full, the information is relayed to a central control station (a large computer such as a VAX 8650 or VAX 8800) that generates a command that stops the flow or redirects it to another tank. In an application like this, response time is the delay between the moment the event being monitored occurs (full tank) and the moment the control station responds. The concern here is that the information be received and acted upon in real-time or near real-time; any information received or acted upon too late is no longer useful and might even be costly.

In batch applications, response time is often known as **turnaround time**, the elapsed time between submitting a job and receiving the output. When an immediate response is not important (such as when you submit a job for overnight file transfers), the concern turns to how much data can be delivered in a given amount of time. This is throughput, however, not response time (Chapter 5 discusses throughput).

The Elements of a Response

Response time is really a series of elements or events in time. For a simple view of an application with remote database access, the elements line up as shown in Figure 4-1:

1. The event (r1) when the local terminal and system handle the request and prepare it for transfer.

2. The event (q1) when the request waits to access the network. This is called the **access time** or **admission time**. The data waits in a queue until the processor has time to transmit the data or until the network is ready to handle the data.

On most networks, you add the **connection time** to the picture. This is the time required to establish a link between the source and destination nodes. On dialup lines, you also add the time to dial and establish the connection (but only on the initial connection attempt).

3. The event (t) when the request travels over the network to the destination. This is **transmission time**, the time taken for a signal to travel in a network from the source to the destination. If there are intervening, **store-and-forward** nodes (those that temporarily store a message before forwarding it to its destination), the transmission time includes the store-and-forward delay(s) imposed at each of these nodes.

4. The event (q2) waiting for the destination processor to accept the request for processing.

5. The event (r2) when the destination processor processes the request and generates the response.

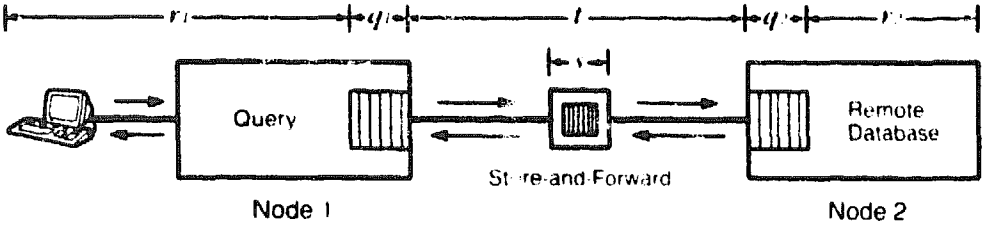


Figure 4-1 Elements of Response Time

After the response is generated, the same series of events continues on return to the requesting node, starting with access to the network.

As you can see, the network is not responsible for all the events or delays that constitute the response time. *In fact, network-induced delays often constitute only a minor part of the total delay experienced, especially in LANs.* In Figure 4-1, the network is solely responsible for only the transfer time and, where applicable, the store-and-forward delay. The network only *shares* responsibilities for delays at the other points. For example, the queuing delay for accessing the network depends on both network and local system conditions. *The delays contributed by the physical transmission are often negligible, depending on the size of the network and the amount of network traffic.*

When evaluating the performance of a network in terms of response times, you can consider all the elements shown in Figure 4-1. But, you should distinguish the *delay*

contributed by the network from *the delay contributed by the application(s) and other CPU processing unrelated to the network.*

Network delay can be broken down into two major parts: (1) the time to access the network and (2) the time to transfer the data across the network. These parts have varying degrees of importance depending on the application. For example, in one application, users have to access the network to type a command that transfers data over the network. After typing the command, users are done; they do not have to wait for a response. Here access time is more important than the transfer time, because users do not have to wait for the transfer to complete.

WHAT RESPONSE TIMES USERS EXPECT

You should be aware that the response time users expect depends mostly on the applications they are using. Users expect quicker response times for what they perceive as the simpler tasks. On the other hand, they can expect and tolerate longer response times for more complex tasks.

Table 4-1, copied in part from [7], shows a possible range of responses for a document-preparation and information-retrieval system. The table lists the approximate response times users can expect for each operation performed by the system. Some operations access both local and remote systems.

Obviously, users should understand that accessing data from a local source should be quicker than accessing the same data from a remote source (assuming the local and remote systems are similar and operating under similar conditions). However, as Table 4-1 shows, this is not to say that network operations necessarily take longer than local operations. For instance, making a paper copy (locally) of a long document located on the local system may take as long as the transfer of a long document over a communications line. The response time for the transfer depends largely on the size of the document, the line speed, and the distance. On a high-speed network, such as an Ethernet LAN, most operations involving the network may happen as quickly as if they were performed locally without network use.

User confusion or dissatisfaction occurs more often in environments where the network is **transparent** (invisible) to users, when they cannot tell whether the commands and operations they use involve the network or not. For example, network access is transparent when user commands that access operations on local systems are indistinguishable from commands that access operations on remote systems. Operations that appear similar in complexity will have different response times. Such irregularity in response times can make the system appear unpredictable and can cause user frustration. Thus, network transparency is not always desirable.

Table 4–1 Response Times

Response time	Task
	Note: Task may or may not involve network operations. An asterisk (*) means that it definitely does.
No visible delay:	Cursor movement on the terminal screen Echoing (the appearance of the character typed)
Up to 1 second:	Procedural informational requests (help, status) Acquiring menus Scrolling, page flipping Detecting syntax errors Displaying document, file, report, and so forth First feedback from a local database search
Several seconds:	Retrieving a single document by content (not keyed field) Making a paper copy of a single-page document Retrieving a document from remote on-line database*
More than 30 seconds:	Archive retrieval External database inquiry* Making paper copies of long documents

Responsiveness is not just a matter of how quickly an event occurs over the network but in what manner. This assumption is the basis for the following rules:

1. Response predictability is as important as average response delay. Users become frustrated when the same operation elicits widely varying response times. For example, users prefer a system that echoes consistently at 50 milliseconds per character over a system that echoes anywhere from 1 to 500 milliseconds per character, even if the latter averages 50 milliseconds per character. The user likes to build a conceptual model of how the system or network works and responds. Variation in response time conflicts with the model and user expectations.

   Varying loads on the network, or on the source and destination systems, can cause unpredictable responses. For example, the response time for an operation performed during low user activity will be shorter than for the same operation performed during high activity.

2. Tasks involving hand–eye coordination, such as editing text or manipulating graphics, require real-time feedback; otherwise, any delay will interfere with the user's coordination. This can pose a special problem to remote text editing. When characters are not echoed locally to the screen, users have to wait for each echo over

the network before they can press the next key. Long delays for each character can hamper the user's productivity. (Local echoing is preferable, for this reason.)

3. Users need some feedback for each action. For example, when a task takes a long time to complete, they need some indication that the task is in process. Human factors studies show that, after two to four seconds, a user starts to wonder if his transaction succeeded; then he becomes distracted by something else.

4. Sometimes a response can be too short. Psychologically, users can feel forced to keep up with the computer. For example, automatic-banking customers who are new to data entry equipment can be intimidated if the screen information comes too quickly, especially if they have no control over the rate at which the screen information is displayed. The ability to control the scrolling of the screen avoids such a problem.

5. A response is good, sufficient, or bad according to how the user perceives it. A user is more tolerant of long delays when he thinks that the task being done involves much work from the computer and communications system. The amount of work is not always understood. Well-written software will cue the user.

6. A response time is reasonable if it does not cost too much to achieve it. While slow response times may increase the cost to *use* the network, fast response times often increase the cost to *build* it. For fast response times, you may need more expensive communications hardware and software. However, some of the network components available today, such as Digital's Ethernet products, are cheaper and faster than their older counterparts.

User dissatisfaction can occur because users do not understand conditions that affect response times. For example, when a function performed over the network takes a long time, users often say "the network is slow." The cause, however, may not be the network, but rather that 50 other users are using the same CPU. A CPU has to divide its time to service the demands of each of the users. Network-related tasks may get less CPU attention than other CPU tasks when user demands are high. This means that the network tasks must wait longer before the CPU services them.

RESPONSE TIMES THAT APPLICATIONS REQUIRE

Response time requirements vary from one application to another. Real-time and interactive applications usually have very high demands. In particular, response times are crucial in situations where a user (or process) has nothing else to do until a response arrives, or when several transactions are waiting for service at the same time.

Table 4-2 lists some applications that typically require short response times. It also describes the pattern of data flow typical for each application.

As the delay increases, the usefulness of a communications service to users decreases because (1) the delay often results in unproductive time for users and (2) user data often becomes less relevant with time.

However, as response time requirements increase, the cost of designing and equipping the network to meet those requirements can also increase. In this light, response time requirements must be realistic and practical.

Table 4-2 Applications Requiring Relatively Short Response Times

Applications:	Data Flow Characteristics:
Information retrieval:	
Credit checking, bank account status, law enforcement, hospital information systems	Relatively low-character volume per input. Response required within seconds. Output message lengths are usually short but can vary widely with some applications.
Source data entry and collection:	
Point of sale system, airline reservations	Transactions arrive frequently (every few seconds) and demand quick response.
Conversational timesharing:	
General problem solving, engineering design, text editing	Interactive (conversational) responses are needed within a second.
Real-time data acquisition and process control:	
Numeric control of machine tools, remote meter and gauge reading, production lines	Remote sensors are continually sampled and monitored at widely varying time intervals.

How to Formulate Response Time Requirements

Applications can be classified into three categories with respect to their response-time requirements:

1. **Applications that are relatively insensitive to delays**, such as batch-type processing or remote printing. Usually, the length of the delay does not affect the value or usefulness of the transferred information.

2. **Applications that can tolerate some delay**, but which work best if the delay is kept under a particular minimum value. Typical examples are interactive applications involving user terminals. Users become dissatisfied when the delay increases.

3. **Applications that cannot perform their function if the delay exceeds a given minimum value**, such as real-time process control applications. Beyond that value, the transferred information loses its value or usefulness.

Some applications do not fit neatly in any one of the above categories. For example, an application may include several functions, some that are highly time-sensitive and some that are not.

To devise requirements for an application, consider various possible response times and rate how useful or valuable each is to the user. Decide what the minimum and maximum tolerable delays are.

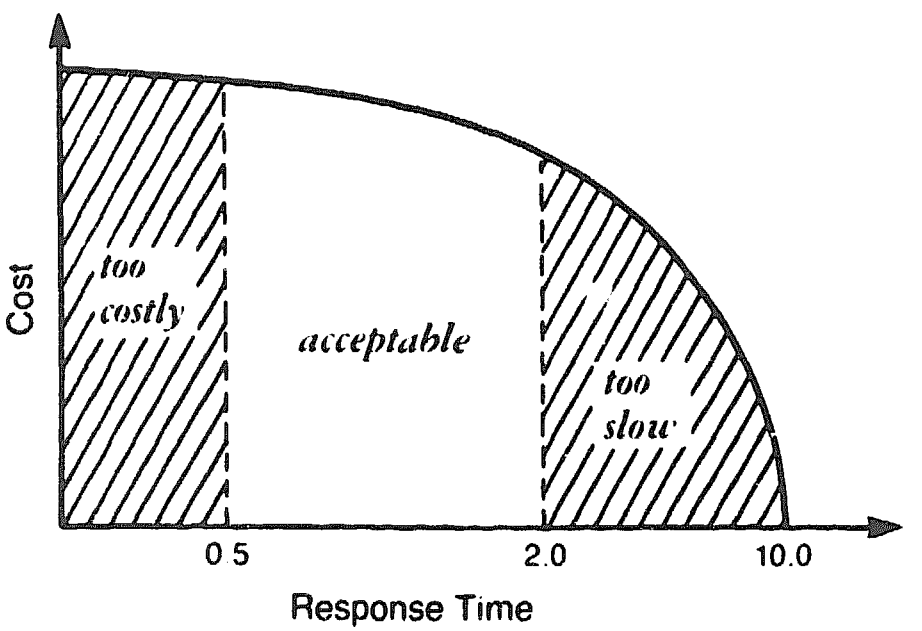


Figure 4-2 Performance Boundaries

Specifying Requirements for Non-Critical Applications

For simple applications that are not critically time-sensitive, ask questions such as:

- What is the maximum acceptable response time? What is the maximum delay that the application can tolerate?
- What is the minimum reasonable response time? What values would provide no additional benefit to the user (keeping in mind that response times can be too quick or costly)?

Take your answers to these questions and define a range of acceptable response times, such as shown in Figure 4-2.

Specifying Requirements for Critical Applications

For complex applications that are time-critical, you can plot graphs showing the usefulness of the network's service as a function of response time. With this approach, you have more than just the upper and lower limits. You also show what you expect between those limits: (1) the *rate* at which the usefulness increases or decreases, and (2) *critical points* where usefulness increases or decreases sharply.

Figure 4-3 includes three such graphs. When specifying application requirements, you can show where on the curve you would like the mean response time to be. The acceptable range depends on the rate at which the usefulness of the application declines.

Graph 1 shows a direct relationship between usefulness and response time. This relationship would apply, for example, in applications where the transferred data is used for process control on a factory floor.

Any increase in response time causes a similar decrease in the production rate of the controlled process. When the response time reaches point "b", the application or information becomes useless or too costly, such as when workers or customers have to wait around with nothing to do.

In Graph 2, the usefulness of the network decreases more rapidly as the response time increases. This relationship is typical of applications where the transferred data quickly loses its value as the delay increases, such as in aircraft traffic-control monitoring. (Usefulness is zero when the delay is dangerous.)

In Graph 3, usefulness is relatively unaffected by any response times less than the critical point, "c". But, as the response time approaches the critical point and passes it, the usefulness of the network decreases sharply. This relationship might apply in inventory control applications where each outlying store accumulates transactions and transmits them to a central office at the end of a business day. The delay to the central office is insignificant as long as the data reaches there before 6:00 P.M. (the critical point), when the central computer begins processing the data.

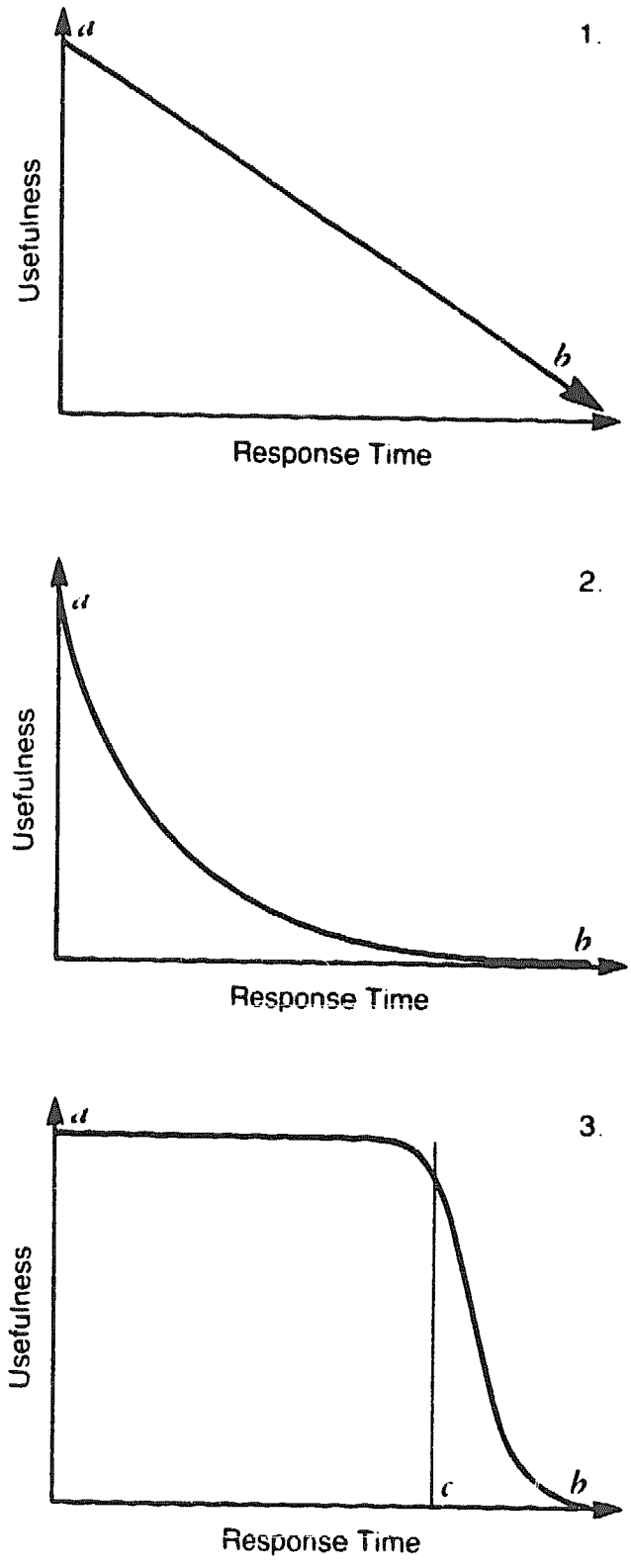


Figure 4-3 Response Time and Usefulness

Figure 4–4 illustrates intervals where usefulness "comes back to life." For example, the usefulness revives when the response time becomes long enough for the user to turn to another task and complete it while waiting for the first response. The user switches back to the first task when its response appears.

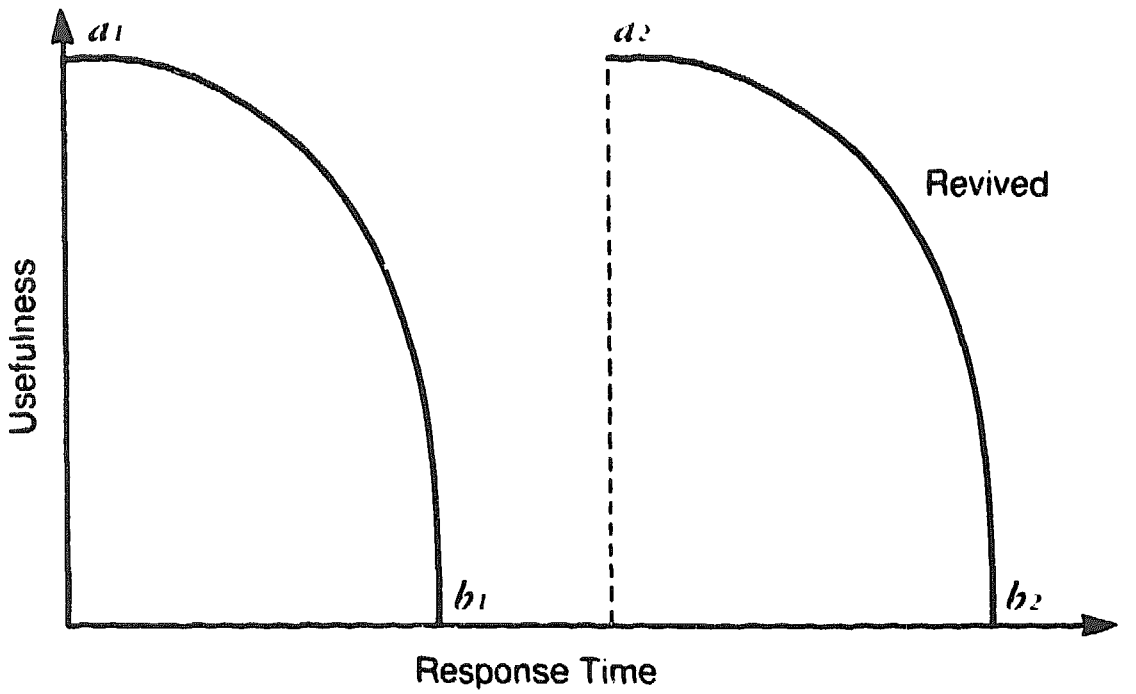


Figure 4–4 An Application Where Usefulness Is Revived

Applications with Conflicting Demands

What about a network used for a variety of applications, each with unique demands? In cases where none of the applications have tight demands, the majority rule may suffice: The responsiveness of the network is satisfactory if the demands of the majority of the applications are satisfied the majority of the time. In many cases, this is not enough.

Usually one or more of the applications do have critical demands that must be met at all times and costs. In some cases, the entire network must be designed or tuned to meet those special demands.

A cost-effective approach would be to set up the network so that critical applications have priority as needed, rather than to invest indiscriminately a large amount of the network's

resources for all applications. For example, you might dynamically allocate more resources to these applications when they need it, and then return the level of resources to a normal state when the special needs pass. Two applications requiring considerable resources at different times of the day could share the extra resource reserve alternately when needed. This saves the expense of having two separate reserves of resources, one for each.

Dynamic bandwidth switching for data links is another possibility. Here, network software automatically allocates greater bandwidth to links as the need arises. (**Bandwidth** reflects the information-carrying capacity of a channel or line, usually expressed in bits per second.)

HOW TO MEASURE AND EVALUATE RESPONSE TIMES

The most important considerations when evaluating response times are as follows:

1. Take a valid sample of times to give a complete picture of the responsiveness of a network.

2. The most accurate way to measure response times is under actual workload and traffic conditions. However, because of the variability of the real workload over an extended time, it may suffice to sample the response times during the peak load and during an average load.

3. To make valid comparisons, you must know the conditions under which you make each time measurement.

4. You must measure response times in a consistent manner.

5. To give the most objective measure of the network's part in a transaction, you should measure response times when no other tasks are competing with network tasks for system resources.

Measuring Response Times

It is best to measure response times with automatic measuring equipment, software tools, or your own command file. For example, to measure file transfers with a command file, you can create a file of just three commands such as the following VMS commands:

```
SHOW TIME
COPY LABDOR::TEXT.LIS BARGIN::TEXT.LIS
SHOW TIME
```

The COPY command transfers a file named TEXT.LIS from node LABDOR to node BARGIN. To determine the response time, you take the difference of the two times recorded at your terminal by the SHOW TIME commands.

Some software programs may be available for measuring response times. For example, Digital's DTS/DTR utility is available on most Digital systems (which run DECnet) to measure the performance of a variety of operations. This utility has the advantage of allowing you to measure the effect of varying message sizes and other parameters.

Measuring Access Time

The access time is the average time between issuing a request for communications and the actual start of the data transfer. You begin measuring access time when the request to the network is made at a source node. You stop measuring when the first message reaches the destination. Do not include unsuccessful accesses in your average.

Measuring Network Delay

You measure network delay in terms of the block transfer time. **Block transfer time** is the time required to transfer data from the source to the destination and back, excluding the time taken by the destination node to generate a response. A **block** is any contiguous unit of user information grouped together for transmission, such as the user data within a packet (excluding the protocol overhead).

More specifically, the block transfer time begins when a block has been input to the local system and the transmission has been authorized; it ends when the block has been output to the destination user. (In applications where the user must be notified of the incoming block, the block transfer time ends when the user is notified.) Block transfer time is especially relevant for file transfers.

An Example of Measuring Network Delay

As an example of measuring network delay, consider an inquiry/response application:

1. Measure the response time for an inquiry accessing information locally (on the system to which your terminal is connected). Assume the terminal is directly connected.

2. Measure the time for the same inquiry accessing information on a remote system connected by the network.

3. Take the difference between these two response times. That difference is the network delay (this result includes the time for accessing the network, plus the time for transferring the request and response across the network, plus the time for accepting the request and response from the network).

To be valid, (1) the local access must not involve network software, (2) the local and remote systems must have the same type of CPU and capacity, (3) the workloads on each system must be equal, and (4) once the remote system is accessed, then accessing the information on it must take the same time as accessing the information locally.

To best meet these conditions, you should test between two equivalent single-user, dedicated systems. On these systems, workload conditions are more stable and simpler to control and evaluate. This is a common way to measure response time for network applications.

If the above method of measuring network delay is inapplicable (for example, when the application cannot be run both locally and remotely), what other methods are there? Technical support personnel commonly use hardware monitors. These monitors can mark the first instruction executed by the network module for processing a query and the last instruction for receiving the response.

Estimating Response Times

The following formula lets you estimate the response time for transferring a file of a specified size:

$$R = A + (P \times S)$$

where:

R = Response time

A = Access time (for example, time required for the network to establish a connection between the source and destination nodes)

P = Seconds per block for processing, disk access, and transferring the file over the link

S = Size of the file in blocks

To find A, measure the time to copy a file of 0 blocks (use a command file like the example in the section on **Measuring Response Times**).

To find P (seconds per block), perform the following steps:

1. Measure a number of file transfers of various sizes and plot a graph of the results (as in Figure 4-5), showing response time as a function of file size

2. On the graph, choose two files of different sizes (s1 and s2) and mark their respective transfer times (t1 and t2).

3. P equals the difference in transfer times (t2 - t1) divided by the difference in file sizes (s2 - s1):

   P = (t2 - t1) / (s2 - s1)

Often the seconds-per-block value is the most important consideration, such as in applications where the access time is relatively insignificant (for example, where a commonly-used remote system is accessed once, and access remains open throughout the day). By knowing the transfer time for a block, you can easily estimate the transfer time for file transfers to and from that remote system.

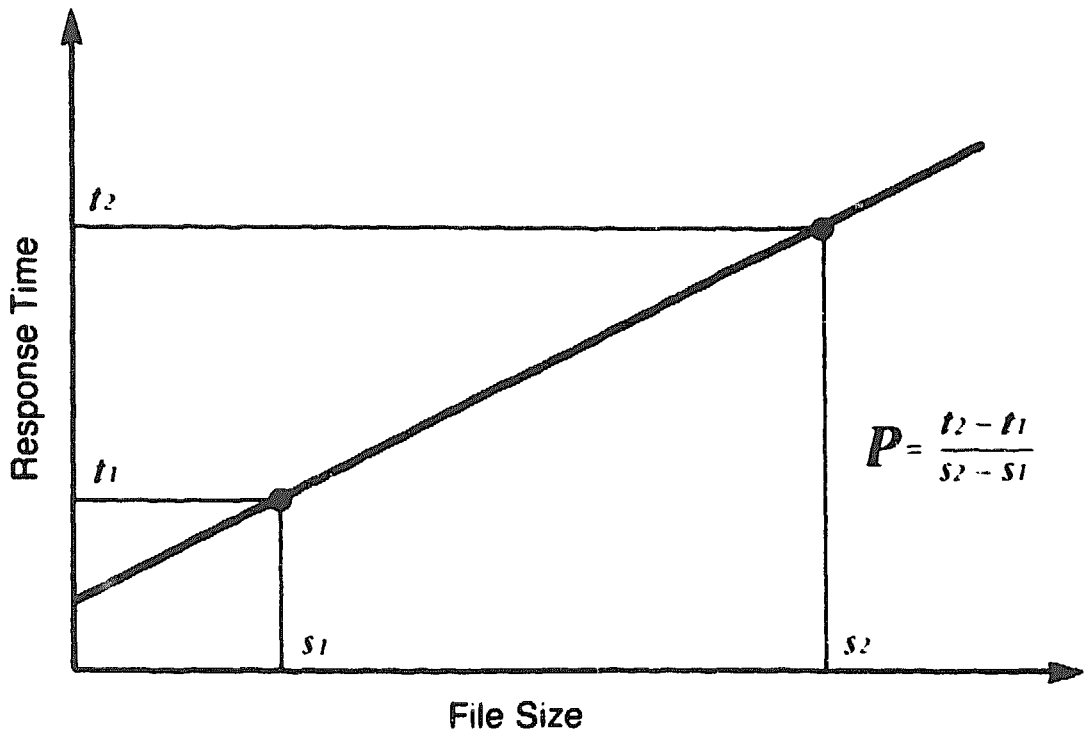


Figure 4-5 Finding Seconds per Block

FACTORS THAT AFFECT RESPONSE TIMES

As shown in Figure 4-6, the main factors affecting response time originate in three areas:

1. The local system (the source of the transaction)

2. The network (the transmission media)

3. The remote system (the destination of the transaction)

The shaded regions in Figure 4-6 show that certain influences on response times are shared by more than one area. The transmission time (t) is the only influence for which the network is purely responsible.

Generally, response time depends on the characteristics of the components that handle the communicated data in each of these three areas, the workloads on the source system and destination system, and the traffic on the network.

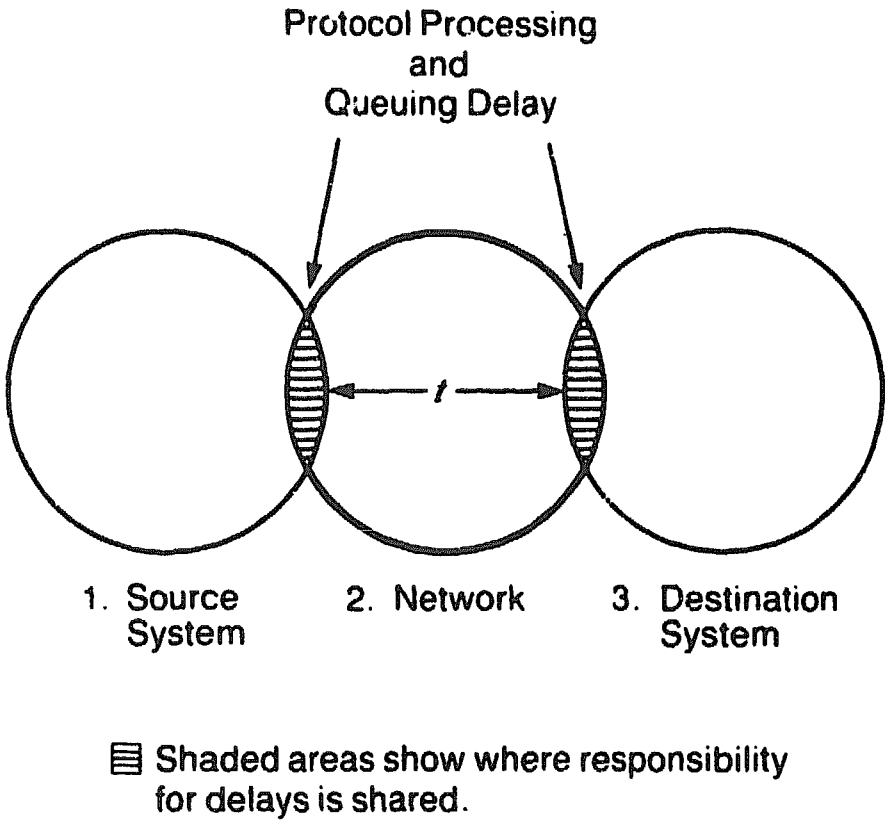


Figure 4-6 Sources of Delay in Network Communications

Factors at the Source and Destination

Numerous factors at the source and destination systems affect response times. Those discussed here are listed below. Those not discussed are memory access and priority schemes, memory access speeds, and application design.

1. Capacity and availability of CPU time
2. Type of system
3. Terminal line speed at the source node
4. I/O processing capability
5. Buffering
6. Message size

Capacity and Availability of CPU Time

Perhaps the greatest factor on any system that handles network communications is how much of the CPU is available for the network. Obviously, a small CPU has relatively less capacity for handling network activity than does a larger CPU.

Consider as well the workload on the CPU: a CPU with a heavy workload has relatively less capacity (or time) available for handling network activity than does an equivalent CPU with a lighter load.

You should select the CPU(s) with the right capacity to handle your application needs. Also, consider upgrading CPUs or clustering CPUs to provide more capacity, memory, reliability, and so forth. Consider moving some of a stressed CPU's workload to another node that can handle it without detracting from the performance of network applications. Have analysts check the application software to make sure it is designed to make best use of the CPU's capacity.

Type of System

The type of system is important, such as whether it is a timesharing system or dedicated system. A dedicated system usually delivers faster response times because network resources do not have to compete for CPU time as is the case with timesharing systems.

Timesharing systems must share the resources with a group of users or tasks. The "weight" of the workload depends on the number of users *actively* using the system and the number and nature of the tasks being processed. Users or tasks require CPU time. More active users and more complex tasks require more CPU time. Therefore, because network tasks must compete for CPU time, they will have to wait longer for attention from the CPU. This increases response times.

The architecture of the system also matters. It determines how much time is given to communications processing.

You should make sure the operating system on your node is appropriate for the tasks that it must perform. If high performance is necessary, you may choose to install a dedicated system. On a timesharing system, you can have the system manager limit the number of users who can use the system at any time, or you can restrict the use of the system to favor performance of network-related activities, especially during peak activity.

A system manager can improve the performance of applications that use the network by configuring dedicated communications systems (front-end processors or router servers) to handle communications processing for the computer. A dedicated communications system performs network tasks more effectively, because it is not interrupted by other tasks. Also, if the timesharing system is freed of communications responsibilities, it performs local tasks faster.

Terminal Line Speed

Another factor that affects response time is the transmission rate between the terminal and CPU at the local system. For example, if the transmission rate is only 300 bps, you have a transmission rate of 30 characters per second (assuming asynchronous transmission with each character consisting of eight bits plus a start- and stop-bit). It takes 10 seconds to transmit a 300-character message. (The ensuing transmission from the local system to the destination across the network might only take one or two seconds.) If you used a terminal with a transmission rate of 4800 bps, you could transmit the same message in less than one second.

I/O Processing

With respect to I/O processing on the systems, the issues are:

1. How long does the application software at the source take to process the request and get it to the network?

2. How long does the corresponding software at the destination take to process the request and generate the response?

Applications can employ pipelining and multibuffering techniques to speed up I/O processing. **Pipelining** is a technique in which several stages of a process are handled simultaneously rather than one stage at a time. **Multibuffering** is a form of pipelining in which two or more buffers are provided at a time to handle data. Single-buffer processing provides only one buffer at a time, causing subsequent operations to wait until that buffer is available.

In a small memory machine, however, multibuffering could degrade performance. Multibuffering uses more of the available network buffers than single buffering. A

program's buffering level can actually slow down the CPU and increase user response time. Also, multibuffering is more difficult to program.

The application at the destination node often imposes the largest delay, such as in inquiry/response systems where a short request generates a large amount of output as the response.

The destination system can impose a large delay because of a large workload. For example, suppose you use the VAX/VMS SET HOST command to connect your terminal to a remote timesharing system that is being used by a large number of users. You may find that the connection is established quickly, but once you log in to the remote system, response times are slow.

You should have analysts check the application software to make sure it handles I/Os efficiently. See if higher buffer levels are warranted. If I/O processing is a bottleneck, check whether you can offload the I/O processing onto dedicated servers (in a LAN) or front-end processors (in a WAN).

Buffering

If a terminal has smaller buffers than those used by the application, long delays can result when the application sends large amounts of data for display at the terminal. Each time the terminal runs out of buffer space, it has to request the application to stop sending until space is available.

Larger buffers bring more efficient communication and higher throughput, regardless of the relative sizes of the buffers at the sending and receiving ends. Thus, in many cases response time improves with larger buffers. However, if system buffers are too large, especially when a high number of messages is sent one-per-buffer, the local system's resources can be used up quickly. When this happens, the CPU may have to delay processing of tasks until resources are available. On some systems, when buffer space runs out, an error condition results which requires error recovery procedures that add to the CPU load and processing time.

The system manager should make sure that the sizes of buffers used for communications are appropriate for the applications and that enough memory is available for those buffers. Buffer sizes should be selected according to the average size of user messages.

Message Size

A small message takes less time to process and transmit than a large one. However, in some systems an interrupt is necessary for each message. Thus, smaller messages can mean more interrupts and greater delays.

Small message sizes are not conducive to quicker response times if there is a large amount of data to send. It imposes a larger delay than larger message sizes, since there are more messages to process.

Note that DECnet segments larger messages and, therefore, maintains quicker response times.

There are many types of messages, and they differ from each other in terms of their demands on resources. The average (mean) size of all messages sent affects the overall performance more than does the size of any one type of message. (This assumes the frequency distribution of message sizes is a normal curve.)

User messages are usually fixed, so you have to work around them to optimize performance. However, application software can be designed to block messages for optimum network performance, especially where messages are sent in bursts.

Factors at the Network Level

Major network factors that affect response time are (1) protocol processing (2) traffic intensity and (3) transmission time.

Protocol Processing

The time needed for protocol processing is spent mostly ensuring reliable data transmission: detecting and correcting bit errors, acknowledging correctly received data, and retransmitting unacknowledged or erroneous data.

A common carrier can condition dedicated lines to provide higher transmission reliability. (**Line conditioning** is a technique that reduces noise and thereby improves the capability of the line for more reliable transmission at higher data rates.) Where transmission problems exist and line conditioning is not possible, you may require technical experts to analyze the problem.

Traffic Intensity

One of the key factors affecting response time is the workload (traffic) on the network, the workloads at the source and destination nodes, and the workloads at any intervening (routing) nodes.

Response time is most frequently measured and evaluated as a function of system workloads and network traffic. Figure 4–7 shows a sample graph of the relationship between traffic and response time. As the traffic increases, so does the response time. One of the best ways to find the ideal workload is to test the network under various traffic conditions. Then adjust the traffic flow to maintain that ideal level.

As Figure 4–7 shows, response time usually stays within an acceptable range, increasing slowly until the network cannot handle any more traffic, or until a critical component reaches capacity. At that point, response time increases rapidly. (The knee of the curve is the critical point. In a good applications design, traffic level should not exceed it.)

The level and nature of network traffic determine the length of queues within the network. Think of a network as a series of queues. Each network module or component has at least one queue. Within the network software on each node, queues exist for the different layers. For example, there is a queue for the physical layer (the line) and one for the data link layer. While the data link layer is checking the validity of data presented to it from the physical layer, subsequent data from the physical layer may have to wait momentarily in a queue there.

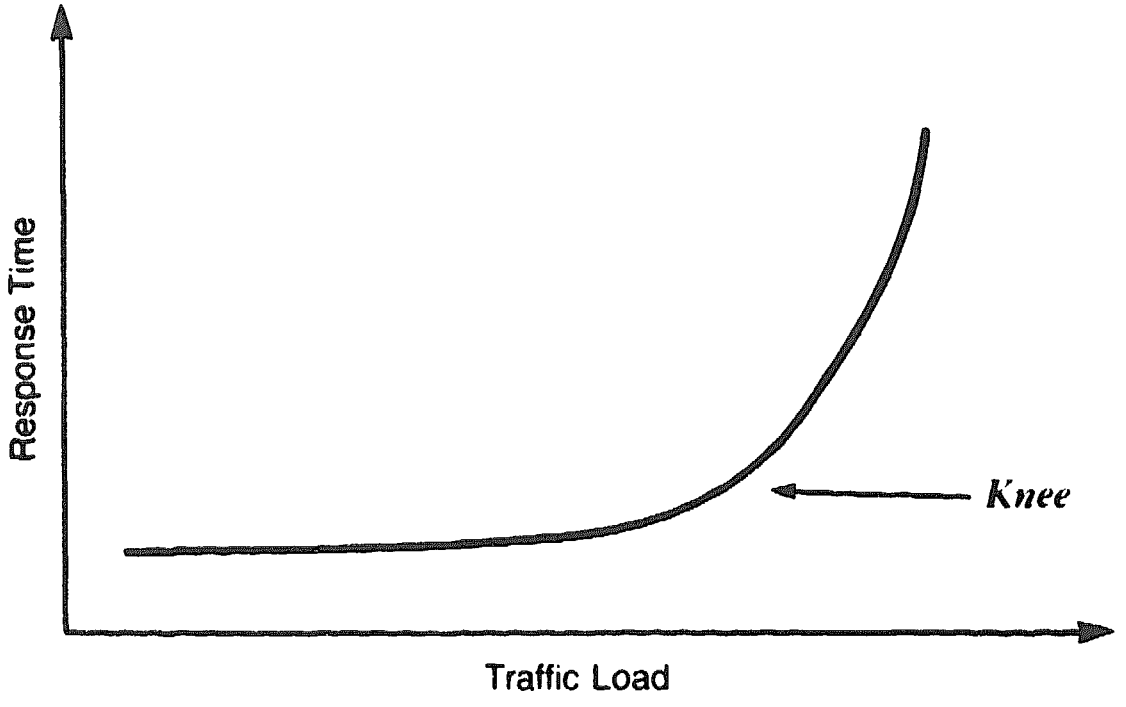


Figure 4–7 Evaluating Response Time Against Traffic Load

The delay at each queue depends largely on the workload of each component, the ability (capacity) of each component to process that workload, and the rate of incoming traffic or requests.

As shown in Figure 4–8, you could think of the line at a bank as a queue. The wait in line depends on:

- The complexity and amount of work for the bank tellers.
- The rate at which tellers perform the work.
- The rate at which customers enter the bank for service.

Notice that the latter factor (incoming traffic) depends on the output of the adjacent component(s) preceding it. The passenger bus, sidewalk, and street are the bank's adjacent components. How much traffic they deliver affects the business of the bank.

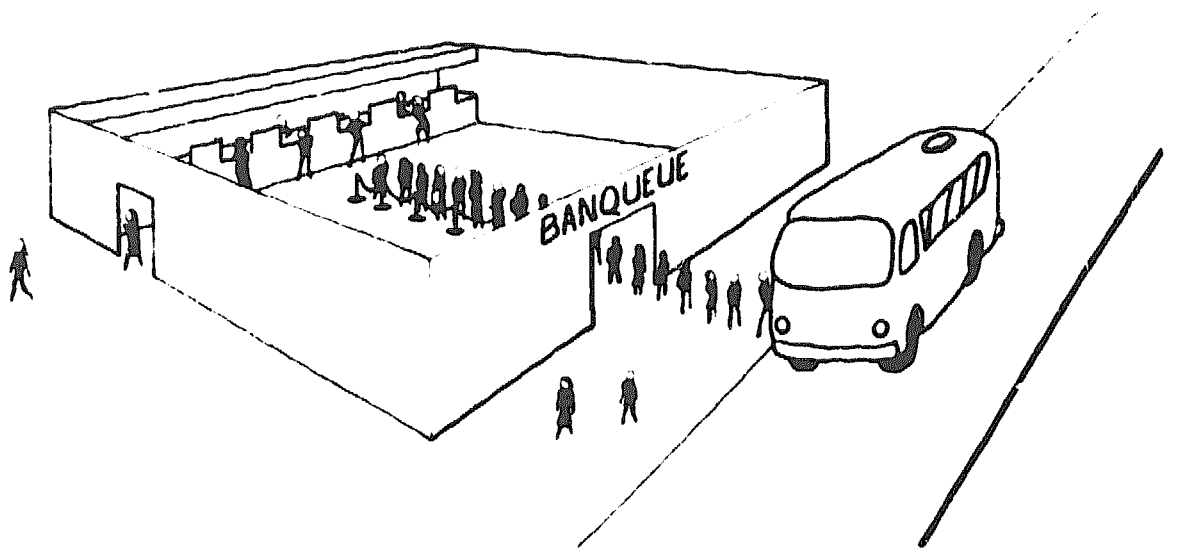


Figure 4-8 Queuing Delay

You should monitor the network to see if it is configured appropriately to handle the workload. As you identify bottlenecks, increase the bandwidth or processing capacity where needed. You can also redesign applications to redistribute traffic and relieve the workload on stressed parts of the network.

Transmission Time over the Network

Transmission time can have a significant effect on response times, especially in transmissions over long distances. In Figure 4-9, the starting point for measuring the transmission time of a packet is the moment the first byte enters the network medium; the ending point is the moment when the last byte of that packet is received by the destination.

The transmission time depends on the speed of transmission, the size of the data unit, the distance traveled, and the conditions of the transmission media.

Propagation Time

Propagation time is part of the transmission time. It is the time a signal takes to travel from one point to another in a network. As shown in Figure 4-9, its starting point is the

same as for transmission time, but the ending point is when the *first* byte of the packet reaches the destination. (The transmission time accounts for how long it takes to transmit a packet, as well as propagate it.)

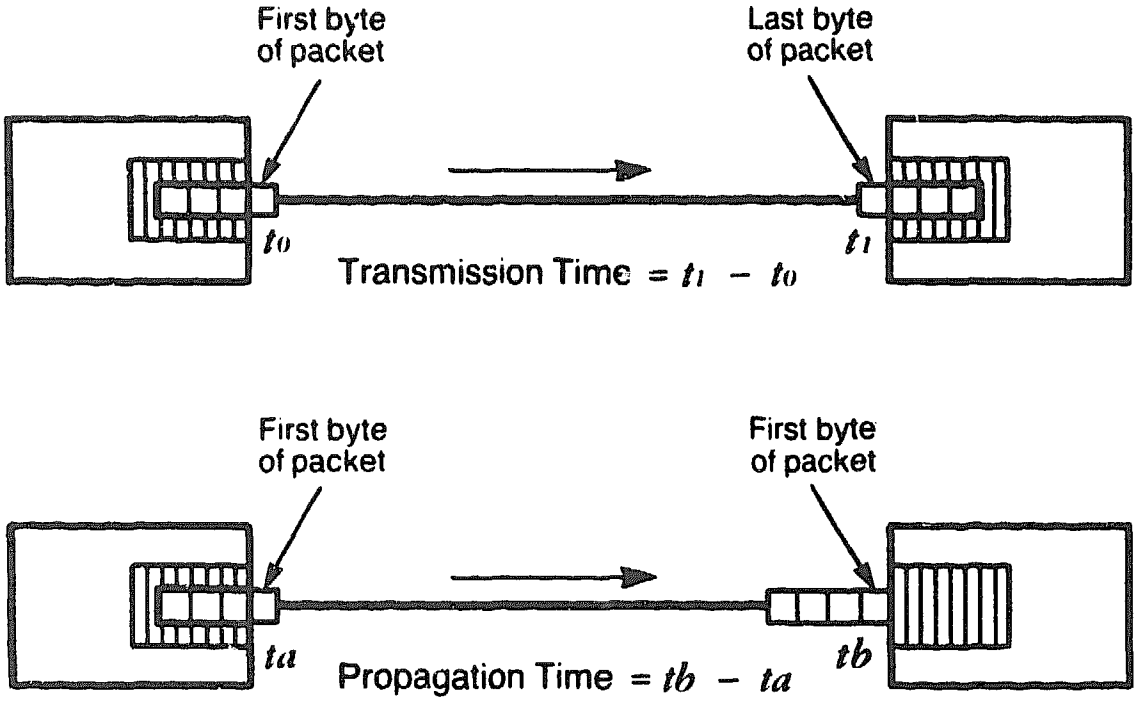


Figure 4-9 Transmission and Propagation Time

Queuing delays along the way increase the propagation time. A major part of the propagation time can include queuing or buffering delays at such devices as multiplexers and concentrators. Also, synchronous modems on voice-grade telephone lines impose minor delays.

Forwarding Delay

The propagation time also includes the forwarding delay at one or more store-and-forward nodes on the link. As traffic builds, more data has to be stored at each node before it can be forwarded. Thus, the forwarding delay increases. For this reason, design and configuration of networks must be such that store-and-forward nodes do not become bottlenecks. (Though routing nodes in DECnet networks do not impose a significant delay, they require proper configuration to avoid bottlenecks.)

Bridges, devices used to connect two LANs, impose a slight forwarding delay for transmissions from one network to another. However, bridges such as Digital's LAN Bridge 100 improve performance for traffic between LANs. The LAN Bridge 100

manages inter-LAN traffic and selectively forwards packets to keep local traffic local. Only data destined for different LANs passes through the bridge and continues on to the appropriate remote destination. Thus, the bridge improves performance by reducing the total traffic on each LAN.

Other Network Characteristics That Affect Response Time

Several other characteristics of the network that may influence response time include:

- The size of the network
- The complexity of the network topology (how many nodes must be traversed) and how conducive it is for the application
- The capacities of the routing nodes
- The size and number of buffers used for messages
- The transmission media and speeds
- Errors and retransmission of data incorrectly transmitted
- Switching characteristics

The relative weights of these factors depend on whether the network is a LAN or a wide area network (WAN). For example, protocol processing has more importance in LANs. Bottlenecks are more likely to occur because of the delay for protocol processing than for that of transmitting across high-speed LAN links.

Size and Complexity of the Network

Distance has more impact for WANs than for LANs because WAN transmissions usually travel a greater distance. Note, however, that the transmission times for WANs are not necessarily longer than those for LANs. For example, WANs using high-speed lines (such as 56 Kbps) may have relatively smaller transmission times.

The number of nodes affects both WANs and LANs. As the number of nodes increases, traffic usually increases. Heavier traffic increases the load on network components, such as on routing nodes or store-and-forward nodes. It also increases queuing delays, such as for accessing the network.

Usually you have no choice about the size of the network. The size is governed by user demands and the environment. However, you can have a network divided into smaller, easier to manage "sub-networks," called areas in DECnet Phase IV terminology. Within each area, level 2 routers are responsible for routing data between areas. Level 1 routers

are responsible for routing data only between nodes within the area. With the inter-area routing overhead restricted to level 2 routers, you have a reduction in routing traffic and a more efficient network.

Buffering Level

The **buffering level** refers to the number of buffers that the network software provides at one time to handle data. With a multibuffering system, a network can send several buffers of data in quick succession between nodes. Figure 4-10 shows three stages of a process being handled simultaneously. The system segments the process so that several resources are used at once. Note that for multibuffering to work, sufficient memory must be available on the node.

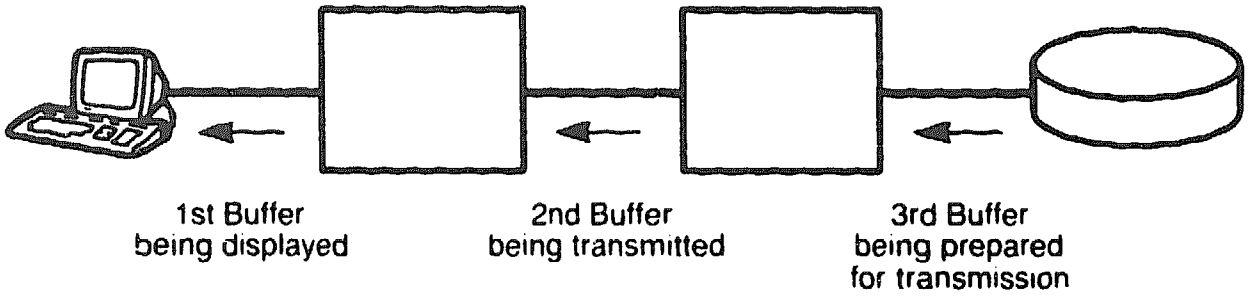


Figure 4-10 Multibuffering Can Get the Job Done Quickly

In a single-buffer system, response times can be longer. Before the system can send the next buffer, the source node has to wait for the destination to acknowledge that it has received the previous buffer.

Programmers can take advantage of higher buffer levels by employing multibuffering in user application programs. For higher efficiency throughout the network, system managers of the different nodes should ensure that they all use similar buffering levels. If, for example, one node allocates eight buffers for the network while another allocates two, the former node is wasting considerable memory (that required for six unused buffers).

Transmission Media

Another major factor at the network level is the transmission media. For example, satellite transmissions have long round-trip delays, telephone lines can have high error rates, leased lines can have high transfer rates, and so forth.

Another factor is transmission speed. High speeds bring faster response times (this assumes all systems or devices receiving the incoming data are fast enough to handle it). A link with heavy traffic requires high-speed transmissions, or else it can become a bottleneck.

One final consideration is whether a link is full-duplex. Full-duplex provides faster response times by allowing the source and destination nodes to communicate in both directions simultaneously. It also avoids the modem turnaround time required for half-duplex transmissions.

Switching and Connection/Disconnect Time

In circuit-switching networks, you may consider the delay required to connect or disconnect a link. The effect of this delay on response time is important at least for the first of a series of transactions over the circuit. Once a circuit has been established, connection time is no longer an issue. Connection time becomes an issue mostly for users who must make frequent, brief accesses to the network. To quicken the connection time, you can use automated dialing (the computer initiates the call).

The **disconnect time** is the time required to disengage a connection; that is, the delay from when the disengagement is requested to when it is completed.

When you use a network to connect your terminal to a remote node, such as by using the virtual terminal facilities on DECnet (SET HOST on VMS, for example), connect and disconnect times can play an important part. Numerous factors can affect the delay for each, such as the workload, the number of concurrent sessions at your node or at the remote node, the number and characteristics of intervening nodes, and so on.

IMPROVING RESPONSE TIMES

"Appropriate response time can be one of the hardest human interface requirements to achieve. Because technology and available resources are frequently cost sensitive, designers need good information on how users react so that they can argue for the resources to make the system fast in the right places and can save money in areas that don't demand high performance."[7]

Always Weigh Your Choices Against Cost

Numerous options are available for improving the response times seen by users of network applications. Often the first choice is to upgrade network hardware and resources. Usually, though not always, this option requires the heaviest investment. For this reason, you should first explore more cost-effective possibilities, such as the following (beginning with the cheapest):

1. Stagger or adjust the load

   If responsiveness degrades considerably during peak hours, see if any of the processes generating network traffic can be done at other, less busy times. If a particular point in the network is becoming a bottleneck, see if traffic can be redirected over other routes.

2. Tune the system(s)

   Tuning requires expertise to analyze the network's performance and diagnose problem areas. Tools may be available for helping tune systems, such as Digital's VAX/VMS System Performance Monitor (which you can purchase from Digital).

3. Tune the application

   Does an application use multibuffering for I/O to and from the network (where system memory permits)? Does it allocate the optimum size and number of buffers? Digital provides a tool called the Performance and Coverage Analyzer for this purpose.

4. Distribute processing

   This is perhaps the most effective option. The idea is to move the time-sensitive processes from a central location to the areas where the users need the output. This helps ensure that processing will take place in real time. Also, by making most of the processing local, you reduce network traffic.

   To improve responsiveness on large CPUs, you can confine text editing to PCs in the network. Text editing requires a CPU interrupt for each character of input. By moving text editing to PCs, the larger CPUs are free to process larger jobs or tasks, such as transferring files or filing memos.

To meet user requirements in real-time or interactive applications, the network must always be available to handle the traffic. To ensure availability, redundant paths and hardware may be necessary. If one path or component fails, then another will be ready as a backup. You must determine whether the performance demands are worth the costs of purchasing the extra equipment needed to provide redundancy.

Also, it would be a waste to inves in an improvement that goes unnoticed. For example, it is not practical to spend an enormous amount of money to get a response time of one second when a three-second response is acceptable. Below a certain threshold (about 150 milliseconds), users cannot even perceive differences in response times. *The test is not whether there is a noticeable delay, but whether the delay impedes the user's performance.*

Study How Users React

Perhaps one of the least costly options for improving response time is studying the user's reactions to the responses and determining how the *perceived* response time can be reduced.

For example, when responses are long, the perceived response time can be reduced by programming the application to display a message periodically at the user's terminal indicating that the transaction is being processed. Without such notification, the waits appear longer than they are.

Another important consideration is how a delay impacts the user's productivity. "Sometimes users make good use of waiting time to plan their next task, or to talk to a client, and the impact of waiting is minor. The challenge for designers [or managers] is to know what it is worth to speed up a system. Often the decision requires a tradeoff between resources and costs versus productivity. One approach is to ensure that the user can always do something useful while waiting for something else."[7]

Good application design can enhance a user's productivity. In a transaction involving a series of operations, those operations that require long delays should not be interspersed (if possible) among the interactive operations. For instance, a data-entry application should ask the user for all the data first, and then update the database, rather than updating the database repeatedly after the user inputs each field of data.

Make the Network Fast in the Right Places

By studying user reactions, you are more likely to ensure improvements are made in the right places. Often managers waste time and effort improving the network where improvement is not needed. Or, they make improvements without considering the consequences to other parts of the network. Increasing the capacity of one component can increase the load on another part of the network, forcing one or more components there to become overloaded.

For example, a group of users need to use terminals to display large amounts of data accessed from a remote computer. The manager chooses 9600 bps lines for connecting the terminals to the computer, assuming that the faster lines will ensure better performance. However, the resulting performance is much poorer than expected. Users are dismayed by response times that vary significantly, some too long to be acceptable. The manager tries using even faster lines, but the results are even worse. A consultant suggests slower lines. When 2400 bps lines are used, response times improve.

The consultant explains why the faster lines bring slower and less predictable response times. At 9600 bps or faster, the data is transferred to the terminals too quickly. The terminals request more data from the computer that much sooner. This puts a greater

demand on the computer (1) for processing the I/O, and (2) for interrupt processing during the I/O.

First, the CPU has to work faster to prepare data for transmission to the terminals (thus completing more I/Os in a given time). Second, the faster lines force the terminals to send more interrupt requests to the CPU. The terminals cannot handle data as fast as the CPU, so they have to repeatedly stop the CPU from transmitting until they are ready for more data. Interrupt-processing places a heavy burden on a CPU. So, changing the speed to 2400 bps reduced and distributed the overall load on the CPU (the load contributed by the terminal requests).

Often the hardware is not the source of the problem. The right place to make the network fast can be in the application itself. When response times are important, applications should be designed with response time in mind. For example, if you expect a high rate of traffic between two points, the applications there should provide ample buffers to handle the traffic; otherwise, delays waiting for buffer space will lengthen response times.

As discussed earlier, you can use multibuffering in applications to reduce delays. The initial response comes sooner to the user, giving the advantage of improving the perceived response time.

Remote terminal applications, such as remote text editing, can be designed so that editing occurs a page at a time, rather than line-by-line. This reduces I/O processing delays and the number of interruptions for the terminal user.

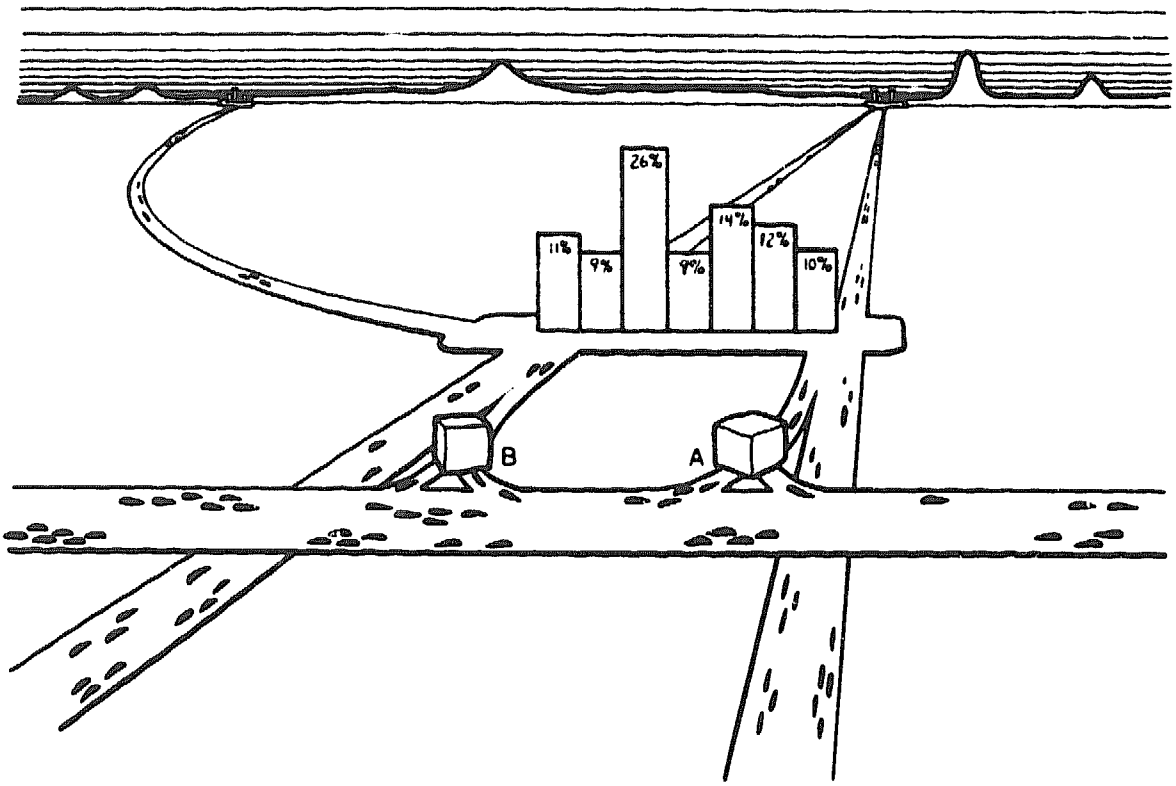
Finally, if the delay is because of excessive retransmissions over an unreliable line, then you can repair or replace the line. If you have to live with the line as it is, then you can change the application to transmit smaller messages over the line.

SUMMARY

Although response time is the most familiar and commonly used index of network performance, it often does not fairly represent the network's performance. Response times depend on numerous factors, many of which are *not* related to the network. Even with the best designed network, you may see poor responsiveness, such as when one or more CPUs are bogged down with heavy local-processing workloads.

Response time is a relative or conditional index of network performance. It is also subjective, depending on the user's knowledge and experience, and on the situation or application. For these reasons, response time is not the most reliable measure of a network's performance. The measure of available bandwidth or throughput usually tells more.

5 THROUGHPUT



By understanding and monitoring throughput, you can help ensure that the network is able to handle all the traffic generated by the applications.

The preceding chapter showed that response time can be an ambiguous and subjective index of a network's performance. Chapter 5 shows how throughput, when clearly defined, is a more useful and objective measure of performance than response time. This chapter also shows how throughput measurements can be misused or misunderstood if they are not clearly defined.

AN OVERVIEW OF THROUGHPUT

Throughput measures a network's ability to transfer data from one point to another. It is the total amount of data communicated during a specified unit of time.

Contrast this definition of throughput to response time, which measures the speed of a network in taking a specific request and returning the corresponding response. Throughput measures the network's ability to handle any data, regardless of the application or direction of traffic. It emphasizes quantity or capacity rather than speed, though it does depend on speed. For this reason, throughput is usually more important in applications involving large transfers of data.

In one sense, throughput is a measure of productivity. In fact, it is a more direct measure of productivity than response time. (Responsiveness does not necessarily imply productivity.) As such, throughput answers such questions as: How much work can be done over the network in a given amount of time? How many jobs or transactions can be completed in a certain amount of time?

In another sense, throughput is a measure of the *ideal, theoretical,* or *potential* volume of work or traffic, as distinguished from the *real* (measured) volume. Here throughput is expressed purely in terms of data units, rather than work units (for example, bits per second, instead of jobs per second). In this sense, throughput is synonymous with bandwidth. It tells you the theoretical **maximum transfer rate** of the component, disregarding any of the limiting factors that play in real life. Here, throughput is also synonymous with the **line speed** or **signaling rate** of a communications line. (In broadband media, this applies to a single channel only, each channel having its own bandwidth.)

In the first sense, "high throughput" means the system is pushing data through the network at a high rate. In the latter sense, it means the system *can* push data through at a high rate.

When discussing throughput, two major distinctions are important. The first distinction differentiates the rated, theoretical throughput from the real, applied throughput.

1. *Rated:* the potential or bandwidth of a component or network (at the physical layer).
2. *Real:* the achieved or measured performance of an application (at the end-to-end layer of the network).

The other distinguishes between gross throughput and net throughput:

1. *Gross:* the total amount of data passed in a unit of time, called the **total throughput** or **gross throughput.**
2. *Net:* the total amount of *user* information (usable data) passed in a unit of time, called the **user throughput** or **net throughput.**

Appendix C discusses another distinction important for understanding throughput: the difference between baud and bit rates.

Rated Throughput Versus Real Throughput

Rated throughput is similar to the capacity of a particular part of the highway system (for example, the capacity of the highway between points A and B).

In contrast, real throughput is the average number of cars traveling between points A and B in a given time. Real throughput depends not only on the capacity of the road between points A and B, but also on (1) the number of cars that happen to be traveling from A to B during the measured time interval, (2) the rate at which they travel, and (3) the rate at which they enter the highway (the number of cars per unit of time), which is conditioned by the rate at which the toll booth processes cars.

In network terms, rated throughput measures the performance capacity or total bandwidth. The rated throughput of an Ethernet is 10 Mbps. This capacity is not the same as the net rate at which the network actually transfers information. This capacity does not consider the overhead.

In contrast, real throughput measures the actual processing/transfer rate of the user information and includes the negative effect of the overhead. This is the net rate, sometimes called the **transfer rate of information blocks (TRIB).** It is less than capacity because of the time that the application and the network require to prepare the data for transmission and because of the speed of the hardware devices. Also, the protocol overhead and errors in transmission (requiring retransmission) help reduce real throughput to levels lower than the rated throughput.

Several other factors limit real throughput: (1) the rate at which traffic is generated at point A (the source node), and (2) how quickly the traffic is absorbed at point B (the destination node). This, in turn, depends on how the application is designed to use the network, such as:

- The buffering scheme
- The application protocol
- The application's structure

Thus, the real throughput is associated with the application layer of the network. It measures the amount of data transferred as a result of the application. The rated throughput is associated with the physical layer of the network. It measures the theoretical (physical) capacity of the component, regardless of the amount of data actually transferred when the application is run.

The real throughput also depends on the characteristics of the hardware and software used in the network, such as the available CPU power at each node, the bandwidth of the transmission media, the network protocol, and so forth.

The upper limit for throughput at the application level is determined by the component or process with the least bandwidth or processing capacity. This is the bottleneck. Where multiple paths are available between two end users, then the actual limit is the slowest component or process on the best path. (The best path is that which the algorithm or user selects, based on the relative conditions or parameter values of the paths.)

Look, for example, at an Ethernet LAN. Here, the upper limit (ideal) for an end-to-end application is the maximum transfer rate of the Ethernet cable itself: 10 Mbps. The actual limit or bottleneck could be in any of several places, such as in the interface(s) between computer and network, or in protocol processing.

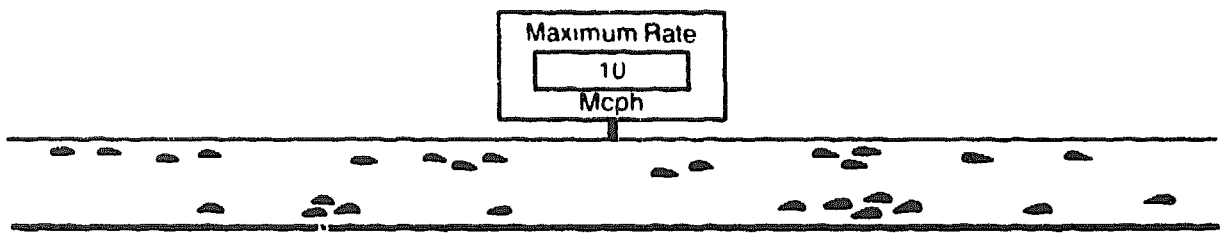


Figure 5-1 The Maximum Transfer Rate

Take the case of Mr. Smith, who manages Packet Auto Manufacturing, Inc., an automobile enterprise that includes a manufacturing plant and a car dealer. Sales are up. Demands exceed the company's ability to transport the cars to the consumer. Thousands of customers are on a waiting list. The only road connecting his plant to the dealer does not allow him to ship new cars fast enough. Alas, he learns about the construction of a new super highway that has a maximum transfer rate of 10 "megacars" per hour (10 Mcph). As Figure 5-1 shows, this rate is for ideal conditions. With no heavy traffic and no cars slowing the pace while entering or leaving the highway, very high speeds are possible.

Mr. Smith imagines sales skyrocketing at 10 megacars per hour. But, to his dismay, the first week after the new highway opens, completed sales are not as high as expected. Although he is shipping new cars at much better rates, sales still lag behind. He is upset that the new highway has failed to produce the results he expected.

Mr. Smith did not foresee several factors that would limit the actual throughput or shipping rate of new cars (see Figure 5-2). On closer examination, Mr. Smith realizes that the limiting factor is *off* the main highway, not *on* it. The source of the problem is a bottleneck at the toll gate on the entrance ramp. Another manufacturer is pouring cars out onto the same ramp. Thus, before Mr. Smith's new cars can gain access to the highway, they must contend with those of the other manufacturer that seek access. The toll booth at the entrance cannot keep up with the arrival rate of cars, so the cars have to wait in line. A similar wait occurs at the toll gate on the off-ramp: cars arrive faster than it can handle them. In addition, Mr. Smith has only two employees at his dealer to handle incoming new cars, and only a few sales personnel to serve the many customers.

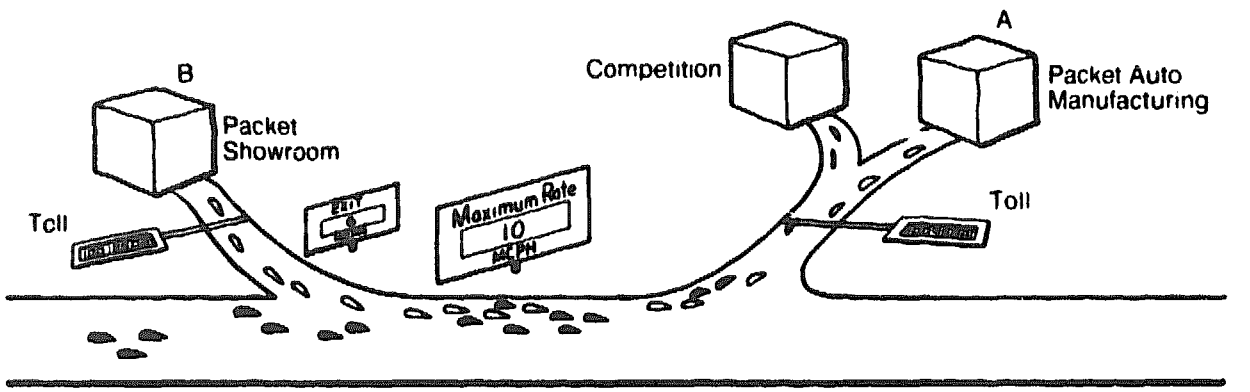


Figure 5-2 Throughput for the End-to-End Application

The result? Mr. Smith's new cars finally enter the highway at a slower rate than expected. They enter at intervals determined ultimately by the processing rate of the toll booth. Though all the cars travel on the highway at the maximum speed limit, the actual number of cars exchanged between manufacturer and dealer in a given time is less than the potential at that speed limit.

In network terms, the toll gates are the interfaces mentioned previously. (In Digital's PDP-11 and VAX/VMS nodes, the Ethernet interface [DEUNA, DELUA, DEQNA, and so forth] acts as a toll booth for incoming and outgoing network data.) The manufacturer's production rate is the application's message-rate: the rate at which the messages arrive into the network's "toll gate." The message rate depends largely on the user's level of activity or productivity. The manufacturer's production rate depends on how quickly each car can be processed for delivery (analogous to protocol processing).

The other manufacturer is another application at the same node that uses the network. At the other end of the highway (B), the dealer's rate of handling and selling new cars is the application's rate for handling messages received from the network, processing them for output, and displaying them. In Mr. Smith's situation, the prime bottleneck is the toll gate. But, it could be at any of the points just mentioned, depending on the relative capacities and loads on each component.

Given that the rate at which cars arrive on the highway affects throughput, the highway's real throughput does not fairly represent the highway's performance potential. Chapter 7 discusses a performance measure (the "utilization factor") that incorporates the effect of arrival rate. Also, the disappointing throughput of the highway reflects the throughput for exchanges between only two points on the highway. If Mr. Smith adds several more manufacturing plants and dealers, he can amass a very high throughput over the highway.

Like the highway, the Ethernet's throughput capacity exceeds that of any of the components that feed it. As a multiaccess channel, it more than compensates for this by allowing the flow of data among many sets of applications, not just between one pair of applications. Thus the combined, real throughput for all communications over the Ethernet can indeed approach the 10 Mbps mark.

In subsequent discussions, this book uses the word **throughput** *to denote the real, application-level meaning. It does not use this term to mean the rated, transmission capacity of a component.* When you read elsewhere about the hardware rating of a device, do not equate this with the actual-application throughput that you can achieve.

Be aware of another misconception. Suppose your network has a data link rated at one Mbps. You want to double the throughput, so you replace the link with a 2-Mbps link.

The mistake here is the assumption that doubling the rated throughput will also double the real throughput. This disregards the limiting factors mentioned previously. It would be similar to Mr. Smith replacing the 10 Mcph highway with a 20 Mcph highway. The limiting factors are still the same, and so the maximum *achievable* throughput for his enterprise would be no higher on the 20 Mcph highway than on the 10 Mcph highway.

Thus, the ratio of the *real* throughputs of the 2-Mbps and 1-Mbps links are not two-to-one; they are more likely to be one-to-one.

Total Throughput Versus User Throughput

The relationship between total throughput and user throughput is analogous to the relationship between the gross weight and net weight of a box of cereal. The gross weight measures the entire box, including packaging. The net weight measures the cereal only. Similarly, **total throughput** (or **gross throughput**) measures the amount of data transferred in time, without distinguishing the "consumable" user part of the data. User

throughput (net throughput, effective throughput, productive throughput, or TRIB) measures the amount of *user* data transferred. These two measures separate the user information (the cereal) from the protocol control information (the packaging). So user throughput is the measure of throughput after subtracting the protocol overhead.

User throughput also excludes retransmitted data (because of error conditions) and protocol control messages such as ACKs. In Mr. Smith's situation, the rate of sales could be affected by the error rate. The error rate can be problems that happen in transit or they can be manufacturing defects. In either case, these cars must go back to the manufacturer. Thus, the user throughput (cars that actually can be sold) for Mr. Smith is less than the gross throughput (all cars transported).

Throughput can refer to either user throughput or gross throughput. Be clear about your meaning of it. Usually, throughput refers to gross throughput.

Knowing both total and net throughput of a network can be useful. *The ratio of user throughput to gross throughput tells you how efficiently the network transfers user data. It reveals the extent of network overhead. More specifically, it measures the efficiency of the network protocol. It also reveals the reliability of transmission (extent of retransmissions necessary).*

At the application level, the goal is to minimize the impact of the protocol overhead. An effective way to achieve this goal is to send larger messages or send the data in larger blocks. This serves the same ends that buying cereal in bulk does: you pay proportionately less for packaging. Another way to minimize protocol overhead is to employ full-duplex lines instead of half-duplex.

Keep in mind that protocol information is a necessary part in any transfer of information. It can be minimized, but not eliminated. Also, where requirements call for more sophisticated network control procedures, this means more protocol control information.

With the introduction of faster communications lines, the impact of protocol overhead on performance decreases. More significant are the delays associated with the action of the protocols themselves (such as flow control, acknowledgment schemes, and so forth).

Aggregate Throughput

Thus far, the definition of throughput specifies the transfer rate of data from one point to another. But what is the throughput of a network as a whole, where messages are being transferred among many points at any given moment? And what about networks that provide **broadcast** functions that send one message simultaneously to many nodes (just as a television program is sent to a multiple of television sets)?

Aggregate throughput is the sum of the lengths of all terminating messages received on all nodes across the network per second, where a **terminating message** is any packet that has reached its destination node.

Notice that the definition considers the size of messages, not just the number of messages. This is because throughput measures the *amount* of data sent. The number of messages does not truly reflect the amount of data, because a message can be large or small in size.

The term **aggregate throughput** is often used to denote network throughput. However, be aware that this term has many different associations.

Summarizing Throughput and Its Distinctions

The definitions of throughput so far have provided various meanings in different contexts. Throughput can mean:

- The rated throughput of a device or link, meaning its maximum transfer rate or capacity.
- The real throughput attained with applications.
- The gross throughput, which includes overhead and user data.
- The user throughput, which includes user data only.
- The rated network throughput, which is the total bandwidth available throughout the network.
- The real network throughput, which is the real throughput produced by all applications in the network.

Throughput efficiency is the ratio of the real, user throughput to the rated throughput. High throughput efficiency means that the real, user throughput is high relative to the potential throughput. You can use this term with respect to one application or to the aggregate throughput of all applications. You strive for throughput efficiency given a fixed set of resources (the capacity or bandwidth of the components is fixed). Within this fixed set, you try to restructure the configuration, improve the software design, tune the network software parameters, or adjust the traffic loads.

WHAT THROUGHPUT USERS EXPECT

Application users would like to see the highest possible throughput for their respective applications. (If you are a network manager, you expect to see adequate throughput available for all users.)

Since the attainable throughput depends on the capacity of each communications component and on the available bandwidth of the network as a whole, no one should

expect more throughput than the capacity of the component or network under consideration. Components do not fall apart when the load approaches capacity, but their performance might. This is especially true for the performance of CPUs, multiplexers, and other "active" components of the network, rather than for the "passive" components such as the transmission media (which do not process or pump the data).

Consider a configuration where terminals are multidropped on a single high-speed line to a cluster controller. A common misconception is that the user throughput at each terminal will be comparable to the speed of the line. In truth, however, user throughput will be much less. The total throughput of the line is divided by the number of terminals sharing it.

Figure 5-3 shows some of the ways that throughput can behave as the load on the network increases. There is always an upper limit to throughput. The "ideal" throughput steadily increases with the load presented to the network until it reaches the maximum. It remains at that maximum value regardless of how much more the load increases.

The "good" curve is an example of what is typically the best *attainable* throughput behavior. The "undesirable" curve shows throughput peaking below the maximum value and soon decreasing as the load increases further. The "bad" curve shows throughput peaking well below the maximum value and decreasing suddenly after that (throughput then becomes zero, at which point the network is in a **deadlock**).

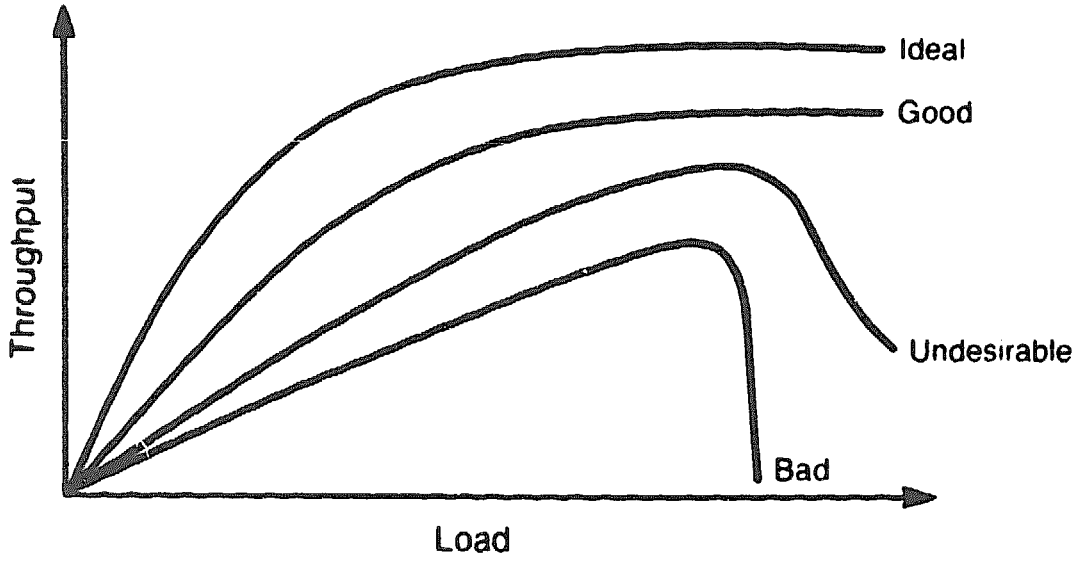


Figure 5-3 Throughput As Network Traffic Increases

THROUGHPUT THAT APPLICATIONS REQUIRE

As with response time, the desired throughput depends on the application. Applications that need to transfer large amounts of data need high throughput. Examples of such applications include remote batch processing, batch database updating, and bulk file transfers.

How to Formulate Throughput Requirements

To formulate your throughput requirements, you must estimate the amount of data that users will transfer across the network. Specifically, you must understand the nature of network traffic and of application processing at each node. The network traffic characteristics that you should know include such statistics as:

- Peak and average volumes of data to be transferred over the network.

  The components must have the capacity to handle peak traffic adequately. Usually you optimize throughput for the average volume of data flow, since that is the most likely volume. However, certain applications may require optimum throughput at the peak load. (Such requirements are more costly, of course — perhaps too costly. For that reason, it might be wiser to configure a network with the capacity to handle something like 95% of the expected peak load rather than 100%.)

- Sizes of data blocks transferred between network nodes.

  Typically, the limiting factor in data transfers is the number of blocks or packets that the system can send in a given period of time. Hence, throughput will increase if more data is packed in each block.

  If the traffic is mostly file transfers, for example, you must know what the block size is, the average number of blocks per file, and the average number of files to be transferred per unit of time.

To determine the required speeds and capacities of nodes and links, you must estimate the traffic patterns to and from the node. This means understanding the nature of the traffic that the application(s) at the node generates and receives. Specifically, you should know the following:

- Source and destination locations
- Message size (minimum, average, maximum)
- Nature of data flow (bursts or steady stream)
- Average and maximum intervals between bursts or messages

Knowing the source and destination locations enables you to determine the throughput requirements for the links between them. The total throughput requirement then is the sum of the link requirements. For instance, consider the configuration shown in Figure 5-4. Assume applications on this network use two logical links sharing the same line (MN): (1) exchanges between A and B, and (2) exchanges between C and D.

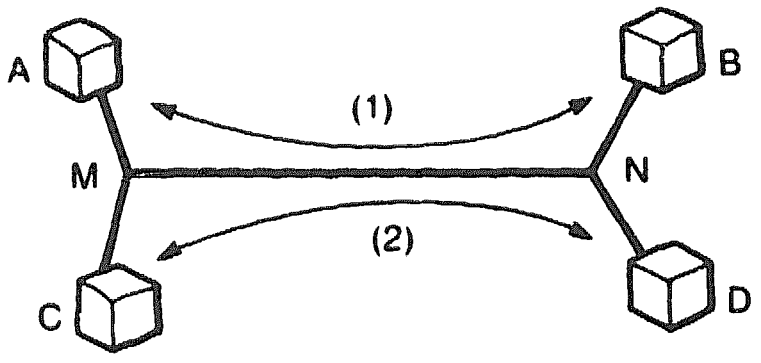


Figure 5-4 Defining Requirements for a Shared Point-to-Point Line

The total throughput tells you what bandwidth you need for the line MN. For example, if A and B need to communicate at 4800 bps, and C and D need to communicate at 9600 bps, then the line must have a bandwidth of at least 14400 bps (the sum of the two).

Once you have determined your throughput requirements, you may decide to add alternate routes. If several routes are already available, the throughput requirements may help you decide which ones to use. Once you have decided on the routes, then you can determine the requirements for specific components on the routes.

Figure 5-5 shows a bridge connecting LANs A and B. How do you figure your throughput requirements here? In this case, you would consider all possible exchanges that involve logical links between the two LANs. Then, you would take the sum of the required throughputs for those links.

In any case, you need to determine the traffic flow and loading needs to make sure the network can support those needs. This implies that you know your network's capacity. Without knowing that, you cannot determine the optimum throughput.

After estimating the amount of traffic you expect, try to project the growth rate of traffic, say for the next five years. Add this to the volume. Also, many planners increase the estimate another 15% or so to accommodate unforeseen traffic growth.

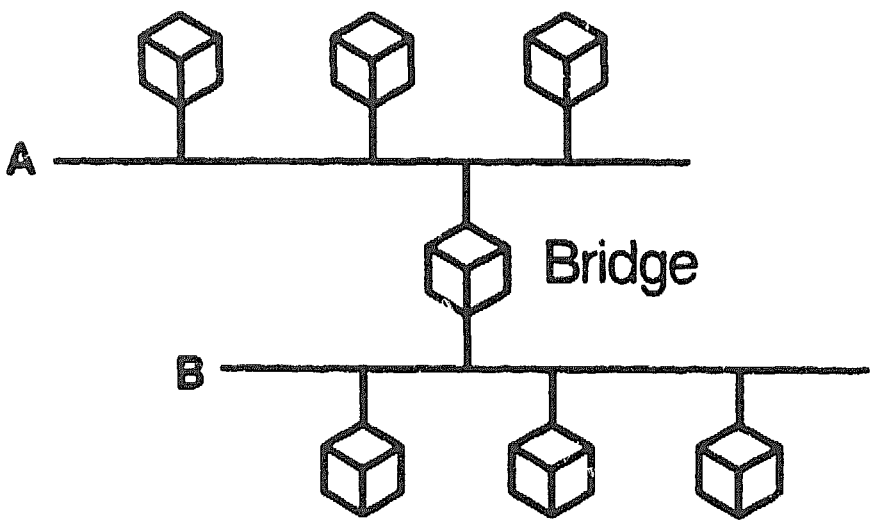


Figure 5-5 Defining Requirements for a Bridge or Router

Specifying Requirements for Non-Critical Applications

To devise the throughput requirements for an application, you can set up boundaries in the same way as in the preceding chapter. For simple applications, answer the following questions:

- What is the minimum acceptable throughput?
- What is the maximum throughput value beyond which there is no additional benefit to the user?

What Is Excessive Throughput

What determines throughput (the upper boundary) is the capacity of the network, or of any of its components, to absorb the traffic demand. As traffic increases, the load causes some point of the network to saturate. For example, a low-speed line can become saturated. If you have high-speed lines, then one or more CPUs at the receiving end can saturate. If you have a small computer, and you connect a 56 Kbps line to it for incoming data, your computer may not be able to handle the incoming data fast enough.

Also, cost may determine the upper boundaries. Large CPUs, for example, are more expensive. Furthermore, expensive wide-bandwidth lines are unnecessary where transfers of data are likely to be small.

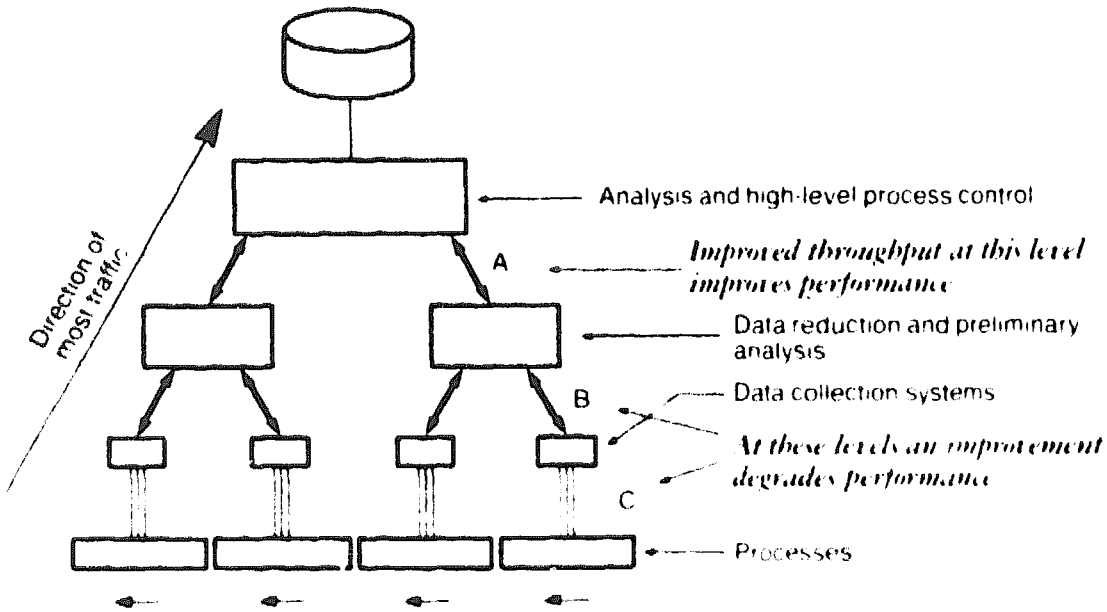


Figure 5-6 Excessive Throughput at B Degrades Performance at A

If you fail to balance throughput across the network, excessive throughput in one area can degrade performance elsewhere. In Figure 5-6, the data flow consists of several branches mostly in one direction, merging at the central computer. The greatest throughput requirements are at level A, where the greatest volume of traffic occurs (accumulating from the outputs of levels B and C). If the throughput generated by levels B and C is excessive, a bottleneck occurs at level A.

Specifying Requirements for Critical Applications

For complex or critical applications that require more than just an upper and lower boundary, you can plot curves showing the usefulness of a network's service as a function of throughput. For example, for file transfers over a low-speed line, the throughput increases until the line becomes saturated. The rate at which the usefulness of the file transfers increases or decreases depends on:

- The type of information being transferred.

  Is it volatile, meaning that it loses value to the user over time?

- The type of medium used for the transmission.

  Is it a public-switched line, in which case you are charged for the amount of time the line is in use?

The three curves shown in Figure 5-7 are basically the complements of those shown in Figure 4-3 (response time and usefulness). Graph 1 applies to those applications where the usefulness (productivity) is directly proportional to the amount of information that can be transferred.

Graph 2 shows the usefulness of an application increasing rapidly as throughput increases, but then gradually leveling off. For example, in using a virtual terminal set at 9600 baud, increases of throughput are very desirable to a point. But, beyond the point where the terminal cannot handle any more data, increases in throughput are less useful. Furthermore, when reading text output at your terminal, you can read only so fast. (You would not be able to read as fast as a 56 Kbps line pipes data to the terminal.) Also, as throughput increases, the level of traffic starts to cause queues to build. As queuing delays increase, response times begin to degrade noticeably.

Graph 3 might apply for an application where the transferred data is generated at a constant rate and the source has little or no buffer capacity. The network must provide enough bandwidth to handle the input; otherwise, flow control mechanisms may delay transmission of the data or request that the data be retransmitted. However, the traffic flow needed for high throughput cannot exceed the transmission capacity of the source.

These curves can help you match and tune network components, reduce bottlenecks, and ensure user satisfaction. The key is to understand the load characteristics. For example, if applications produce traffic in bursts, you have to tune the network much differently than for traffic that comes in a steady stream.

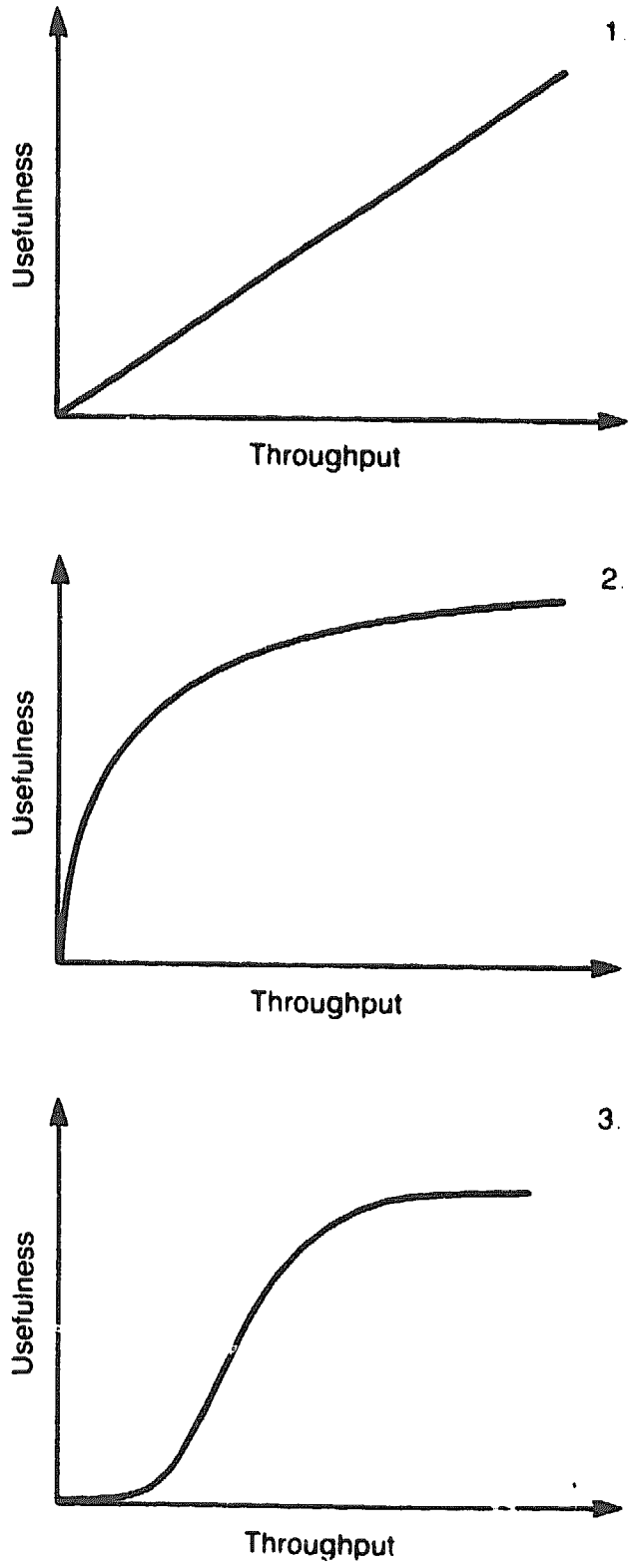


Figure 5-7 Throughput and Network Usefulness

HOW TO MEASURE AND EVALUATE THROUGHPUT

Throughput is relatively easy to measure. Many utilities and tools are usually available for this purpose. For example, Digital Equipment Corporation provides the DTS/DTR utility, Network Control Program (NCP) LOOP command, and network maintenance application programs such as the NMCC/DECnet Monitor and NMCC/VAX ETHERnim Monitor.

Expressing Throughput: The Units of Measure

The throughput of an application on a stand-alone computer system (a computer not connected to a network) reflects the output or production rate of the application. The throughput is usually expressed in such terms as jobs per minute or transactions per hour. You can measure the throughput of a network application in similar terms, as if the network were invisible. Or, you can measure in terms more clearly associated with the network, such as transfers per minute, blocks per second, or bits per second. Bits per second is the most common measure.

The unit of measure depends mostly on the application and type of network. For example, sometimes throughput is expressed as user-messages per second. For DECnet file transfers, throughput is usually expressed in bits per second or sometimes blocks per second.

Packets per second is used for expressing the throughput of DECnet routers or bridges. Packets per second is an important unit of measure at the routing level, or any other of the lower layers, because packets are the units of data transferred at that level. Refer to [5] for an explanation of why routing throughput is measured in terms of packets and not bits.

Calculating Throughput

The simplest way to calculate the user throughput of an end-to-end application is to divide the amount of data transferred by the time elapsed. (To clock the transfer time, use a SHOW TIME command file, as explained in the preceding chapter.) To minimize the effect of the connect time on the calculation, send a large amount of data, such as 1,000 to 10,000 blocks. (In this context, **block** means the smallest logically addressable unit of data that a specified device can transfer in an I/O operation, usually 512 contiguous bytes. Other areas of this book refer to block as a message block.)

Calculating User Throughput of File Transfers

Take a file transfer as an example. Suppose you transfer a 100-block file between two nodes on an Ethernet in 4 seconds. To find the throughput in bits per second, first

convert blocks to bits (512 bytes to a block, 8 bits to a byte). So one block equals 4096 bits (512 x 8). Throughput equals:

(100 blocks x 4096 bits/block) / 4 seconds = 102,400 bits/second

To find the gross throughput of an application, you also have to figure in the number of protocol bits per block. Naturally, the gross throughput will be higher.

Calculating User Throughput Based on Error Rate

Using the following formula, you can also calculate the user throughput of a communications channel if you know the error rate of the line. The block error rate (P), is the total number of user blocks received in error divided by the total number of user blocks received (DECnet's NCP utility displays counters that give error blocks and total blocks). For example, if the number of blocks received in error is 40 and the total number of blocks received is 40,000, the block error rate is .001.

$$\text{Throughput} = \frac{M \times (1 - P)}{(M/S) + T}$$

where:

M = message length (in bits)

P = Block error rate (error blocks divided by total blocks)

S = Line speed in bits per second

T = Time between messages in seconds

The value M is the average message size (or the weighted average). You can calculate the weighted average by giving more weight to the sizes that comprise larger percentages of the traffic (as discussed earlier).

(A message consists of one message block, or a series of message blocks, that constitute a logical grouping of information. Each message block is delimited by communications control characters.)

This formula shows the importance of the message size, especially over communications lines that are susceptible to errors. Recall that larger messages are more vulnerable to error conditions. (However, over LANs and most high-speed lines today, where the error rate is relatively low, larger message sizes are conducive to higher throughput.) The value of T (time between messages) is less significant in transfers of larger messages, because T decreases as the message length increases.

Estimating Throughput

Method 1: You can estimate user throughput over a data link if you already know the nature of the protocols used, such as how many control bits are added to each user message and how many messages are likely to be retransmitted because of error. To do this, use the following formula:

User throughput = U x (1 - E) x S

where:

U = Ratio of user data to total data per message

E = Probability that a message will be retransmitted

S = Line speed in bits per second

For example, assume you want to know the user throughput at the data link layer for asynchronous transmissions over a 9600 bps line. (You may want to compare a regular leased line running DECnet with a backup X.25 link, or perhaps you want to test the effectiveness of the communications hardware and the network protocols.) Suppose the probability that a message will be retransmitted because of error is .01. In asynchronous transmission, you know that two bits of control information are added to each eight bits of user information. Thus, the value of U is 8/10 or .8. Substituting the known values into the formula, you calculate the user throughput as:

User Throughput = .8 x (1 - .01) x 9600 bps = 7,603 bps

Note that this calculation only considers the user throughput with the overhead incurred by the data link protocol. It does not consider further overhead incurred at higher layers of the network.

Method 2: The preceding chapter explained how to estimate response times for file transfers. The formula was:

R = A + (P x S)

where:

A = Access time

P = Seconds per block

S = Size of the file in blocks

You can also use this formula to estimate the throughput. The value P (seconds per block) is the reciprocal of the throughput T (blocks per second). By solving for P, the preceding formula becomes:

P = (R - A) / S

and the reciprocal is: T = S / (R - A)

For example, if a 100-block file takes an average of 10.5 seconds to transfer, and the access time (average time to set up the link) is .5 seconds, then the throughput is:

T = 100 / (10.5 - .5) = 10 blocks per second or 40,960 bps

This figure eliminates the effect of the access time. If access time is not subtracted from transfer time, then throughput is less (about 38,994 bps).

Method 3: You can transfer a large file over a link of some fixed speed and see how long it takes. From the result you can estimate the real, user throughput. Suppose you transfer a 244-block file (approximately 1 Mb) over a 1 Mbps line. At the rated (theoretical) throughput of 1 Mbps, the transfer of the 1 Mb file would take 1 second. However, if transferring the 1-Mb file actually takes 2 seconds, then the real throughput is half of the 1 Mbps maximum: .5 Mbps.

(Note, however, that the transfer speed of the disk at either end of the link may be less than 1 Mbps, in which case the disk is a bottleneck. In this case, the calculated throughput reflects the performance of the disk more than that of the transfer across the link.)

Testing Throughput to Estimate What It Should Be

Once you have installed a new network or link, there are ways to test it to determine what the normal throughput should be. This is important so that you can identify problems and project expansion needs. You can create a file of about 2,000 blocks and create a batch job that does the following (running when no other users are on the network):

1. Zeros the network counters.

2. Starts three processes on the local node: each copies the same file to the same remote node.

3. Records the executor statistics and counters before and after the files are transferred.

Have this job run for about 48 hours. Based on the recorded statistics and counters, you can then figure the throughput (roughly). This tells you whether the throughput is sufficient for users *before* they start using the network. (DECnet software comes with a procedure that you can use to verify the functionality of the network.)

How Not to Measure Throughput

Do *not* measure throughput by equating it to utilization, which is a measure of how much of a resource's capacity is being used, expressed as a ratio or percentage (see Chapter 7). The fault with equating utilization and throughput is especially evident when applied to a CPU. High utilization may not be so much a virtue as a sign of poor software design (the CPU could be working very hard because the software fails to handle data efficiently). For example, an inefficient communications protocol may require a large amount of processing time for each packet transmitted, or an excessive number of I/Os may cause a high number of CPU interrupts.

Also, do *not* measure throughput by equating it to the number of users or processes that can be handled (which is insignificant). What matters is *what* they are doing and when. For example, a computer might be able to support 64,000 processes, but how many of these can be active at the same time?

FACTORS THAT AFFECT THROUGHPUT

Many factors affecting throughput are the same as those affecting response times. The factors originate in the same three areas:

1. The local system (source of the transaction)
2. The network (the transmission media)
3. The remote system (destination of the transaction)

The major factors include:

- Bandwidth or capacity of the hardware components that handle the communicated data (including CPUs, communications devices, and transmission media).
- Efficiency of the software components that handle the data.
- Workloads on the source and destination systems.
- Traffic on the network.

The Chain of Events for Throughput

Because throughput is a function of time, it depends on the same delays mentioned for response time (except throughput does not include the return time for a response). These are, in the order that they occur:

1. Processing time for the application data at the source node
2. Queuing delay entering the network
3. Access or connection time
4. Transmission time
5. Queuing delay for reception by the destination node

Processing of Data at the Source Node

The rate at which the data is processed and prepared for transmission depends on the CPU speed, CPU type, software efficiency, and perhaps the disk speeds.

It also depends on the message size, or the record-blocking factors that the transmitting application employs. Transmission of large messages or blocks of data has higher throughput than that of smaller ones because proportionately more user data is sent in relation to the fixed overhead.

Also, when it is necessary to send a large amount of data, it is more efficient to send the data in larger messages or blocks. Otherwise, the CPU has to process more messages for the transmission, and this causes more CPU-processing overhead. It takes the CPU longer to transmit 10,000 bytes of data in 100-byte messages than in 1,000-byte messages.

However, larger message sizes demand more from the system, such as buffer space. More important, a larger message is more likely to be affected by error conditions while traveling over a line (causing a greater number of retransmissions).

System buffers should be sufficient in size and number to handle the messages. If the buffer size is too small, the CPU has to segment the messages to fit into the buffers (and the CPU at the receiving end has to reassemble the segments into the original message). If the number of buffers is insufficient, more CPU time is spent dealing with data that "overflows" the buffers (delaying transmission until buffers are available).

However, if the number or size of buffers is too large, then CPU memory is wasted. This could degrade the performance for other tasks.

Queuing for Access

To access the network, data first enters a queue at the source node. The size of the queue depends largely on the amount of arriving data and on the waiting time for access to the network. The arrival rate of data depends on the processing rate of the application (or the rate of user input).

Keep in mind that other applications at the source node could also be pumping data onto the network, thereby increasing the arrival rate. The source node could also be a routing node, in which case routed data is added to the queue.

Another significant factor is the pattern of the data flow. If the data arrives in bursts, longer queues are more likely than if the data arrives in a steady stream.

Accessing the Network

The data at the head of the queue has to wait for an opportunity to enter the network. For dedicated lines, there is no wait.

In contrast, consider the access time for shared networks or multipoint lines. In Ethernet LANs, for example, the CSMA/CD protocol waits for the line to be free before allowing transmission. In event of collision, the transmitter backs off and tries again. (Collisions are more numerous as the traffic increases, so the access time increases too.) In token ring networks, the device that wishes to transmit has to wait for the token to pass by (the token must not be in use by another node). In polling schemes, the transmitter has to wait to be polled before it can transmit.

For transmission over dial-up lines, the transmitter waits for the dialing and setting up of the logical connection or circuit. With file transfers, this delay can hamper performance, especially for transfers of short files (where the time to set up a logical link could be longer than to transfer the file). Furthermore, if connections are frequent, the delay detracts considerably from throughput. If connections are infrequent, then the connection time is less significant.

Another factor here is window size. In some networks, such as DECnet, you specify window sizes for **window flow control.** The window size is the maximum number of data packets that can be in transition at a time. When the number of packets equals the window size, transmission stops until the number of outstanding packets drops below the window.

If the window size is too small, throughput is limited. As the window size increases, throughput increases. However, if the window is too large, throughput starts to decrease when the network becomes congested (routers become overworked and queues build up at destination nodes). This is called the **point of congestion.** This point depends upon a number of factors, including the amount of buffers at routing nodes, line speeds, CPU speeds, and the distribution of packet sizes.

Also significant here is the type of protocol. Some protocols require that the receiver of the message send an acknowledgment (ACK). This decreases throughput because the transmitter has to wait for an ACK for each message before it can continue. Some protocols, such as those used by DECnet and SNA, require an ACK for each block of messages, instead of requiring an ACK for each message. This increases the throughput per ACK.

Most networks let you limit the total number of messages sent without an ACK. In DECnet/VMS, this is called the **pipeline quota.** The pipeline quota affects the maximum window size. If the receiver cannot handle the rate of incoming messages (because the

pipeline quota is high), then the network software automatically decreases the window size. When the receiver is able to handle the incoming messages, the window size automatically increases.

Transmission Time and Error Handling

The error rate of the transmission media affects user throughput. If the error rate is high, then more retransmissions are likely. To minimize exposure to error conditions, you can reduce the message size. However, throughput improves with larger messages. Usually there is an optimum message size, which is a compromise. Figure 5-8 shows that, all other factors being equal, user throughput (TRIB) increases as the message size increases, but then reaches an optimal point (at *m*). After that point, the probability of error is so high that the number of retransmissions increases and throughput falls.

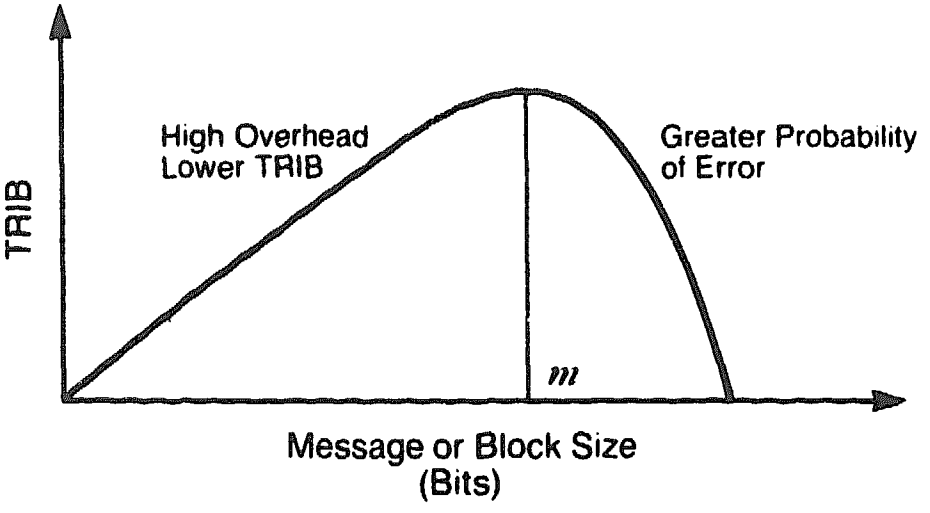


Figure 5-8 Optimum Message or Block Length

Low-speed lines tend to be more error prone than most high-speed lines. (This is because of the technologies used, not the speeds. For example, 56 Kbps lines have very low error rates because they use digital signal transmission.) Retransmits on a low-speed line take up a relatively larger amount of bandwidth.

Forward-error correction is sometimes used in the protocol for handling errors. It detects and corrects the erroneous blocks without retransmission, but because it does not catch every error, applications requiring high data integrity use it along with the conventional error-detection techniques. However, note that forward-error correction

works by adding extra overhead bits. Thus, it limits user throughput if used indiscriminately. Also, the hardware necessary to handle forward-error correction is expensive.

The user throughput of a line will be a percentage of its line speed, depending on the efficiency of the protocol and the number of error-caused retransmissions. This is sometimes called the **effective line speed.** Over typical point-to-point connections, throughput is between 70% and 90% of the line speed (ignoring connection time). DECnet/SNA Gateway connections transfer data at 80% of the line speed.

Another consideration is whether the communications device is a direct memory access (DMA) device or a character-interrupt device (see Appendix A). The DMA gives higher throughput because it transfers a block of data directly to memory before interrupting the CPU for I/O. The character-interrupt device interrupts the CPU for every character and, therefore, requires much more CPU overhead.

Intelligent devices that access memory directly (such as DMA) save much CPU time for other tasks on the system. Also, transmitting a large block of data takes no more CPU time than transmitting a much smaller block. Thus, the DMA lets users have the benefit of sending larger blocks of data without the greater costs in transmission time.

Delay at the Destination

When the data arrives at the network interface of the destination node, it enters a queue. Several factors can increase the delay:

- Buffers are not available for incoming data.
- Buffers are too small; segmentation of packets is necessary to make them fit.
- A previous packet is still being processed.
- Insufficient window size. The window is too large, causing an overload in the receive queue.

Another consideration is the capacity of the node's peripherals used for storing incoming data. For example, magnetic tape units and disks can impose a delay if they are not fast enough to handle the incoming traffic. Their slowness can adversely affect applications using high-speed links (such as those used in LANs).

Workloads and Traffic

As with response time, a major factor influencing throughput is the load applied to the network. Figure 5-9 shows that throughput usually improves significantly as the load increases. You attain maximum throughput when all queues have data ready to be transmitted on to the next link in the network (there is no lapse waiting for new data).

When the network saturates, throughput stays at the maximum (or falls if the network is poorly designed). A similar curve can apply to any component of the network, including the CPU at each node. Contrast this curve to response time (shown in the inset in Figure 5-9), which worsens minimally, then at the critical point takes a steep turn for the worst.

On a transmission medium, the data does not slow down as cars do when the traffic reaches saturation. All signals travel at the same, constant speed. And, unlike cars, signals travel at that speed even in "bumper-to-bumper" traffic. However, when the medium becomes saturated, the throughput of that medium levels off.

In Ethernet LANs, an unusually high amount of traffic can increase the number of collisions, thereby degrading the throughput for the network as a whole.

On high-speed, point-to-point lines, the throughput depends on the transmitting CPU. The CPU is usually not fast enough to pump data at the line's capacity, even when the data is processed in a constant stream. The processing time of the CPU or interface imposes some delay (gap) between transmission of packets. These delays decrease the throughput, just as the number and width of the spaces between the words you are now reading decrease the total amount of characters that you read in a given amount of time.

Even on low-speed lines, the CPU imposes a slight delay between packets. However, the delay has less impact. The same delay on a low-speed line costs less bits (in throughput) than on a high-speed line.

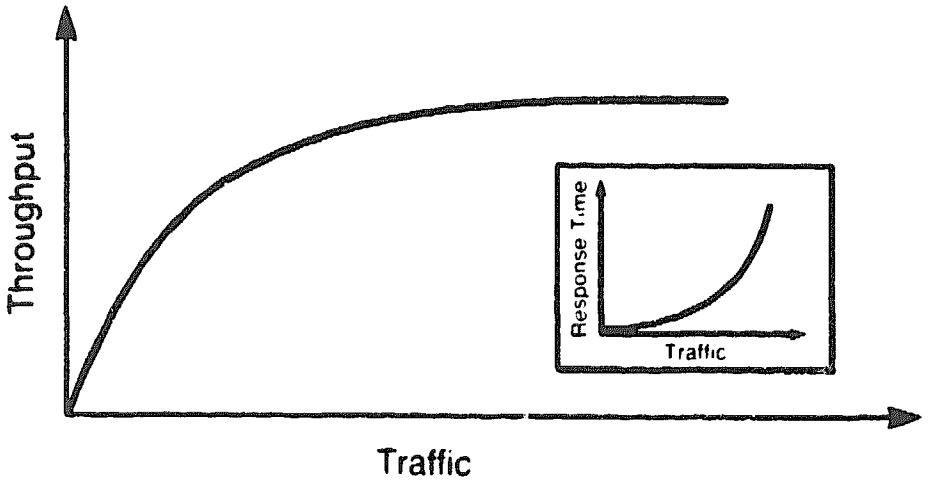


Figure 5-9 Throughput and Traffic

The Application

The application can have a very positive or very negative effect on throughput. For this reason, it bears repeating that the application must be designed with the network in mind. This requires an understanding of how the network functions and also of how the local operating system functions. For example, the designer should understand the buffering techniques employed by the network and system.

As another example, task-to-task and file-transfer applications differ in how they use system resources; therefore, they provide different throughput rates. Task-to-task is faster because data is not transferred to disk as in file transfers.

IMPROVING THROUGHPUT

The discussion of Mr. Smith's auto enterprise indicates several ways to improve throughput for both the end-to-end level and the network as a whole. One solution that covers both levels is to use larger message or block sizes. This is the key, for example, to improving throughput of Ethernet LAN data transfers.

In Mr. Smith's case, the toll booth was his major bottleneck — that is, the interface to the network. When the interface is not the bottleneck, and when the application at the source node is not producing data fast enough to keep up with the interface, then you might consider stepping up production at the application level. This requires streamlining the application or tuning the system. One goal is to find the optimum buffer size for the throughput needed. Another possibility for the application is multibuffering.

Also, consider **data compression**, which removes unnecessary gaps, redundancies, and empty fields from data streams. Thus, it reduces the transmission time and buffer/storage space requirements. Data compression is especially important when transmission time, line contention, or storage space are critical issues. This technique is costly to implement, so you have to weigh its merits against cost.

As mentioned earlier, the first step to improving network throughput is to find the bottlenecks. You do not want to improve the capacity of a component if it is not the bottleneck. Not only are you wasting your money, but you may also cause more trouble "downstream" from the component. If you do decide to expand the capacity of a component, consider the consequences for other parts of the network. You may have to make similar changes downstream, to avoid bottlenecks there.

Keep in mind that it is possible to improve network throughput, while at the same time degrading the throughput for one user or a portion of users. Also, network throughput is not necessarily affected by a single low-throughput link, unless that link provides the only path between certain portions of the network.

Once you know the bottlenecks, there are several alternatives to consider, listed below in order of expense to you. These are similar to those mentioned in the preceding chapter.

1. Stagger the load to reduce the intensity of traffic at peak loads and, therefore, reduce the demand on the components under greatest stress.

2. Tune buffer sizes to optimize performance.

3. Buy more bandwidth.

4. Buy faster CPUs.

To get the best throughput for an application, you require a network that is designed well. A poorly designed network can limit the throughput of an application. A well-designed, easily managed network offers you options that you can use to enhance throughput efficiency, such as flow-control options, and pipeline quotas.

One new possibility for improving throughput cost-effectively is dynamic bandwidth switching. For example, in video conferencing, which requires multimedia communications over a link, extra bandwidth can be switched to a link while the conference is in session. When the conference ends, the extra bandwidth can be switched elsewhere, as needed.

Note that increasing the throughput does not necessarily increase the cost in the long run. For example, if you are using a public network to transfer a large amount of data over dialup lines, it would be worth the cost to buy a more expensive 2400 baud modem than a 1200 baud modem. The resulting higher throughput will decrease the cost you pay to the common carrier, since charges may be based on the length of each connection.

There are other ways to improve throughput. Several methods are discussed below:

- Make changes that decrease the error rate over a line.

  For example, you should fix or replace noisy lines. You can buy conditioned lines from common carriers. Common carriers guarantee the error rate for these lines. You can install modems with forward-error correction. These correct errors, eliminating the need for retransmission.

- Increase the number of alternative paths between nodes so that, if one path is not available, or if its performance is poor (such as a noisy line that requires frequent retransmissions), then the system can choose an alternative.

Another consideration when improving throughput is this: what are the consequences for response time? As the following chapter points out, improvements to throughput can degrade the responsiveness of the network.

SUMMARY

Throughput can be defined in many ways, each expressing some idea of the work performed or the amount of data transferred during a unit of time. Throughput measures the amount of data transferred in one direction, while response time considers the round-trip delay for request/response applications. Throughput matters most in applications requiring transfers of large amounts of data.

You can consider the throughput of (1) individual components such as a communications line or device, (2) an end-to-end application, or (3) the entire network (aggregate throughput). As applied to individual components, throughput usually means the rated capacity or bandwidth, such as the line speed or signaling rate of a line. The real (measured) throughput on the application level is always less than the rated throughput. The extent of that difference depends on factors such as protocols used, buffering allocated, and design of the application.

As a measure of the available bandwidth in a network, throughput tells what amount of resources can be tapped at any time for communications purposes. It reflects the potential productivity. As a measure of the real, attained transfer rate of user information, it tells how much useful information is being communicated in time; as such, it is a good measure of the *actual* productivity, compared to the *potential*. In either context, throughput gives a solid account of the productivity of the network for any application or configuration. Also, it gives a sound basis for evaluating what you need in the network to support the traffic that your applications generate.

6 RESPONSE TIME VERSUS THROUGHPUT

The preceding two chapters treated response time and throughput separately. Many users, however, consider these performance measures inseparable when it comes to satisfying their needs. Unfortunately, the main ways to improve both measures simultaneously, for all users (network-wide), are the most costly. For example, you can upgrade the hardware, buy more powerful CPUs, and increase the bandwidth at strategic points in the network. Most other alternatives favor one measure over the other or at the expense of the other. This chapter discusses the balance and major tradeoffs between response time and throughput.

BALANCING THE SCALES

Since response time and throughput are two of the most important criteria of network performance, you can judge a network's performance by how well it trades off throughput for response time, or vice versa. To help measure the quality of the tradeoff and optimize the network service for both criteria, you can determine the power of the network, which is the ratio of throughput to response time:

$$\text{Power} = \frac{\text{Throughput}}{\text{ResponseTime}}$$

Power is expressed as the number of bits per second-squared. Throughput (bits per second) divided by response time (seconds) produces the unit of measure bits per second-squared.

The power of a network reflects how much throughput the network delivers relative to how quickly it responds. For throughput, you consider the aggregate throughput of the network. For response time, you use the mean response time. (Or, you can consider the power for each specific application, taking the mean throughput and mean response time for each.)

There is no predetermined, optimum value of power for networks. The purpose of this measure is to provide a benchmark for comparison of (1) different networks or (2) the same network before and after changing the configuration or parameter(s).

Network power increases as the throughput increases, or as the response time decreases, or both. By maximizing the power of a network, you can find the "happy medium" or point of balance that brings the greatest satisfaction to users as a whole.

Notice that you attain optimum power by a *balance* between throughput and response time. The best values of both throughput and response time — where one is not sacrificed miserably for the sake of the other — are usually somewhere between extremes. If throughput is extremely high, response times will usually be too long; if response times are extremely short, throughput will usually be too low.

This idea of balance assumes that both response time and throughput have equal importance. What if one of the measures has more importance than the other? The next section explains how to apply the power ratio to optimize the network in favor of either throughput or response times.

TIPPING THE SCALES

To find the optimum power value that favors either throughput or response times, use the following formula:

$$\text{Power} = \frac{(\text{Throughput})^{a}}{\text{ResponseTime}}$$

where the exponent "a" is a positive real number. (Note that "a" is the exponent of the numerator, throughput, not of the entire fraction.)

To give more weight to throughput, you can let "a" be greater than one. In this way, increases in throughput bring relatively greater increases in power, regardless of the increase in response time. Power will be greatest when throughput is highest.

To give less weight to throughput, let "a" be less than 1 so that power will be greatest when response times are shortest. (A fractional exponent is a root; for example, an exponent of 1/2 means you take the square root of the throughput.)

The following three examples show how the power ratio is applied where (1) throughput and response time are of equal importance, (2) throughput is of greater importance, and (3) response time is of greater importance. Suppose the measured mean throughput is 6400 bps and the mean response time is 2 seconds.

Example 1: To give both measures equal weight, let "a" equal 1. Thus, power equals 6400 bps/2s, which is 3200 bits per second-squared.

Example 2: To give more weight to throughput, let "a" equal some number greater than 1, say 2. Here, power equals the square of 6400 divided by 2, which is 20,480,000 bits-squared per second-cubed.

Example 3: To give less weight to throughput, let "a" equal 1/2. Here power equals the square root of 6400, divided by 2. The result is 40 bits-to-the-power-0.5 per second-to-the-power-1.5.

The power values in the above three examples vary greatly because they are expressed in three different units. Therefore, it would be invalid to try to compare the values. However, once you have chosen an appropriate value of "a", you can monitor the relative power usage of your network for that value.

Another more common use of power is to compare two different configurations under the same traffic load. A configuration that gives higher power for the same load is better.

POWER TUNING

Power is the most "compact" measure for evaluating the performance of networks. It is a two-for-one deal: it covers both throughput and response time. You can use the power measure to determine the relative effectiveness of various parameters. To tune a network, you can test various parameter values to see which one brings the greatest power.

To tune the network to satisfy the conflicting needs of different types of applications, such as a throughput-oriented application and a response-oriented application, look for the parameter values that bring the highest power or balance between the two applications. This is called **power tuning.** For example, you can tune the network by varying the:

- Intensities of traffic on the network or workloads on a CPU
- Sizes of packets, messages, or buffers
- Sizes of windows for flow control
- Queue sizes

Keep in mind that for each of the parameters listed above, a change in value usually has opposite effects on response time and throughput. For example, smaller messages are conducive to better response times (because of shorter transmission time), but not necessarily to better throughput. (This assumes that small messages, such as inquiry requests, are sent intermittently; if you have to send a burst of data, then larger messages are more efficient for both response time and throughput.)

The key parameter for performance tuning is the traffic intensity or workload. You can learn much by comparing the performance of a network at various intensities of traffic. For example, given a particular hardware and software configuration, you can apply various levels of traffic to the network and compare the power achieved at each level.

The series of graphs in Figure 6-1 shows ways to find the optimum level of traffic for a network. The first three graphs show that, at traffic levels beyond the critical point (m), the network becomes saturated. Throughput levels off, response time degrades considerably, and as a result, power decreases.

Graph 4 shows power with throughput emphasized (a is greater than 1). Notice here that maximum power occurs at a higher level of traffic (m1). At this point, response times are longer, but throughput is closer to its maximum. Graph 5 shows that when response time is given more weight, maximum power comes at a lower level of traffic where response times are shortest.

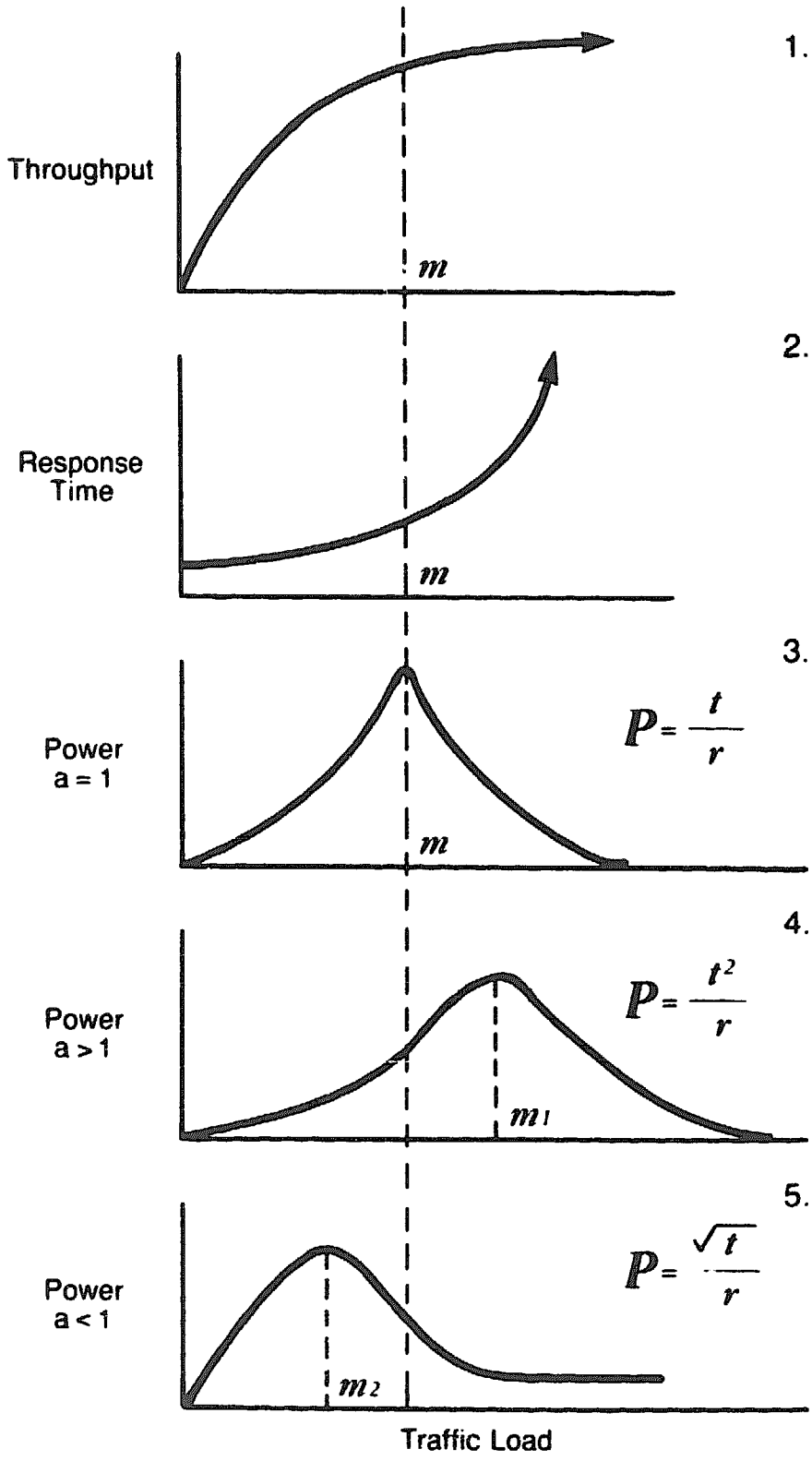


Figure 6–1 Finding the Optimum Traffic Flow with Power

FACTORS THAT AFFECT POWER

The factors affecting the power of a network include all those that affect the throughput and response time, as described in previous chapters. What matters for power is the relationships among these factors. The major factors are:

- Level of traffic
- Pattern of traffic
- CPU capacity and availability (CPU time) at nodes
- Available bandwidth of communications lines
- Error reliability of lines
- Processing time/user message rate at source node(s)
- Buffering for network communications
- Message, block, or packet size
- Protocol efficiency
- Size of queues; queuing delays
- Transmission/propagation time (line speed and distance)
- Network topology
- Communications devices used
- Switching characteristics
- Application design

IMPROVING POWER

Because networks are often judged by how well both response time and throughput are achieved, power serves as an effective guideline for improving network performance.

The ultimate solution for acquiring more power, without consideration of cost, is to increase the bandwidth of the network. This means buying high-speed lines, larger computers, and faster communications devices. To satisfy both throughput-oriented and response-oriented applications, you can employ separate mechanisms to benefit the demands of both. For example, you can allocate separate links for each application, a high-speed link for fast, small messages, and a less expensive low-speed link for larger messages that are not time critical. The entire network can be dedicated in this way.

A **dedicated network** is one that consists of dedicated lines and computers. Each line or computer is dedicated to a specific type of application or user. You can tune each computer or link for a specific use. In addition, users do not have to compete for resources as on timesharing systems or switched networks. The tradeoff with dedicated systems is the loss of flexibility. A dedicated system may perform optimally for a given time or purpose, but if circumstances change, resources will be wasted until they are rearranged or rededicated. Also, when dedicated users expend a resource or need more resources, they have to buy a whole new set of resources — you cannot share resources (by definition) in dedicated networks. With dedicated networks you have a significant cost/performance tradeoff.

You can get more power by distributed processing instead of centralizing it in one large CPU. By distributing the processing and buying more, smaller CPUs, you increase the amount of CPU time available per user. Thus, you improve responsiveness and throughput. (As more computing power goes to the office, the communications capacity of networks becomes a limiting factor. The most cost-effective solution to this problem is LANs with large bandwidths, such as Ethernet.)

To avoid investing in more equipment, you can optimize power by adjusting the traffic levels of the network or moving the applications. Some ways to adjust the load for better performance include:

- On WANs with a large number of routing nodes, you can change the *costs* of various links so that the traffic is distributed evenly throughout the network. In DECnet, each link is assigned an arbitrary cost value that helps determine the best path for a packet. A router uses the path with the least total cost.

- Where certain parts of the network are easily congested, you can move applications to an area where more bandwidth is available.

- You can move applications in "time," too. For example, you can run file transfers in batches at night instead of during the day when traffic is already high.

- You can close down low priority applications during peak hours.
- You can add dialup lines for use during peak hours. As shown in Figure 6–2, when traffic along path B–C–D–E is heavy, you can use the dialup line to remove some of the load from B–C–D–E.

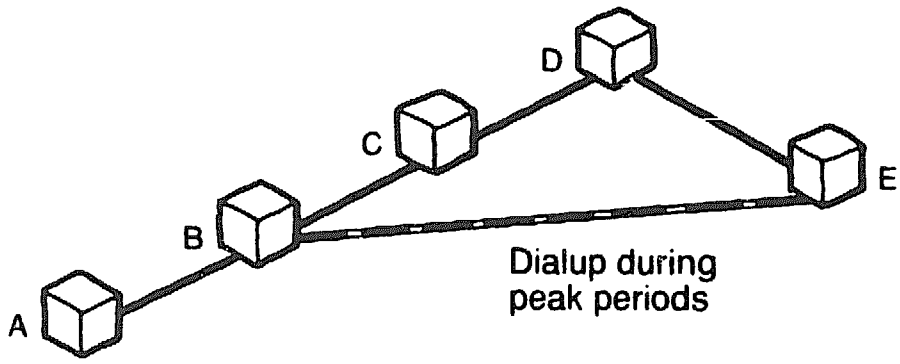
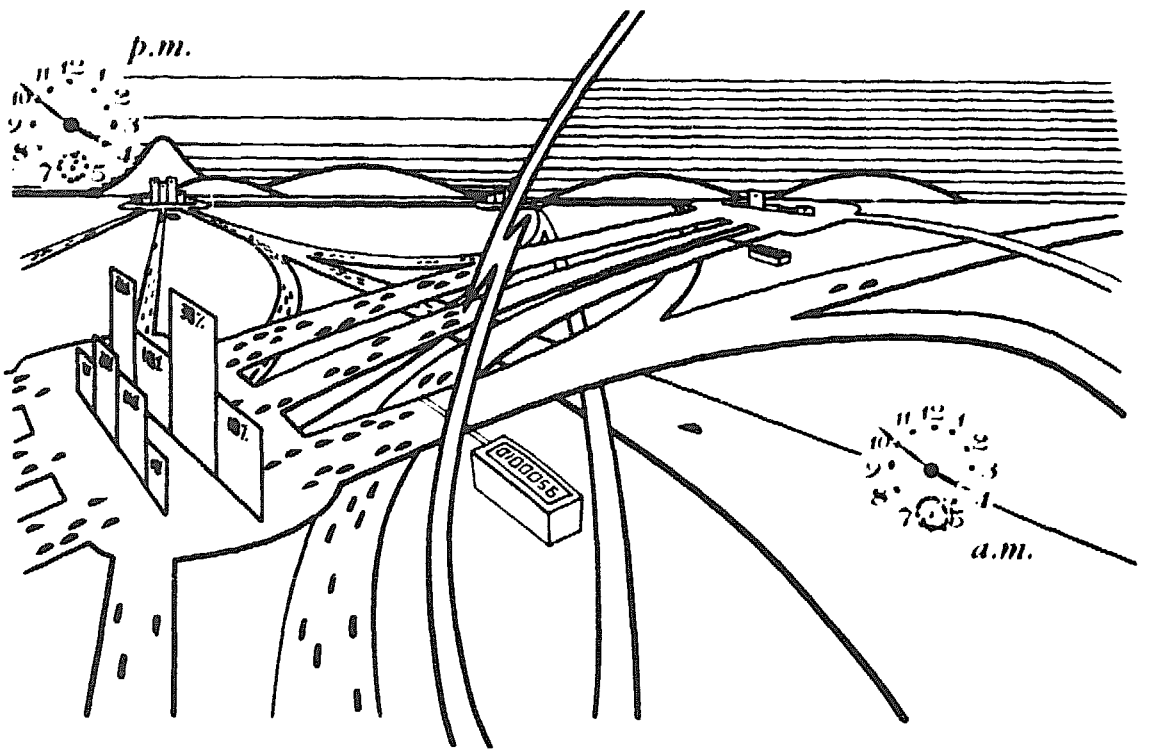


Figure 6–2 Adjusting the Load

7 RESOURCE UTILIZATION



Chapters 4, 5, and 6 covered *external* measures of network performance — measures that reflect the network's productivity. Chapter 7 covers *internal* measures of network performance — measures that reflect how efficiently the components work individually and together as a unit. Response time, throughput, and power of a network depend on how its components work.

These internal measures are known as resource utilization measures. The following sections discuss resource utilization in general. Subsequent sections deal specifically with CPU, line, memory, disk, and network utilization.

AN OVERVIEW OF RESOURCE UTILIZATION

Resource utilization is a key to helping you maximize the performance of your equipment, at the least cost. By measuring and observing resource utilization, you can identify or prevent bottlenecks, and you can assess future needs as network traffic changes over time.

Resource utilization measures the percentage of time that a resource is busy. For example, 80% CPU utilization means that the CPU is busy 80% of the time. The remaining 20% is called **idle time**: the time that the computer is available for use, but not in operation.

You can talk about the utilization of the network as a whole, or of any of its components in particular. On one hand, network utilization reflects how all the cooperating parts are being used as a unit. On the other hand, the utilization of each component shows how each contributes to the performance of the whole.

You approach resource utilization in the same way that business or department managers manage their resources. These managers are responsible for purchasing equipment and hiring persons to do the work that needs to be done. To justify the expenditure for employees and equipment, there must be sufficient work to be done. If the utilization of a resource is relatively small, it might be put to better use. In a sense, it is an "underachiever:" its capacity is being wasted, and its productivity or throughput falls below its potential. You usually try to avoid this, much for the same reason that managers do not want any employees to be idle.

This does not mean to imply that low utilization is synonymous with low performance, nor that it should always be avoided. There are situations where the utilization of certain resources is intentionally kept low. Low utilization often ensures better response times. It also ensures higher availability. For example, with applications requiring high availability of a resource, you can keep a backup resource as a standby for times when the primary resource becomes overloaded or fails. When not serving its backup role, the backup can take care of other activities that consume a small part of its capacity or that can be interrupted quickly. In this case, low utilization of the backup is normal.

If utilization is high, you can say that the component is at full use (no waste of capacity). However, the utilization can be too high, causing the component's performance to degrade. An office manager does not want any of the staff to be so busy that new orders must be postponed. Similarly, when a network component becomes overburdened with traffic, queues grow quickly, causing delays that grow exponentially. The component becomes a bottleneck.

By monitoring resource utilization, you can help identify or predict the source of performance degradation. It can also help you identify parts of the network that can tolerate a greater workload.

Gross Utilization and Net Utilization

Often it is helpful to distinguish between net utilization and gross utilization, for the same reasons that net throughput is distinguished from gross throughput. **Net utilization** is the average percentage of a capacity occupied by processing of user data. **Gross utilization** measures the percentage of capacity occupied by the overhead as well as user data.

For example, net line utilization measures the percentage of the line's bandwidth used by user information. It does not include the protocol overhead, such as control information, retransmissions, acknowledgments, and so forth. Thus, the net utilization of a line may be around 70 to 90% of its bandwidth or capacity. The gross line utilization includes the overhead as well as the user data, which together may take up the line's total bandwidth.

As for a node's CPU, its net utilization measures how much of the CPU is consumed by the processing or transfer of user information. The gross CPU utilization measures the total CPU consumption, including overhead. The overhead includes both network and supervisory functions (scheduling, interrupt processing, memory management). Another term for net utilization is **effective utilization.** Net or effective utilization can also be thought of as the ratio of the real (user) throughput to the rated throughput.

Note the difference between effective utilization and resource utilization. Effective utilization measures how efficiently a resource is being used. Resource utilization measures how busy the resource is.

Effective utilization reflects how productive a piece of equipment is relative to its capacity, while resource utilization reflects how much it works relative to a given lapse of time. The former tells you how much you are getting for your money (you may have a big computer that produces little for its cost). The latter tells you how to get more for your money (you may have a big, expensive computer with the workload of a small, inexpensive computer).

Often the terms *efficiency* and *utilization* are used interchangeably. Efficiency is synonymous with *effective* utilization (the ratio of actual user throughput to the rated throughput), not with resource utilization. An efficient machine can have low resource utilization. In fact, its low utilization is probably a result of its efficiency: it gets the job done quicker, and therefore it has more idle time. Similarly, an inefficient machine can have high utilization, because so much of its time is spent with overhead.

The Utilization Factor

The **utilization factor** of a component is the ratio of the average arrival rate (incoming traffic) to the average service rate. This measure is important for studying queuing delays within a device or network.

The utilization measures discussed previously measure the performance of a component with respect to the total time it is used or to its capacity. They do not measure the

performance with respect to the amount of work or data given to it over time. Thus, the effective utilization of a component may be low not because of the component's inadequacy, but rather because of a low arrival rate of data. (Similarly, the *measured* throughput of a device may be low because there is little traffic during the measured interval.)

Looking to a bank for an analogy, the utilization factor of the bank teller would be the ratio of:

- The *average* arrival rate of customers at the teller's window; that is, how many customers arrive in a given unit of time.

  to

- The *average* rate at which the teller processes the transactions.

A bank manager who judges tellers solely by how many transactions they handle in a day could be inadvertently penalizing tellers who might have been approached by fewer customers than the other tellers.

One flaw of the utilization factor is that it does not always reflect peak loading or bursts of traffic. If the traffic flow is a steady stream with no unusual or excessive demands, the utilization factor accurately reflects the average fraction of time that the resource is being used. With such traffic, queues usually stay small or empty. But when data arrives randomly, in bursts, queues can grow rapidly, causing a ripple effect on performance throughout the network.

By using the *average* arrival rate to reflect the activity of the resource, actual variances of traffic flow are "smoothed over." Thus, the arrival of traffic in bursts, which is a very important performance factor, may be overlooked. For this reason, consider the variance (standard deviation) of the arrival rate of data, not just the average. If the variance is low, the average is enough to understand fully the flow of traffic. If the variance is high, other statistics are necessary (see Appendix B).

HOW TO MEASURE RESOURCE UTILIZATION

Software utilities are usually available to monitor and record the utilization of various resources. The following sections describe how to calculate certain utilization measures and how to evaluate them.

Some Calculations

To measure resource utilization, you simply divide the total time that the resource is busy by the total time elapsed. So, if a resource is busy for 75 minutes during a 100-minute interval, its utilization is 75%.

One of the most important calculations is the effective utilization of a component:

$$\text{Effective Utilization} = \frac{\text{Real Throughput}}{\text{Rated Throughput}}$$

In Chapter 5, for example, the calculated throughput for a transfer of a 100-block file taking 4 seconds was 102,400 bps. The transfer occurred over a 10Mbps Ethernet LAN. The effective utilization of the Ethernet channel for that file transfer would be 102,400 divided by 10,000,000 (or about 1%). This file transfer consumes a very small portion of the Ethernet's bandwidth. The Ethernet has *plenty* of room for more activity.

Also in Chapter 5, the user throughput (at the data link level) over a 9600 bps line with asynchronous transmission was estimated to be about 7600 bps. Thus, the effective utilization of the line at that level is about 80%.

These examples reveal several things. For a high-speed, shared resource such as Ethernet, the demand of a particular application is small (1% in this case) compared to the resource's capacity. The bottleneck is more likely to be the interfaces, the disks, or the CPUs that are transmitting and receiving the data over the Ethernet.

To optimize the performance of the network, you must use the CPUs efficiently (so that they can move the data to and from the Ethernet quickly). You should balance the load on each CPU so as not to detract from its communications processing needs.

File servers, units dedicated to file transfer services for host nodes, are ideal for optimizing the performance of file transfers and the utilization of the Ethernet. Other servers may also help offload specific processing activities so that the CPU can respond more effectively to the data coming from or going to the Ethernet.

As for the 9600 bps line, the bottleneck is less likely to be the CPUs. On the receiving end, a CPU will probably handle as much data as the line sends to it. On the transmitting end, the CPU can usually transmit with ease at the limit of the line's capacity. Thus, the workload on the CPU need not be a major concern here.

Making Careful Observations and Analyses

As discussed previously, resource utilization is a time-sensitive measure. If you measure a resource's utilization during an atypical time, it will not reflect the resource's usual utilization.

Another consideration is whether the resource is dedicated or shared. For example, CPU utilization studies must take account of how many users or activities are consuming CPU

time simultaneously. Measurement of CPU utilization is simpler for a single-task, single-user system. But in a multiuser or multitask system, more discrete measurements are necessary.

Digital's VAX/VMS systems include software that provides such measurements. Called VAX/VMS Image Accounting, this software records resource usage during each user session. It also records resource usage for each program image activated by each user, including I/O counts. This provides very useful statistics about workloads and how, for example, certain applications impact system performance and network performance.

One final point: vendors often quote "average utilizations." For example, someone may tout a LAN as having an average utilization of only 3%. This attempts to portray the LAN as having tremendous bandwidth. What is *not* said is how the LAN performs under peak loads. What good is an average utilization of 3% if, at peak loads, the LAN is saturated?

WHAT LEVEL OF UTILIZATION USERS EXPECT

The level of utilization that brings optimum performance for users depends on the nature of the resources under consideration. For example, a CPU that is transmitting data to, or receiving data from, a low-speed line can afford to run at high utilization. It will have extra time for processing other tasks while waiting to transmit or receive data from the slow line.

Shared resources perform best when their utilization is low. Some examples:

- When a multiuser CPU is lightly loaded, response times are shortest. When heavily loaded, responsiveness degrades.
- When the traffic on a multipoint line becomes heavy, users have longer waits before they are polled.
- When the load on an Ethernet reaches a very high point, collisions become more frequent and performance may degrade.

A dedicated line can afford higher utilization, mainly because the person paying for the line cannot afford lower utilization! (You are paying for it, no matter how much you use it. Thus, you should make the most use of it.)

The optimum level of utilization also depends on the function of the resource. For example, low utilization is desirable for the CPU that is used as a backup. High utilization is desirable for a line used for file transfers and other similar applications.

WHAT LEVEL OF UTILIZATION SHOULD BE EXPECTED

The practical level of resource utilization often involves a tradeoff between performance and cost. Maintaining low utilization may ensure higher performance, but it also means (1) the component is seldom needed, or (2) that it has to be very efficient or powerful. High utilization ensures maximum use of equipment, but it does not necessarily ensure highest performance. If you tune the network properly, the equipment performs best at high utilization. If not tuned, the result can be high utilization with long queuing delays.

The 100% Utilization Myth

For some resources, high utilization is undesirable. For example, an Ethernet, a timesharing computer system, or other shared resources perform best when utilization is low. When such a resource becomes heavily loaded, performance to all users degrades.

For other resources, high utilization is very desirable, but it is not always attainable. For most resources, 100% utilization is impossible. For example, as the workload applied to a CPU increases, the CPU will eventually reach its saturation point. The saturation point occurs below 100% utilization, sometimes as low as 90% or even 80%. This is because one part of the system often has to wait for another part to finish before it can proceed. For example, a CPU has internal housekeeping functions that delay the user tasks. It also has to wait for comparatively slow memory accesses.

As CPU utilization increases, queues may begin to grow. If the CPU is a node in a network, long queuing delays may occur for data needing transmission onto the network. This may impede responsiveness and throughput. If the node is a router, overly high utilization may impose large routing delays for data passed through it. In any case, the saturated CPU eventually becomes a bottleneck. In effect, *the challenge is to find a balance between delay (response time) and utilization.*

Notice there is no universal relationship between utilization, throughput, and response time. For a CPU, response time usually depends on lower utilization; however, throughput does not necessarily depend on higher utilization. You can increase throughput with relatively low utilization, such as with use of an efficient protocol.

Often the utilization of the high-capacity components of a network is limited by the lower capacity components. For example, one reason that the Ethernet cable will not attain 100% utilization may be the slower speeds of the access devices attached to the cable.

Utilization and Capacity Planning

You have to plan for peak loads, not just for the average load. You can feel safe with very high utilization only when no upward variations (peaks) in data traffic are expected.

Many network managers plan for an average network utilization of about 30%. This gives a good balance for both throughput and response time, and it ensures that bandwidth is available for peak traffic.

Again, the right utilization depends on how much you are willing to spend for performance. A tight budget can push the utilization requirements up, at the expense of performance. Sometimes you have to sacrifice performance at peak loads in favor of costs.

CPU OR NODE UTILIZATION

A CPU is either busy or idle. As already stated, CPU utilization measures the percentage of time that the CPU is busy. Often CPU utilization is expressed in terms of CPU cycles: the percentage of the available CPU cycles used in some unit of time (a CPU cycle is one tick of the CPU's internal clock).

The effective utilization of a CPU measures how much of the computer's capacity is being used. As such, it tells how much is available for additional activities or increased workloads.

How to Find Out CPU Utilization

Often commands or utilities are available on the operating system to compute the CPU utilization. To help monitor CPU utilization, most systems run a program whenever the CPU is not performing user or system work.

On VAX/VMS systems, this program is called the NULL task. The CPU idle time is the time spent in the NULL task. You can command the VAX/VMS Monitor utility to display the percentage of CPU time consumed by various processes, including the NULL task. From that you can deduce what the CPU utilization is for all user or system work. For example, if the NULL task measures 65%, then CPU utilization is 35%.

The net or effective utilization is more difficult to assess. However, if a monitor is available (such as VMS's Monitor utility), you can inspect the utilization of specific user tasks. This will help you figure the net utilization for each task because the utility also shows you the utilization consumed by overhead processing.

You should *not* think of CPU utilization as the number of users supported by the CPU. Such a measure of utilization is very unreliable. For instance, the amount of CPU consumed by 3 users doing tasks that demand much CPU time could be much more than the amount consumed by 30 users doing simple tasks.

An earlier discussion mentioned that more discrete measurements are necessary to give a complete and accurate picture of the CPU's utilization, such as those provided by

VAX/VMS' Image Accounting program. Appendix D gives an overview of the major ones. (Also, see [5].)

How Much CPU the Network Consumes

A major concern when configuring a network is how much CPU processing time and memory the network will require. A CPU can become a bottleneck if it does not have enough capacity to handle the network activity demanded of it. For example, a DECnet node that is a general-purpose processor must handle:

- Operating system overhead
- DECnet software
- Communications interrupt processing
- Routing
- User applications, with or without disk access (some applications may or may not use the network).

Figure 7–1 shows some of the many utilities, functions, and tasks handled by a CPU in a typical VAX/VMS system.

Assess CPU utilization for all CPUs along a path. On each CPU, user requirements for local and network processing should be significantly less than capacity. Depending on the system, the ideal utilization may be anywhere between 30% to 60%. When the utilization exceeds that level, network performance can deteriorate. This may mean, among other things, that a particular application is stealing much of the CPU time. If so, the application may need to be redesigned to use the CPU more efficiently.

When CPU utilization is very high, you must be sure that the load remains stable. If the load increases suddenly, even with just a slight burst of activity, performance could be affected negatively. You should make no additional demands of the CPU without monitoring the impact on performance. Tools such as the VMS Monitor utility measure the amount of CPU consumed by network processes.

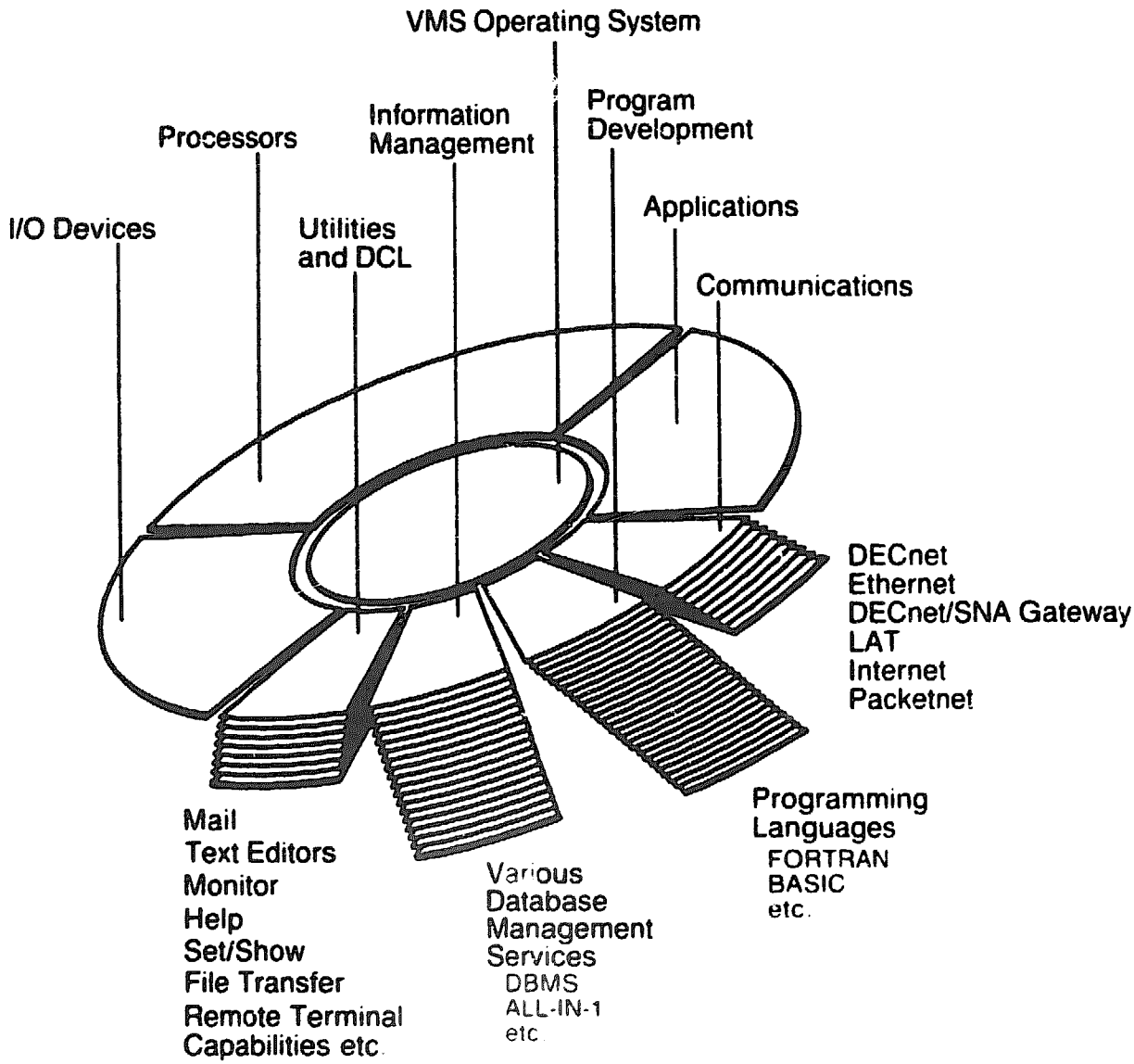


Figure 7-1 Various Activities Performed by a Node's CPU

When local processing is of high priority and in great demand on a CPU, you can use front-end communications processors to take care of network tasks. In this way, the host CPU can concentrate on the local processing without sacrificing the performance of network tasks.

Because the front end controls communications functions independently, it reduces processing time and main memory requirements in the host CPU. Also, because the front end has its own peripherals, it can store a portion of the host's load. This is especially important during peak load situations and can help smooth system and network throughput.

In LANs, you can use dedicated communications servers as front ends for a group of hosts. For example, a terminal server can handle terminal I/O for many hosts; a dedicated router can offload routing overhead for the nodes it serves. For more information, see Appendix A.

Factors That Affect CPU Utilization

Two major influences on CPU utilization are CPU capacity (power) and the applied workload. CPU power is the speed at which a CPU can process a task. CPU utilization is inversely proportional to CPU power. The more powerful the CPU, the less CPU time is required to process a task. Thus, under the same load, the utilization is less for the more powerful CPU. The more powerful CPU has more time available for other functions.

CPU power is often the greatest of all resources in a network. A CPU is usually faster than other components used by the network. Thus, other components are more likely to become bottlenecks than the CPU. These components include disk drives, tape drives, lines (low-speed), device controllers, and the protocol software. Whenever the bottleneck is on the input side, the CPU may have to wait. Its utilization decreases relative to the utilization of other devices.

Whether the CPU becomes a bottleneck depends on the applied workload relative to its capacity. As the workload applied to the CPU increases, more CPU time is consumed (CPU utilization increases).

Communications Devices

No matter how powerful the CPU, it may not be able to overcome the bottleneck imposed by a slow, inefficient device. Interrupt-intensive devices, such as lab peripherals and timesharing terminals, can consume a large portion of CPU time. As mentioned before, direct memory access (DMA) devices save CPU time.

Figure 7–2 shows how a character-interrupt device and a DMA device may differ in their demand on a CPU. Here, CPU utilization is lower when the CPU is connected by a DMA device. However, this may not always be true. The application software could be the limiting factor, perhaps because it requires an excess of I/O's, causing the application to become disk-bound. In such a case, CPU utilization would be the same with either device.

Message Size

The curves in Figure 7–2 also show that CPU utilization decreases as message size increases. (This assumes you have a large amount of data to send.) If the message size is large, the CPU has to process fewer messages for that amount of data than if the message size is small.

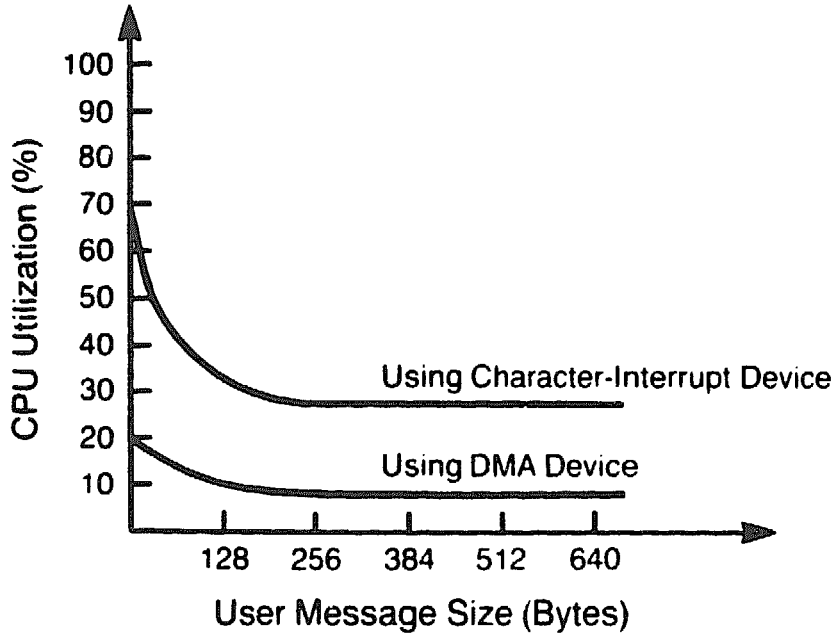


Figure 7–2 Effects of Devices and Message Size on CPU Utilization

Message Rate

The **message rate** is the number of messages given to the CPU to transmit over a line, or received by the CPU from a line, within a given period of time. As message rate increases, CPU utilization increases (see Figure 7–3). If the message rate is greater than or equal to the CPU's service rate, queues will build, and the CPU can become a bottleneck.

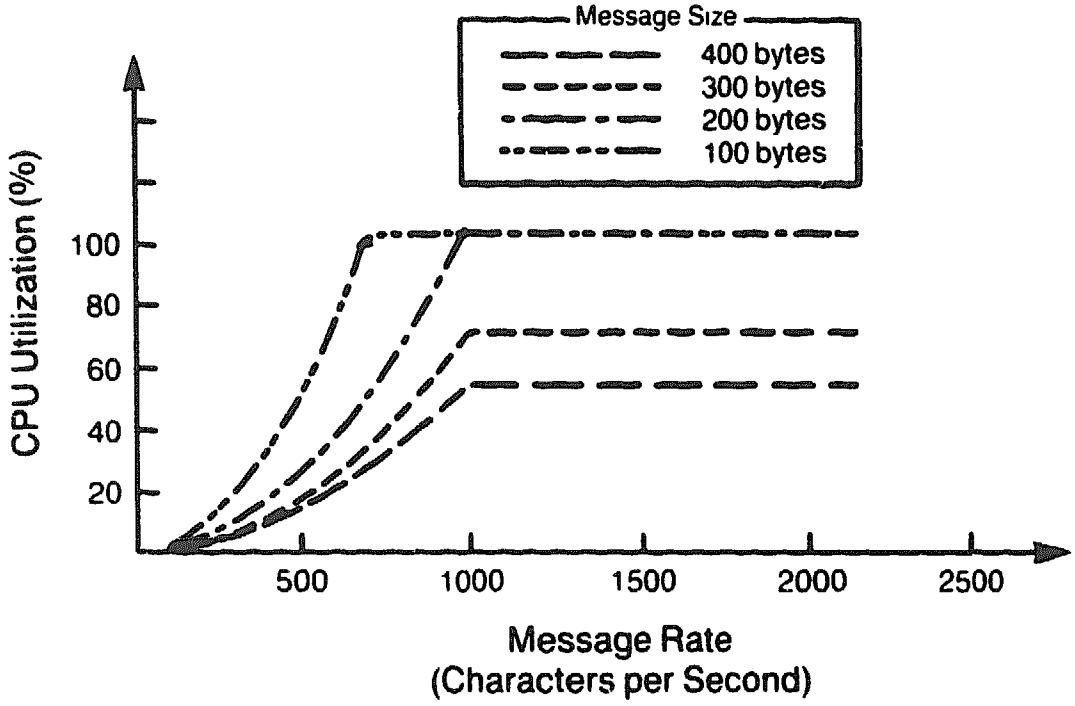


Figure 7–3 Effects of Message Rate on CPU Utilization

Capacity Planning for Network CPUs

The proper sizing of CPUs for a network depends on the level of activity you expect of them. In particular, you should consider the following:

- Message rate
- Distribution of message sizes (at least the average size)
- Average CPU cost to send and receive network data
- Average CPU cost for local processing

The CPU cost is the CPU utilization. The discrete measures discussed in Appendix D can help determine the CPU cost for network and local processing.

The speed of the CPU is important in determining the throughput rates possible in the network. It should be high enough to maximize the usage of network lines. In this way, the CPU will not be the bottleneck.

When several communications lines are connected to a CPU, you should consider their combined throughput to make sure the CPU does not become overloaded.

CPU memory is also important. The memory speed determines the CPU's processing speed, such as for I/O transfers related to network communications. Also, the memory size must be large enough to support, among other things, the buffering required for local and network processing.

Improving CPU Utilization

Improving CPU utilization usually involves reducing the demand placed upon the CPU. If a node must perform a significant amount of network tasks, you should minimize the demands from local tasks, or offload some of the communications processing. You can further reduce the demand on the CPU by:

- Tuning the system and/or network (as mentioned earlier, this includes selecting the appropriate devices, buffer sizes, quotas, and so forth.)
- Scheduling the workload so that fewer compute-bound processes run concurrently.
- Redesigning some applications. For example, you can improve algorithms so that tasks require less processing. When you improve frequently-run programs, the saving on CPU cycles can be significant.
- Controlling concurrent demand for terminal I/O on the CPU and using good terminal I/O techniques. In LANs, for example, terminal servers could replace character-interrupt line-control devices. Digital's terminal servers also provide **load balancing**, which means that if more than one CPU connected to the server offers the same service, the server connects the terminal user to the CPU with the least load.

If none of the above methods helps, then you may need to add CPU capacity. But where in the network do you increase CPU capacity? Consider a single-stream environment where pieces of work are processed in succession by several nodes. Nodes at the end of the line ("downstream") depend heavily on the rate at which preceding nodes process their portions of the work. If those preceding nodes are too slow, they will hold up production. Thus, extra CPU capacity is most needed at those "upstream" points in the network.

To add CPU capacity at a point in the network, you may choose to add one or more CPUs at that point, instead of just increasing the capacity of the current CPU. It is best to add CPUs when workloads consist of independent jobs and data structures, and when system availability is critical. The added CPU's can be faster, the same, or even slower than the original CPU. In any case, they will offload work from the original CPU and serve as backups when the original CPU is unavailable.

In this regard, you can employ multiprocessing techniques or install clusters (VAXclusters). **Multiprocessing** consists of linking two or more CPUs together to share resources efficiently. Resources can be CPU cycles, specialized peripherals, or disk memory. An application can be split into components that interact by sharing databases or exchanging messages. Also, these application components can be designed to exchange messages or share data in the network environment.

LINE UTILIZATION

Gross line utilization is the percentage of a line's bandwidth that is used. For lines that serve as vital links within a network, a crucial issue is keeping enough bandwidth available for handling the largest demands placed on them. For lines owned by a common carrier, the main issue may be how efficiently you can use the line for any given transmission (assuming that service charges are based on the length of time that the line is used or the amount of data transferred). For dedicated lines, the issue is whether there is enough demand to warrant the cost of renting the line.

Net line utilization is a very important measure. The value of protocols and communications devices depends on the level of net line utilization they can attain (the higher the better). For most protocols, net line utilization (also called line efficiency levels) reaches a maximum of around 70%. For those protocols that require a large number of protocol exchanges per message, the net line utilization is significantly lower. For example, some protocols, such as IBM's Binary Synchronous Communications protocol (BSC), require an acknowledgment for every message sent. DECnet's Digital Data Communications Message Protocol (DDCMP) and IBM's Synchronous Data Link Control (SDLC) protocol improve net line utilization by allowing a block of user messages to be pipelined before requiring an acknowledgment. DECnet's DDCMP allows a maximum of 255 messages at a time, while SDLC allows up to 8 messages.

Another technique employed by Digital to reduce protocol overhead in the Network Service Protocol (NSP) is **piggybacking:** taking protocol control messages and embedding them as part of the protocol header on user messages. Instead of sending protocol messages separately, the system holds the protocol message so that it can take a free ride on the next user message headed for the same destination.

The X.25 protocol, one of the more efficient protocols, uses these features to reduce overhead. It can attain net line utilization levels as high as 90 to 95%.

When rating communications products by their maximum attainable line utilization, it is important to be sure whether the rating is gross or net line utilization. Also, when considering figures about line utilization, be clear about the conditions for attaining these figures. Was the traffic intensity at a low level, or at the level with which you are concerned? What types of processors were at each end of the line? CPUs with higher speeds can transmit at higher speeds and thus maintain higher line utilization.

How to Calculate Line Utilization

To calculate the net or effective line utilization, divide the real throughput by the line's bandwidth or line speed. Multiply that by 100%. For example, if 1200 characters (8 bits per character) take two seconds to transmit over a 9600 bps synchronous line, then:

Throughput = (8 bits x 1200 characters) / 2s = 9600 bits/2s = 4800 bps

Net line utilization = (4800/9600) x 100% = 50%

Monitors are available to determine the net line utilization. They measure the amount of user data sent over the line. You can measure the amount of user data sent over the line for one pair of applications or for several pairs of applications.

Line Bandwidth

As with CPUs, you should plan for a line bandwidth that can accommodate the demands of peak traffic. Typically, the bandwidth (capacity) of a single, synchronous communications line is far less than that of the CPUs it connects, so the line becomes the bottleneck. For example, say your line utilization is 90 or 100%, while the utilization of the CPUs at either end is only 30%. Then if you want more throughput, a faster line would be appropriate for the link.

For optimum response times in applications exchanging small messages intermittently, line utilization should not exceed about 50%. Beyond that level, the delay increases very sharply, as shown in Figure 7-4, because an average of one user or request-for-service is already waiting in a queue at any given time. When the line is lightly loaded, the average delay is equivalent to the average response time.

For applications requiring bulk transfers of data, optimum throughput occurs at 100% line utilization. The higher line utilization means that more data is being transmitted over the line.

Figure 7-5 shows several throughput curves for lines of different speeds. Notice the marked difference between the 1 Mbps line and the other lines. Obviously, a 10Mbps Ethernet would show an even greater difference.

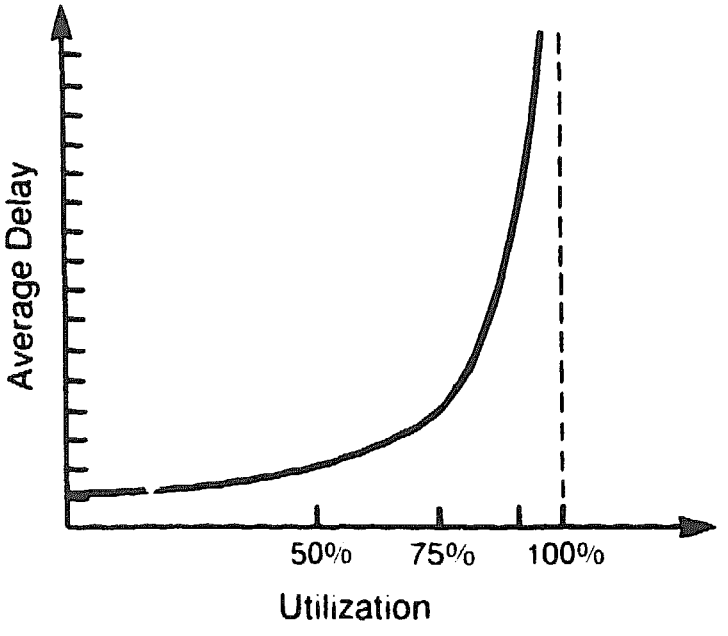


Figure 7-4 Line Utilization and Delay

Factors That Affect Line Utilization

Many of the same factors that affect CPU utilization also affect line utilization. This section discusses these and others.

Line Capacity and Workload

Two major factors affecting line utilization are the capacity of the line and the workload applied to the line. The greater the bandwidth, the less a given workload will affect the line utilization.

Communications Devices

The type of communications device affects line utilization. For networks using telephone lines for communications, modems can be the critical factor in performance. High-speed modems transfer more data; throughput is potentially higher. However, high-speed

modems may be less tolerant of noise and other line problems. Error rates can be high, requiring frequent retransmissions and, thus, degrading the net line utilization.

The type of line interface also plays a critical part, such as whether it supports asynchronous or synchronous transmissions, and whether it is character-interrupt or DMA (see Appendix A).

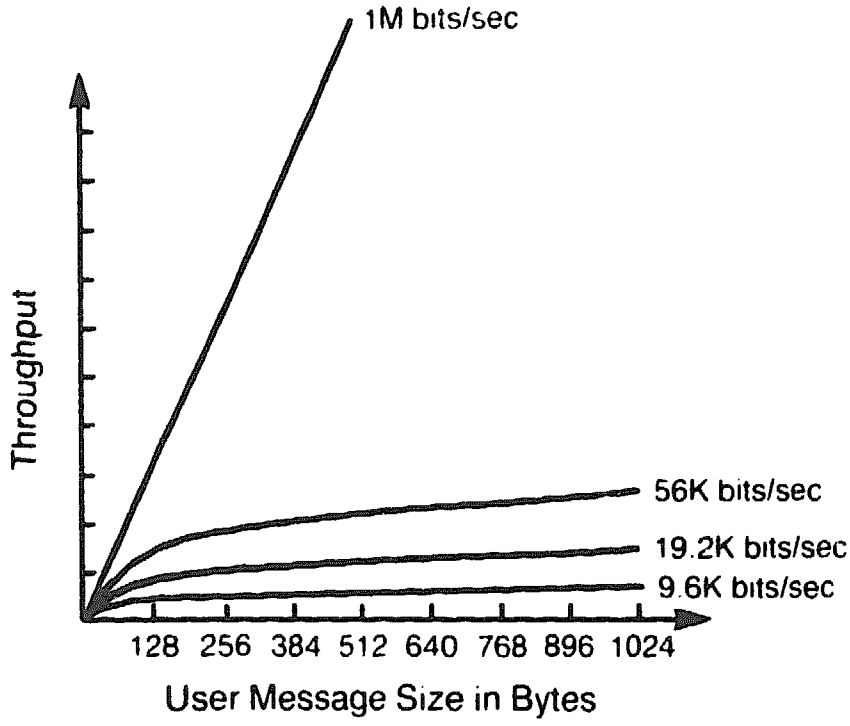


Figure 7-5 Throughput and Line Speed

Line Noise and Transmission Errors

Line problems in transmission can affect net line utilization significantly. The noisier the line, the more retransmissions and protocol messages must be sent. You should transmit only small messages over such lines.

Message Size

As Figure 7-6 shows, net line utilization improves as the user message size increases, especially over faster lines. Smaller message sizes incur more overhead in relation to user data.

Control Overhead

A line can transmit a limited number of bits, and so the protocol bits needed for line control detract from the effective rate of information transfer. The more protocol characters sent, the less the net line utilization.

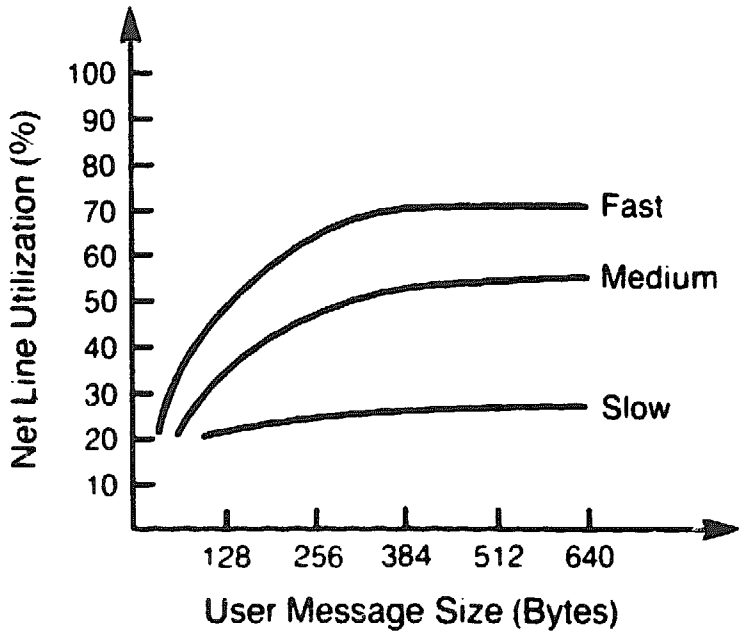


Figure 7-6 Net Line Utilization and User Message Size

Acknowledgment Handling

If each message requires a separate acknowledgment, there is that much more overhead. Furthermore, if the acknowledgment message is separate from the rest of the data, then the transmission of messages involves even more overhead. Protocol pipelining and piggybacking techniques can resolve these inefficiencies.

Note that Ethernet transmissions do not require acknowledgments. Each transmission is assumed successful unless a collision is detected.

Improving Line Utilization

A major way to improve the utilization of high-speed lines is to increase the throughput of the CPUs that use them. This may involve increasing the capacity of the CPU, supplementing it with additional hardware devices (such as communications servers or multiplexers), increasing the efficiency of the CPU, or redesigning the applications to optimize CPU utilization.

You can improve the utilization of low-speed lines especially by using more efficient protocols. Applications using these lines should employ multibuffering when possible, instead of relying on command-response ("ping-pong") approaches. Because the line is slow, the latter approach reduces the throughput of the line. You should schedule traffic so that it is less likely to arrive in bursts (traffic bursts cause bottlenecks on low-speed lines). By maintaining an even flow of data, the traffic flows with fewer backups or queuing delays. Using multiplexers can help maintain even traffic flows.

Multiplexers and concentrators improve line utilization significantly by combining the transmitted data of several users onto one line. Thus, several users can share an underutilized line (or an expensive one). Theoretically, a 9600 bps voice-grade line can transmit 17,280,000 bits in 30 minutes. With only one keyboard terminal connected to that line, only a few thousand bits are sent during a 30-minute session (using less than 1% of the line's bandwidth during that time). Wasting bandwidth is especially costly when the line is being leased. Thus, it is wiser to multiplex several terminals onto such a line.

MEMORY UTILIZATION

Memory can be a tremendous factor in both response time and throughput. To participate in a network, a node requires sufficient memory, both main memory and secondary. The amount of memory needed depends on the number of network services provided at the node. DECnet full-function routing nodes, for example, require more memory than non-routing nodes. In larger networks, nodes need larger memories to maintain the node database. The node database contains information about all the nodes in the network (or in the area, if it is a DECnet multiple-area network).

Nodes with small CPUs and less available memory, such as personal computers, require more disk I/O, and the access time from disks is much slower than from main memory. Memory considerations are important for large CPUs. If not enough memory is allocated, the CPU can become a bottleneck when traffic demands at the CPU require more buffers than the allocated memory can hold.

The system manager controls how much memory is allocated. The amount depends on the degree of performance required of the network. You can also look at this the other way around: the performance of a network depends on the amount of memory available at each node. Memory can be a limiting factor: if available memory is too small, not enough buffers are available for communications. This can degrade performance.

Insufficient memory causes a system to swap processes and/or pages. (The following discussion applies to VAX/VMS and other operating systems of similar design.) A process is a scheduled entity on an operating system. A page is a group of bytes stored in memory or on disks. Memory can hold a limited number of pages at a time. When the demand for pages exceeds those available in memory, a page fault occurs. Certain pages are swapped between the CPU's main memory and disks. The disk I/Os that this requires

may hurt performance (disk I/Os being relatively slow). Keep in mind that a process cannot work until it is in memory. Thus, the entire system is affected by the disk I/O. This does not mean that system managers should allocate memory indiscriminately. But when a CPU bogs down, insufficient memory is often the reason.

Another memory consideration relating to performance is the number of processes running on a CPU. If that number is large, the network processes must wait longer for the CPU to service them. If the amount of memory is small, network performance will be very poor. With more memory, the CPU can handle the processes quicker because there is less page-faulting and process-swapping.

Even when sufficient memory is available, improper system design can cause it to be inefficient. An example of inefficient memory usage is the copying of entire processes in and out of memory instead of just the needed pages.

A system that uses memory efficiently keeps only the most frequently-used programs in memory. With paging, the most frequently used (most recently used) information stays in main memory (instead of on disk, for example) to save access time.

DISK UTILIZATION

Disk activity can affect network performance significantly. Important for disks is the volume of data that it can store and the access delay. For best performance, you should select faster disks where needed.

A measure of a disk's speed or capacity is its **access bandwidth**, expressed in time as megabytes per millisecond. This measures the speed at which you can access data on the disk. It also considers the volume of data stored. If you double the volume of data stored, while the access delay remains the same, then the access bandwidth doubles: you can get twice as much data in the same period of time. The access bandwidth also doubles if the access delay is halved while the volume remains constant.

Good disk management techniques can help optimize performance. The key to high disk performance is the use of fast buffer memory. Also, users should balance disk activity so that a single disk drive is not overworked. If a drive is overworked, users can move some files or directories to other disks.

Effective systems must have balanced processing and I/O capacities. For high performance at low cost, you can use certain techniques to improve disk performance:

- Allocating the best location of data on the disk surfaces.

  The most frequently accessed data can be clustered into a small, easily accessible area. Data used sequentially can be stored on the disk surface sequentially and read/written in large contiguous pieces rather than in individual blocks. This reduces head movement (which hurts performance).

- Buffering the most frequently used data in faster storage areas (called **caching**).

  Data caching involves keeping a smaller buffer of frequently-used data where it is needed. Data that is permanently resident on a disk can be temporarily cached in the CPU's main memory where access is much quicker. Caching can dramatically reduce the average access time for network applications that must access remote disks.

NETWORK UTILIZATION

Network utilization is the ratio of total data flow to the theoretical maximum data flow. More specifically, it is the amount of traffic a network successfully delivers relative to the maximum amount of traffic the network can carry with perfect scheduling.

For an efficient network, channels with the greatest bandwidth should carry a larger percentage of the total traffic volume than channels with smaller bandwidths. See Case 1 of Figure 7-7.

Two major signs of channel *inefficiency* are:

1. Significant discrepancies between a channel's share of the network's total data flow and a channel's share of the network's total flow capacity.

2. Variability among channels in their flow/capacity inefficiencies.

The former is a sign that a particular channel is being used inefficiently. For example, if a channel's share of the total network flow is 10%, while its share of the total network capacity is 40%, as with channel "e" in Case 2 of Figure 7-7, then a large portion of that channel's bandwidth is wasted. A large amount of the flow in the network is being handled by other channels with less capacity to handle it (channels "a," "b," and "d").

On the other hand, if a channel's share of the total network flow is 40%, while its share of the total network capacity is 20% (as with channel "d" of Case 2 in Figure 7-7), then that channel is overloaded, and other channels with higher bandwidths are probably being underutilized. A large flow/capacity ratio at one channel will probably cause a large capacity/flow ratio elsewhere.

If the flow/capacity ratios of all the channels vary significantly, the network is inefficient. Perhaps planning was not thorough enough, and the traffic was overestimated or

underestimated at several points in the network. Or, perhaps unexpected usage patterns developed once the network was installed or once new uses of the network were discovered by users.

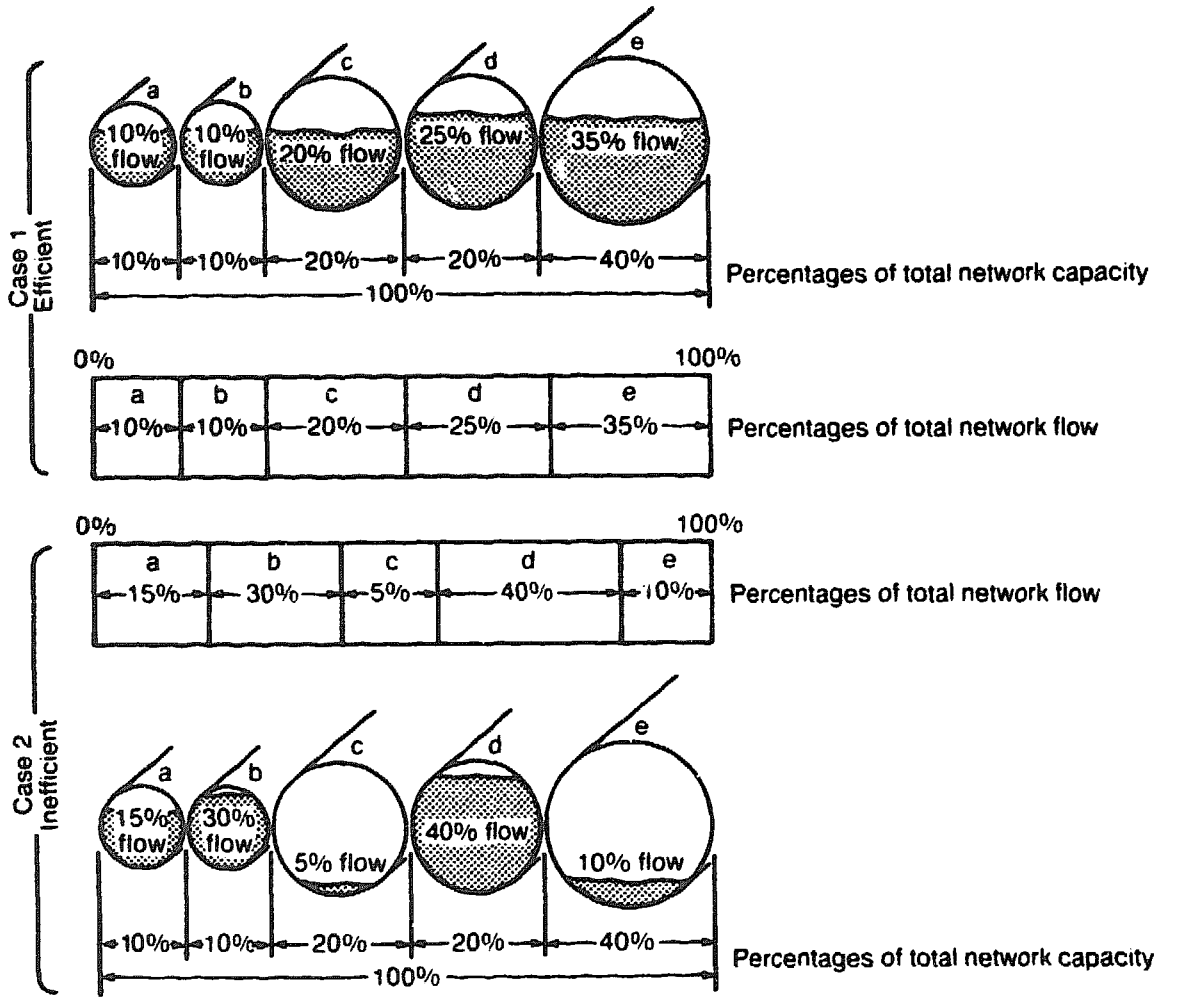


Figure 7-7 Network Efficiency and Flow/Capacity Ratios

There is a useful index that you can use to describe the efficiency of a network (refer to [6]). The index is based on a technique suggested for budget planning (by Henri Theil,

in *The Accounting Review*, January 1969). This index measures the success of a company's budget planning, and it is based on the ratios of actual expenditures to planned expenditures within departments of a company. Actual expenditures are analogous to actual traffic flow, and planned expenditures are analogous to traffic capacity.

The first step to improving the efficiency of a network is reducing the overall disparity rather than concentrating on changing flow/capacity ratios for individual channels.

Network utilization is more complicated to assess for networks with multiple lines. To measure the available bandwidths of a network's channels, note the following:

- When one or more alternate lines are available between two nodes, consider the bandwidth of the line that is most likely to be used. (In DECnet networks, for example, the line with the least cost is used more often.)

- When considering a single chain of lines, the maximum bandwidth depends on the weakest link.

A major factor that can degrade network utilization is the inefficiency associated with packet overheads, such as the extra bits needed for addressing, access information, and synchronization. (This is a design issue: one that you should consider before purchasing a product.)

All the factors that influence node and line utilization also affect network utilization. See the discussion of those topics (message size/rate, memory size of nodes, and so forth) earlier in this chapter.

Network utilization benefits greatly from the use of flow control mechanisms. For example, window flow control prevents a node's CPU from overloading. The CPU sends messages to coordinate the flow of data sent from other nodes, preventing the data from arriving at a faster rate than the CPU can handle. When data comes too fast, the window "shuts," in which case the CPU signals the transmitting node to stop sending further data. This also prevents lost data. Without flow control, buffers can overflow and performance can degrade.

When a CPU or some other part of the network becomes saturated or congested, or when it fails altogether, traffic can be rerouted to bypass it, or other traffic headed for that point can be inhibited or delayed. The former technique is more desirable, since the data transfer continues and users are less affected.

By plotting the utilization of a resource over a period of time, you can monitor:

1. Whether all components are at or near peak performance levels.

2. Where, when, and why any bottlenecks occur.

3. Correlations between two or more resources, or between certain application tasks and bottlenecks.

4. The fraction of time two resources are busy at the same time. This shows, for example, how much of a particular application is processed concurrently.

You can repeat the above procedure for different software, processes, and so forth, noting any differences in resource utilization. Then you can seek answers to such questions as:

- Is there a particular process that slows down the performance of a resource?
- Is there a process that demands a disproportionate amount of a resource's capacity or that generates the majority of the load on a bottleneck?
- How can the equipment be better configured to get maximum returns?
- Can certain processes be distributed over various resources so as to reduce the demand on an overloaded resource?
- Can the bottleneck be moved to another resource to improve network performance?

IMPROVING UTILIZATION – A GENERAL VIEW

When a component's utilization is too high, you can improve performance by upgrading the overloaded component, adding another component to share the load, or adjusting the load applied to that component. When utilization is too low, you can swap the component with another, less powerful model. Perhaps the underutilized component can be moved to another point in the network where there are higher traffic demands.

In general, high-capacity components should carry relatively larger portions of the total network traffic. Lower-capacity components should carry less. When you see that certain components are over-used while others are under-used, you should see if they can be reconfigured, or if the load throughout the network can be redistributed.

One feature of DECnet that can improve the load balance over the network is **path splitting.** Here, a node splits the load over several paths to a particular destination (each path must have the same path cost). This helps prevent any one line from becoming congested, and thus ensures greater reliability for the communications service. You can also use path splitting to optimize the cost-effectiveness and line utilization of several low-speed lines connecting to the same destination. The combined throughput over the lines can equal that attained over a single, expensive high-speed line [1].

Load balancing is another feature supplied by some of Digital's products, such as the Ethernet terminal servers, to distribute the load on network components.

SUMMARY

The resources of a network are the "engines" and "cogwheels" that comprise the network mechanism. If they are efficiently used, you can optimize network performance along with the returns gained on their investments.

For maximum responsiveness, line speeds and CPU sizes should be configured so that their average utilization measures are low enough to leave "room" to accommodate peak demands. Response times benefit from low resource utilization. For example, when CPU utilization is low, it can quickly attend to a user's request or to a network I/O. When utilization is high, the CPU is likely to be too busy to handle new requests immediately. Therefore, new requests have to be queued, and users have to wait longer.

For maximum throughput, high line utilization is desirable. The more data being transmitted on a line, the more throughput will be achieved, assuming that the CPUs at either end are sized appropriately to maintain the flow. High utilization is acceptable for most dedicated, independent, or loosely-coupled systems. Low utilization is better for shared systems or for tightly-coupled systems where a high demand on one component will affect another.

The best network utilization for optimum performance is one where all resources handle loads proportionate to their capacities. An inefficient network has many resources that are overutilized and underutilized.

PART 3
Appendixes

APPENDIX A
BASIC NETWORK TECHNOLOGIES

MAKING THE CONNECTION

There are several types of physical links that you can use to put networks together. The two most common are:

- Point-to-point.
- Multi-access, such as Ethernet, token bus, or token ring.

Point-to-point connections may be fixed, as on a private line, or switched, as on a dialup telephone line. With respect to performance, two main differences between private and switched lines are in the (1) connection time required and (2) reliability or flexibility.

1. **Connection time requirements:** On a private line (also called a leased or dedicated line), the connection is always available. You do not have to dial the destination and wait for a path to be established as you do on a switched line.

2. **Flexibility and reliability:** Switched lines allow you to connect to any destination with the appropriate modem, while on a leased line you are limited to one destination. When a circuit fails on a switched line, you can redial to establish a new connection (if the circuit is not busy and if the line quality is good enough to maintain the connection). When a private line fails, the results are usually more drastic. But private lines are usually conditioned, which makes them more reliable for data communications (less prone to errors and retransmissions). On switched lines the physical circuits chosen vary from call to call, depending on their availability, making connections less predictable. Switched lines cannot be conditioned, therefore, and are thus more difficult to optimize for performance purposes.

Two types of switching that significantly affect network performance are circuit switching (used by telephone companies) and **packet switching** (user data is introduced into the network in packets rather than in a continuous stream).

Packet switching can use communications channels more efficiently than circuit switching. Each packet occupies a channel only for the duration of its transmission. Thus, each channel has higher availability: several pairs of communicators can use a channel

simultaneously. This improves line utilization. Packet switching is used in Packet Switching Data Networks (PSDNs), a service bought from a common carrier.

Tariffs for PSDNs are based on the volume of data transferred, rather than on the distance or connect-time between the communicating nodes (typical of circuit switching networks). Thus, PSDNs are cost-effective for applications with low to medium volumes of traffic between nodes.

THE TRANSMISSION MEDIA

The transmission media say a lot about the performance potential of the network. For example, consider voice-grade telephone lines. Telephone systems have been optimized for voice signals, not for digital signals transmitted by computers. The transmission capacity of telephone lines is smaller than that of other types of transmission media (such as coaxial cable and fiber optics).

It is possible to increase a channel's transmission capacity or throughput by using certain coding techniques for the transmitted signals (Appendix C). Modern communications equipment uses such coding techniques to increase the data throughput capacity of telephone lines, for example.

TOPOLOGY AND CONTROL

LANs are limited to three basic topologies:

1. Star
2. Ring
3. Bus

WANs can take on any number of topologies or structures. The larger the number of nodes contained by a WAN, the greater is the number of shapes it can take.

The proper choice of topology for a LAN or WAN is important for optimum performance of the network. The most suitable topology for network performance depends on three major factors:

1. How the nodes function in the network
2. Where the control lies
3. How much reliability is required

In a network where one or a few nodes function mostly as resource providers for a relatively large number of other nodes, a centralized structure is best. All nodes are subjugated to a central node that controls access to the network. In contrast, where all nodes function both as resource providers and resource users, as in most DECnet networks, a fully distributed structure is more appropriate. In such a structure, nodes have equal power relative to network access. They function as peers.

Regardless of the topology used, any shared transmission medium must allow access in an equitable manner. If users share any resources, you must use some control procedure to allocate those resources.

Examples of access control schemes include **polling**, **token passing**, and **contention control**. Ethernet LANs use a popular contention control scheme called **Carrier Sense Multiple Access with Collision Detection (CSMA/CD)**. Here access control is distributed fairly among all nodes.

The major performance drawback of a centralized topology or control scheme is the effect on the network's reliability. The central node is a "single point of failure," meaning that if that node fails, the entire network stops. Also, nodes must wait for permission from the central node before they can access the network. This wait can affect response times, especially when numerous nodes seek access.

In contrast, a distributed network has no central point of control. If any single node fails, the network usually continues to run smoothly. Also, distributed networks tend to be more responsive to local needs of nodes. Nodes do not have to wait for permission from a central node to access the network.

COMMUNICATIONS DEVICES

The time required by CPUs for communications I/O depends greatly on the abilities of the devices that manage communications over each line. These devices are called **line controllers**.

Character-Interrupt Versus DMA Devices

Two major types of line controllers are character-interrupt and DMA. The most significant difference between these two types of devices is the amount of time they demand from the CPU for communications processing. When data is ready for transmission or reception, the CPU prompts the communications controller to initiate the I/O processing. The character-interrupt device interrupts CPU processing for each character that it receives from the communications line and for each character it is ready to transmit. At each interrupt, the CPU looks at the character and either (1) stores it, if

received from the line, or (2) retrieves it from memory, if it is to be transmitted. When large amounts of data are queued for transmission or reception, the excessive number of character interrupts consumes considerable CPU time and limits character throughput.

In contrast, a DMA device does not interrupt the CPU for each character. Instead, it transfers I/O data in blocks of characters, directly to or from memory. It requires CPU attention only after the entire message segment has been transferred. Because the DMA device offloads much I/O overhead from the CPU, the CPU can support more communications lines and, therefore, provide higher network throughput.

Asynchronous Versus Synchronous

Communications devices can support either asynchronous or synchronous transmission. **Asynchronous transmissions** are transmissions in which the time periods between character transmissions differ. Each character is surrounded by a start bit and stop bit, which helps the receiving device recognize the beginning and end of each character. These overhead bits can detract from user throughput, especially if large amounts of data are being transmitted (two bits added to each 8-bit character imposes an automatic 20% overhead. For this reason, asynchronous transmission is best for small, irregular transmissions at low speeds, such as with interactive terminal traffic.

Synchronous transmission is best suited for sending large amounts of data at high speeds. It does not include overhead bits on a character basis. Data is stored and sent in blocks, rather than one character at a time.

Synchronous transmission requires more expensive communications devices than asynchronous. If the volume of traffic does not warrant synchronous transmission, it is a wasted expense. Sending small amounts of data synchronously can result in the same overhead as asynchronous transmission.

Communications Servers

Communications servers are special, stand-alone systems dedicated to managing communications activities. Here, communications processing is removed from the original CPU (called the **host**) and delegated to another CPU (the **front end**). The benefit is two-fold. The front end processor is dedicated to communications processing, so it does the job more effectively than the host CPU. The front end allows the CPU to perform its local tasks more effectively by offloading the communications tasks from the host CPU.

Terminal servers handle terminal operations for host nodes on the LAN. The terminal server, by attaching terminals to it, connects terminal users to one or more computer systems on the LAN.

Without a terminal server, each terminal in a network usually has to access remote nodes through a local node (which must use much of its processing power to establish the connection between the terminals and remote nodes, and to handle the terminal activity). Introducing terminal servers to the LAN allows you to offload such processing from host nodes. Thus, the host nodes have more time for the application tasks they process.

Another performance-related feature of Digital's terminal servers is that they provide **load balancing:** when more than one node offers the same service requested by a terminal user, the server will connect to that node with the highest rating for the service desired. The rating is based on the current loading of the nodes that offer the service (the least-loaded nodes have the highest ratings).

Other communications servers available through Digital include:

- **Router servers,** which offload routing functions from CPUs that have to handle a significant amount of routing traffic as well as local processing. Digital provides router servers that handle synchronous as well as asynchronous connections. Asynchronous connections allow personal computers (and other systems for which high-cost connections would be unsuitable) to access the network.

- **Gateway servers,** which provide interfaces to different types of networks or systems, such as the X.25 PSDNs or the IBM SNA systems.

APPENDIX B
GRAPHING PERFORMANCE DATA

USEFUL WAYS TO ARRANGE AND GRAPH DATA

To get a good picture of a mass of recorded data, you can graph it or represent it by a curve. This allows you to readily grasp the essential features of the distribution and to compare it with other distributions. You can also represent data with a bar graph or a histogram. Figure B-1 shows a histogram based on the following group of response times: 1, 2, 2, 1, 2, 2, 5, 1, 1, 2, 5, 1, 1, 2, 2, 2. The height of each bar reflects the frequency of occurrences for the corresponding value on the horizontal axis (for example, 2 occurs eight times).

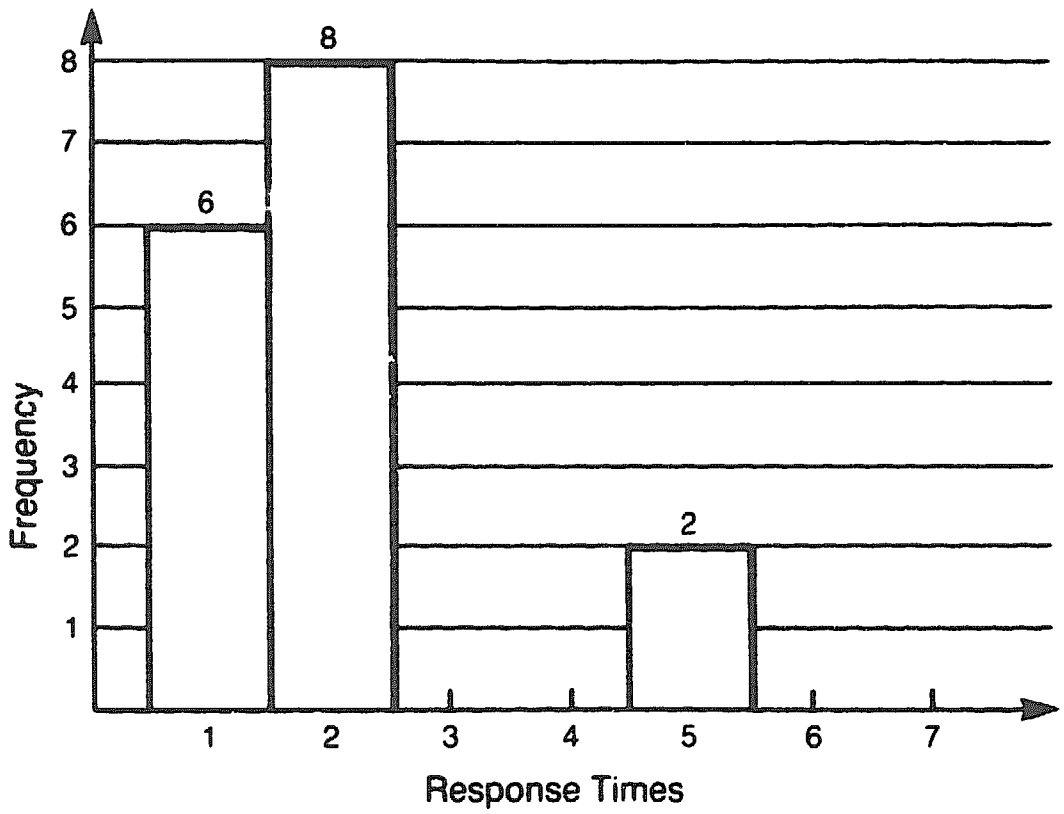


Figure B-1 Histogram

The shape of a histogram or curve tells much about the measures it represents. In Figure B-2 the shape of curves 1 and 2 are examples of **normal distributions** (bel -shaped). (Note that "normal" is not necessarily equivalent to "usual.") Curve 1 shows the response times clustered within a small range of values along the horizontal axis. Thus, the responsiveness of the network under consideration is consistent and predictable, characteristics which are pleasing to users. If the value L is the maximum acceptable response time for an application, then curve 1 shows ideal network performance, since all recorded times fall below that value. The maximum acceptable value specified for a performance measure is called the **specification limit**.

In curve 2, the results are more widely distributed and thus less predictable. Since a significant number of response times exceed the specification limit (L), such results are unacceptable.

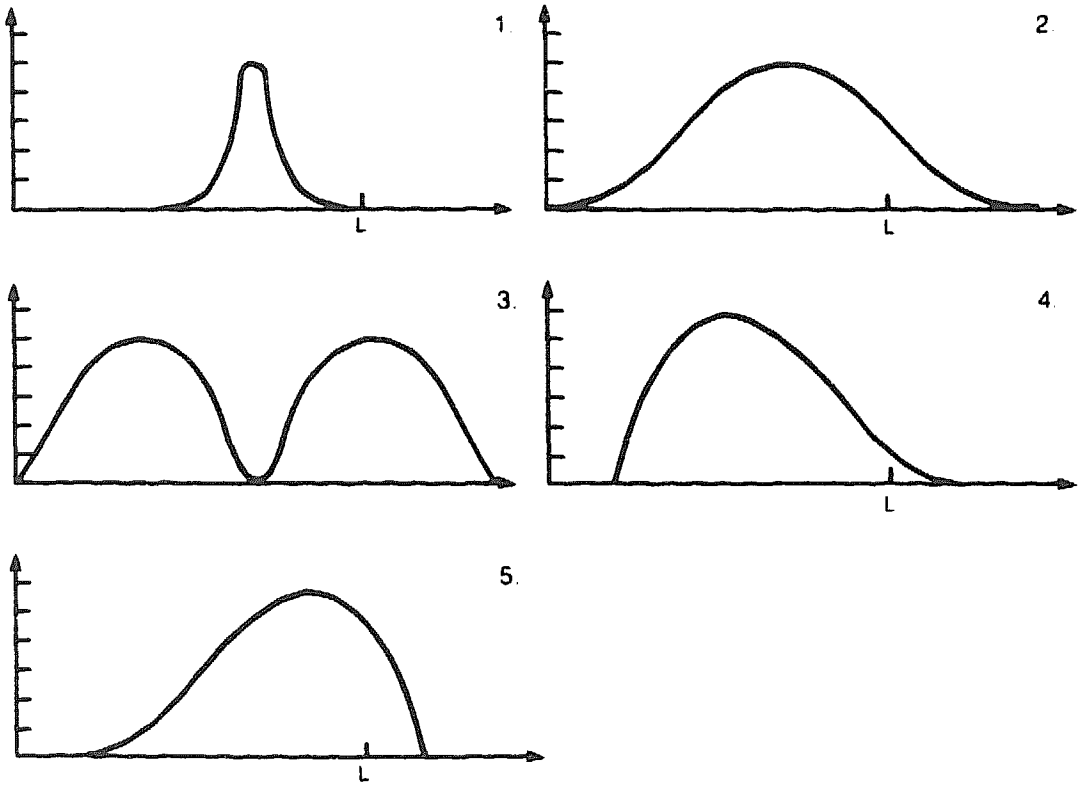


Figure B-2 Various Distributions

Curve 3 is **bimodal**. Such a curve could occur when the frequency distributions of two applications with different mean response times are represented in a single graph.

Response times often have a skewed distribution. Curve 4 is a positively skewed distribution. A large number of response times occur within the limit; only a few extremes occur outside the limit. Curve 5, which is negatively skewed shows a large number of response times outside the specification limit (L).

Listing common percentiles of the distribution is another way to represent data, one which is very convenient for response-time studies. A percentile is a value (on a scale of 100) that indicates the percent of a distribution that is equal to or below that value. This is illustrated in Figure B-3 (a graph called an ogive). If, for example, your application requires that no response time exceed three seconds, then the ogive shows that you can expect (within reason) that the application will perform acceptably 87.5% of the time (87.5% of the response times recorded in the sample are within the three-second limit).

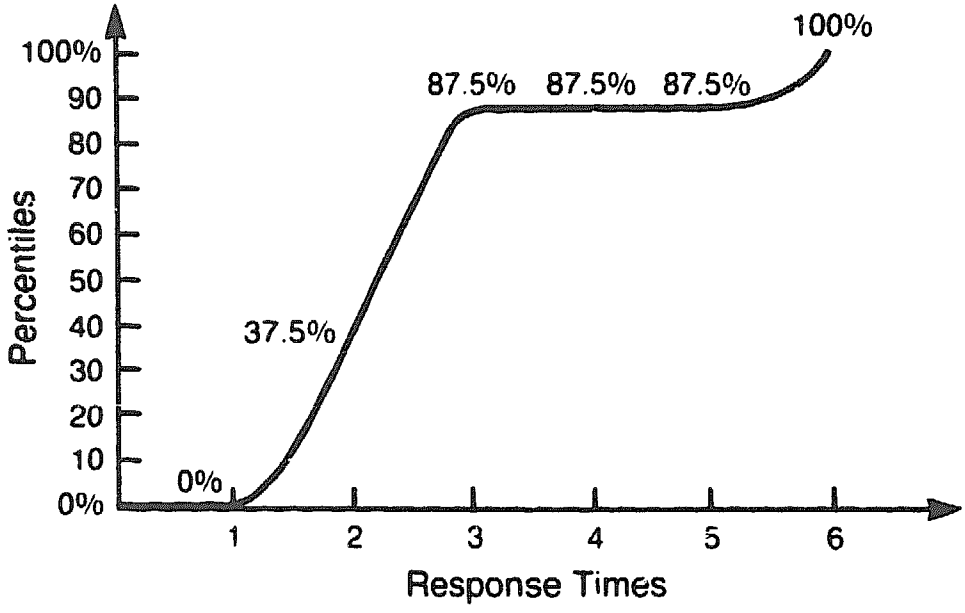


Figure B-3 Graph of Percentile Rankings (Ogive)

This discussion assumes that you are using a valid sample. For example, to get a good representation of the response times for an application, the sampling should cover a sufficient period of time, where you experience all possible workloads. For more on sampling techniques, see [1] and [2].

USING STATISTICS TO CHARACTERIZE DATA

Often it is practical and convenient to quickly summarize data using statistical terms, quantitative values that are typical of a distribution of data. You can use such values, called averages or **measures of central tendency**, to compare one distribution to another.

Measures of Central Tendency

Although the **mean** (arithmetic average) is the most useful statistic for averaging data, you cannot always rely on this value alone to typify a distribution of data. This is especially true when studying such random parameters as response times.

The following example reveals one of the shortcomings of relying on the mean. A politician, who is head of state, boasts that the mean salary of the population is $26,000. It is easy to conclude from this that most people are earning around $26,000. However, a closer analysis shows that half of the population earns less than $9,000, and that the mean was pulled up by several millionaires. Few people actually earned as much as $26,000, and most never came close.

The median, unlike the mean, is not influenced by extreme results (such as the few millionaires in the salary example). The **median** is the measure above and below which one-half of the measurements fall. (By definition, it is the 50th percentile in a distribution.) In the salary example, the median is $9,000.

The median better typifies skewed distributions. In Figure B-4, notice that the mean is pulled in the direction of the skew, while the median is not. You can use this fact to determine whether a distribution is skewed, and if so, in what direction. If the mean is higher than the median, the distribution is positively skewed; if lower, negatively skewed.

Another statistic, the mode, can help typify data. The **mode** is the measure of the most frequently occurring measurement.

All these measures — the mean, the median, and the mode — tell how data cluster around some central value. Sometimes the mean and one or two others of these measures coincide or nearly coincide, in which case the mean may be sufficient to represent the total population.

However, the measures of central tendency can (and often do) fail to show the whole picture. For example, the first three graphs in Figure B-2 reflect distributions that have the same mean and median values. The first two graphs also share the same mode. Yet, the graphs differ tremendously. Thus, these measures represent the data ambiguously.

Besides lending to ambiguity, the measures of central tendency fail to tell how data spreads out around the central measure. They do not tell about extremes. Because most

applications require that response times stay within a prescribed interval, known as a **confidence interval**, or below a certain limit, measures of central tendency are insufficient.

To test the calculated mean's sensitivity to extremes, you can use the **trimmed mean**. You find the trimmed mean by discarding an equal number of extreme values (that is, from both extremes of the graph), and by repeating the process until the trimmed mean is constant. Assume, for example, that the mean of a sample of response times is 2 seconds and the median is 1 second. After trimming the highest and lowest response time in the sample, the trimmed mean is 1 second, much smaller than the total mean. Then, trimming the second highest and lowest values, the trimmed mean is found again to be 1 second. This process reveals that there was an extreme (on the high end of the scale) that biased the mean.

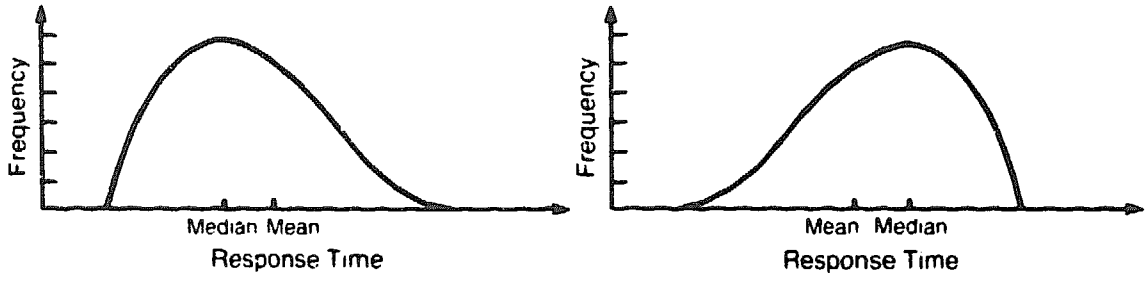


Figure B-4 Skewed Distributions

Measures of Dispersion: Ways to Represent Variability

If many responses diverge significantly from the mean, you need to be concerned about variability. Measures of dispersion and variability characterize how, and to what extent, a distribution diverges from a measure of central tendency.

The simplest measure of dispersion is the **range**, which states the distance between the smallest and largest recorded values. This shows the worst and best values, or the most atypical values. Often these uncover errors in software or hardware. Outside of this, however, the range has limited use.

One of the most valuable measures of dispersion is the **standard deviation** (the average deviation of the data from the mean). It is valuable because you can use it to compare

the variability of two or more distributions. Also, you can use it for more advanced statistical considerations, such as devising confidence intervals and for estimating error and precision (see [1]).

If a distribution is normal, then the mean and the standard deviation provide most of the information needed to understand a distribution of data. In the absence of any other information, the mean is the best basis for predicting results. In other words, the mean is the most probable value. The more compactly the data is clustered about the mean, as reflected by a relatively small standard deviation, the more assurance you have that a particular result will be at or near the mean.

You can use the standard deviation as an estimate of error when comparing two distributions with the same mean. For example, assume you are comparing the responsiveness of a network at two different traffic loads. The standard deviation for the distribution of response times measured at the first load is .5, while the standard deviation for the responses measured at the second load is 1.5. The response times with the smaller standard of deviation are more predictable and consistent.

In terms of probability and prediction, the response times for the workload with the smaller standard deviation are more likely to occur near or at the value of the mean. Response times for the workload with the larger standard deviation are more erratic. This could mean that the second workload is too large for the system to handle responses adequately. At least you can conclude that the first workload brings better, more consistent results.

However, the mean and the standard deviation do not always tell you everything you need to know. For example, you may know that a particular distribution has a mean of 50 and a standard deviation of 5, but those two statistics fail to tell you how much of your data will actually fall within a range of, say, one standard deviation below and above the mean (that is, within the range 45 to 55). If 95% of the data falls within that range, then the distribution can be described as having a mean of 50, plus or minus 5, with a 95% confidence level. If 99% of the data falls inside the range of two standard deviations below and above the mean, then the distribution would be described as having a mean of 50, plus or minus 10, with a 99% confidence level. By using all three statistics (mean, standard deviation, and confidence level), you can both characterize a distribution more accurately and also set up a set of stricter requirements or demands.

If a distribution is not normal, then the percentile ranking and ogive could best be used to reflect the distribution, rather than using the mean and standard deviation. More likely you will have to resort to more complex statistical tools that are beyond the scope of this discussion.

VERIFYING PERFORMANCE REQUIREMENTS

Performance requirements vary according to the needs and demands of the application. Take response times as an example.

1. For applications where timeliness is crucial, requirements could be specified something like this: "99% of all responses must be within two seconds, with absolutely none exceeding three seconds."

2. For other, less demanding applications: "95% of all responses must be within four seconds."

Performance requirements are rarely specified as a mean, for many of the reasons mentioned previously. Stating the requirements as done in the two examples here simplifies the process of verifying them. All you have to do is use the statistical methods described previously.

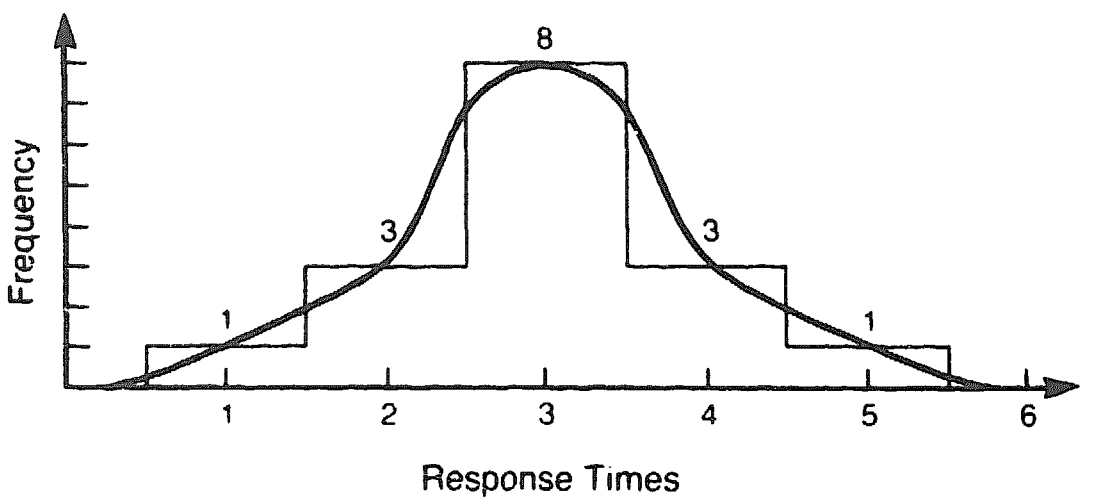


Figure B-5 Standard Normal Distribution

Requirement specifications are often more complex than the examples here, such as when you are considering several functions or applications, each with varying performance requirements. For example, assume you are evaluating the performance of a network for two applications, one being more vital to users than the other. Then the requirement that "95% of all responses must be within four seconds" is a problem if the 5% of responses exceeding four seconds are those of the more critical application. In such cases, you need two sets of requirements, or a weighted mean (as described in Chapter 3).

To verify that requirements will be fulfilled, you would first devise a benchmark, perform it, record an ample number of response times over a significant number and length of time intervals, and arrange the data in a histogram or curve. (To cut time and costs, you could test the network under the worst possible conditions, such as during the peak periods. If response times meet the requirements under *these* conditions, then you are sure they will meet them under less stressful conditions.)

If the distribution of recorded response times is normal, then you would find the mean and standard deviation. Consider the application requiring that 99% of all responses be within two seconds and none exceeding three seconds: if the longest response time recorded is less than three seconds and the mean is three standard deviations below two seconds, then the requirements are met. As for the application requiring that 95% of all response times be less than four seconds: if the mean is two standard deviations below four seconds, the requirements are met.

If the distribution of response times recorded in either application is skewed, you would resort to percentile rankings and the ogive to verify the performance requirements.

APPENDIX C
BIT RATE AND BAUD RATE

BIT RATE VERSUS BAUD RATE

A **bit** is the smallest unit of digital information. Because bits are also used to represent a quantity of data transferred per unit of time (bits per second), the term "bit rate" is often confused with "baud rate."

The **bit rate** defines the physical amount of data transferred during a time interval. Thus, 4800 bps means that 4800 bits of information are transferred in one second.

A **baud** is a unit of signaling rate, equal to the number of signal elements (pulses) per second. Depending on how the signals are sent, the value of the information rate in bits per second may differ from the value of the baud rate.

If each signal element represents only one bit of information, as in (A) of Figure C-1, then the baud rate and bit rate are equal. However, as shown in (B), more than one bit can be coded into a signal element, in which case the bit rate would be greater than the baud rate. Here, signal elements can have four different amplitudes, and each amplitude is assigned a unique 2-bit combination. Thus, two bits are coded into each signal element.

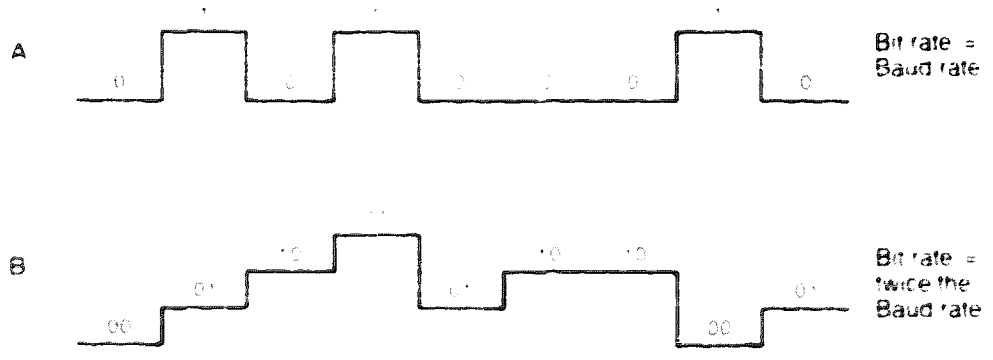


Figure C-1 Bit and Baud Rate

Notice that the coding technique serves to double the throughput of the channel. Other coding techniques can be used to further increase the throughput. Such coding techniques are used in modern communications equipment to increase the data throughput capacity of telephone lines, as explained below.

OPTIMIZING THE THROUGHPUT OF TRANSMISSION MEDIA

To improve the throughput of telephone lines, consider what determines the throughput of a communication channel. The real throughput of a channel is proportional to its bandwidth. By a law known as "Nyquist's relationship," the upper limit on the real throughput of a channel is less than twice the channel's bandwidth. When the limit is reached, the signaling rate is so high that the signals become too distorted for the receiving end to interpret them correctly. Thus, if the bandwidth of a voice-grade telephone line is 3,000 Hz, the maximum transfer rate (throughput) is only 6,000 signal elements per second or 6,000 baud.

If each signal element represents one bit of information, then the maximum throughput is 6,000 bps. When you use the coding techniques mentioned previously, you can increase the maximum throughput. Technological advances have brought the upper limit for voice-grade lines to 19,200 bps, and higher rates may be available in the future.

If you want higher bit-rate capabilities than those that are possible through these techniques, then you may have to move away from the conventional telephone technology to more expensive technologies. For example, certain communications systems can carry digital bit pulses instead of the conventional analog signal. With such signaling techniques, much higher bit rates can be achieved, such as 56,000 bps (56 Kbps), but *not* on voice-grade lines. Digital-signal transmission is less prone to errors than analog; thus, as you increase the proportion of the digital-transmission technology used in a network, the reliability of the network increases.

Since most data communications systems require retransmission of messages received in error, digital-signal transmission could significantly improve user throughput. Digital Data Service (DDS) networks of American Telephone & Telegraph (AT&T) provide a totally digitized service at 56 Kbps. The T-1 rate digital data links offer speeds of up to 1.544 Mbps. (For a discussion of how higher bit rates are attained, see [1] and [2].)

APPENDIX D
CPU UTILIZATION MEASUREMENTS

The following discrete measurements give a complete and accurate picture of CPU utilization. The measurements described here are those provided by the VAX/VMS Image Accounting Program.

PER-ACTIVATION DATA

Per-activation data measures the average resource consumption (time) for each activation of a given program. The resource can be the CPU itself or a CPU operation, such as an I/O. For file transfers, per-activation data tells how much CPU is consumed per transfer. This measure helps performance analysts model the behavior of application programs and construct synthetic workloads for testing purposes.

PERCENTAGE OF TOTAL RESOURCE CONSUMPTION

Percentage of total resource consumption measures the percentage of total resource (time) consumed by all activations of a given program. This measure is evaluated for each program. If programs are ranked according to their percentage of total resource consumption, analysts can determine which programs have most and least impact on the performance of the resource. With this information, analysts can better decide where to concentrate efforts for performance improvement. If program A, for example, consumes only 0.01% of the resource, while other programs consume much larger percentages, then analysts should not waste their time trying to optimize program A for better CPU efficiency. Improvements to this program would have relatively little impact on the CPU's performance. The analysts should concentrate their efforts on the programs having greater impact. Because these programs take more CPU time, and therefore, more user time, their improved efficiency can bring much greater user productivity.

PER-SECOND RESOURCE CONSUMPTION RATE

Per-second resource consumption rates measure the intensity of resource usage by various programs. The rate is found by dividing the resource consumption time by the elapsed time. For example, it tells you the amount of CPU time per second that a

particular operation consumes. A task with a high CPU rate is highly "compute-bound." A small number of such tasks could saturate the CPU.

Similarly, I/O operations per second and I/O bytes per second determine the disk utilization or network link utilization in systems with remote disk servers. (Disk servers are shared storage units dedicated to handling I/O storage operations for any number of host nodes.) Planners can use this information to determine the number and types of disks needed. Analysts and users can use it to find the programs that cause disks to bottleneck.

PER CPU-SECOND RESOURCE UTILIZATION

Per CPU-second resource utilization measures the ratio of resource demand to CPU demand, such as the number of I/Os per CPU-second (active CPU time). Per CPU-second resource utilization takes into account the demand, or arrival rate, on the resource. Thus, it is not as easily biased by extreme variations of demands that occur over time. (The per-second resource consumption rate is measured over an elapsed time. The demands within that elapsed time can vary significantly.)

Programs with higher per CPU-second resource utilization tend to impose higher demand on a resource. For example, programs with a high number of disk I/Os per CPU-second are usually "disk-bound." With too many of such programs, the disk, and not the CPU, is likely to become a bottleneck. Analysts use this information to determine the maximum number of simultaneous users of such programs on a system.

Such information is especially important for analyzing and planning for usage of a shared system. How many users can be handled simultaneously without saturation? What matters here is the number of *active concurrent user sessions*; that is, how many users are logged on and *actively* using CPU resources at the same time? The number of active users is more significant than the number of users logged on the system: it is the active users who consume the most CPU time. Many users can be logged on simultaneously, but fewer can be active simultaneously without saturating the CPU.

In SNA Gateway communications, a similar concern is how many established, active sessions can be supported simultaneously. The key is the number of *active sessions* — those which are simultaneously engaged in network I/O. There may be several established but inactive sessions, which have less impact on the CPU.

REFERENCES

Chapter 1

[1] Digital Equipment Corporation. *Networking. The Competitive Edge*. Order No. EB-27241.

[2] Hirsch, Phil. "Cost Overshadows Benefits in Weighing Value of Nets." *Computerworld*. November 5, 1984.

[3] "Integration Seen Hiking Productivity." *Management Information Systems Week*. Wednesday, September 12, 1984.

[4] Tucker, Jonathon B. "GM: Shifting to Automatic." *High Technology*. May 1985.

Chapter 2

[1] Davies, D.W., and D.L.A. Barber. *Communications Networks for Computers*. London: John Wiley Sons, 1973.

[2] Davies, D.W., D.L.A. Barber, W.L. Price, and C.M. Solomonides. *Computer Networks and Their Protocols*. New York: John Wiley Sons, 1979.

[3] Jain, Raj, and William R. Hawe. "Performance Analysis and Modeling of Digital's Network Architecture." *Digital Technical Journal, No. 3*. September 1986. Order No. EY-6715E-DP.

[4] Sherman, Kenneth. *Data Communications: A User's Guide*. Virginia: Reston Publishing Co., 1981.

[5] Tanenbaum, Andrew S. *Computer Networks*. New Jersey: Prentice-Hall, 1981.

Chapter 3

[1] Ferrari, Domenico, Giuseppe Serazzi, and Alessandro Zeigner. *Measurement and Tuning of Computer Systems*. New Jersey: Prentice-Hall, 1983.

[2] Fitzpatrick, Michael. "A Cure for Trial-and-Error Network Management." *Telecommunications*. January 1985.

[3] Haber, Audrey, and Richard P. Runyon. *General Statistics*. Massachusetts: Addison-Wesley, 1971.

[4] Martin, James. *Systems Analysis for Data Transmission*. New Jersey: Prentice-Hall, 1972.

[5] Martin, M.P., and J.E. Trumbly. "Measuring Performance of Automated Systems." *Journal of Systems Management*. February 1986.

[6] Miles, Martin J., and Neal B. Seitz. "Testing Network Performance: A Statistical Analysis." *Data Communications*. June 1985.

[7] Moore, Connie, and Rob McAuliffe. "An Effective Way to Design and Implement a Local Network." *Data Communications*. October 1985.

[8] Stuck, B.W., and E. Arthurs. *A Computer and Network Communications Performance Analysis Primer*. New Jersey: Prentice-Hall, 1985.

[9] Svoboda, Liba. *Computer Performance Measurement and Evaluation Methods: Analysis and Applications*. Massuchusetts Institute of Technology. New York: Elsevier, 1977.

Chapter 4

[1] Black, Uyless D. *Data Communications, Networks, and Distributed Processing*. Virginia: Reston Publishing Co., 1983.

[2] Digital Equipment Corporation. *Distributed Systems Handbook*. 1978. Order No. EB P1195.

[3] Gorney, Leonard. *Queuing Theory: A Problem Solving Approach*. Petrocelli Books, 1981.

[4] Kleinrock, Leonard. *Queuing Systems: Volume II, Computer Applications*. New York: Wiley-Interscience, 1976.

[5] Martin, James. *Design of Real-Time Computer Systems*. New Jersey: Prentice-Hall, 1967.

[6] Miller, Robert B. "Response Time in Man-Computer Conversational Transactions." *American Federation of Information Processing Societies Conference Proceedings*. Vol. 33, pt. 1: 267-277. Fall 1968.

[7] Rubinstein, Richard, and Harry Hersh. *The Human Factor — Designing Computer Systems for People*. Digital Press, 1984. Order No. EY-00013-DP.

[8] Sherman, Kenneth. *Data Communications: A User's Guide*. Virgina: Reston Publishing Co., 1981.

[9] Stuck, B.W., and E. Arthurs. *A Computer and Network Communications Performance Analysis Primer*. New Jersey: Prentice-Hall, 1985.

[10] Tanenbaum, Andrew S. *Computer Networks*. New Jersey: Prentice-Hall, 1981.

[11] Wortendyke, D.R., N.B. Seitz, K.P. Spies, E.L. Crow, and D.S. Grubb. *User-Oriented Performance Measurements on the ARPANET: The Testing of a Proposed Federal Standard*. U.S. Department of Commerce, 1982.

Chapter 5

[1] Black, Uyless D. *Data Communications, Networks, and Distributed Processing*. Virginia: Reston Publishing Co., 1983.

[2] Gee, K.C.E. *Introduction to Local Area Computer Networks*. A Wiley-Interscience Publication. New York: John Wiley Sons, 1983.

[3] Hancock, Bill. "VMS Remote System and Network Management: Setting Up a Computer Network Means Simplifying Management." DEC PROFESSIONAL, Digital Equipment Corporation, September 1985.

[4] Harper, William L., and Robert C. Pollard. "Understanding System Throughput and Data Flow." Datapro Research Corporation (April 1983). (From *Data Communications Desk Book: A Systems Analysis Approach*. New Jersey: Prentice-Hall, 1982.)

[5] Jain, Raj. *A Dynamic Window Congestion Control Scheme for Digital Network Architecture*, Appendix C. Digital Equipment Corporation. September 1983. Technical Report DEC-TR-275.

[6] McNamara, John E. *Local Area Networks: An Introduction to the Technology*. Digital Press, 1985. Order No. EY-00051-DP.

[7] Sherman, Kenneth. *Data Communications: A User's Guide*. Virgina: Reston Publishing Co., 1981.

[8] Stuck, B.W., and E. Arthurs. *A Computer and Network Communications Performance Analysis Primer*. New Jersey: Prentice-Hall, 1985.

[9] Tanenbaum, Andrew S. *Computer Networks*. New Jersey: Prentice-Hall, 1981.

[10] Wortendyke, D.R., N.B. Seitz, K.P. Spies, E.I.. Crow, and D.S. Grubb. *User-Oriented Performance Measurements on the ARPANET: The Testing of a Proposed Federal Standard*. U.S. Department of Commerce, 1982.

Chapter 6

[1] Black, Uyless D. *Data Communications, Networks, and Distributed Processing*. Virginia: Reston Publishing Co., 1983.

[2] Giessler, A., J. Hanle, A. Konig, and E. Pade. "Free Buffer Allocation — An Investigation by Simulation." *Computer Networks*. Vol. 1, no. 3: 191-204. July 1978.

[3] Ilyas, M., and H.T. Mouftah. "Performance Evaluation of Computer Communications Networks." *IEEE Communications Magazine*. Vol. 23, no.4. April 1985.

[4] Jain, R. "Using Simulation to Design a Computer Network Congestion Control Protocol." *Proceedings of the Sixteenth Annual Modeling and Simulation Conference*, pp. 987-993. Digital Equipment Corporation. April 1985. Technical Report DEC-TR-358.

[5] Kleinrock, Leonard. "Power and Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications." *Conference Record, International Conference on Communications*. Vol. 43, no. 1: 1-10. June 1979.

Chapter 7

[1] Cypser, R. J. "How to Improve Line Utilization in an Existing System." Datapro Research Corporation (January 1982). (From *Communications Architecture for Distributed Systems*. Massachusetts: Addison-Wesley, 1978.)

[2] Digital Equipment Corporation. *Routing and Networking Overview*. Order No. AA-HS15A-TK.

[3] Fitzpatrick, Michael. "A Cure for Trial-and-Error Network Management." *Telecommunications*. January 1985.

[4] Gorney, Leonard. *Queuing Theory: A Problem Solving Approach*. Petrocelli Books, 1981.

[5] Jain, R. K., and R. Turner. "Workload Characterization Using Image Accounting." *Proceedings of the Computer Performance Evaluation Users Group 18th Meeting (CPEUG 82)*, pp. 111–120. 1982.

[6] Johnson, Joseph T. "Universal Flow and Capacity Index Gives Picture of Network Efficiency." *Data Communications*. February 1985.

[7] Kleinrock, Leonard. *Queuing Systems: Volume II, Computer Applications*. New York: Wiley–Interscience, 1976.

[8] Martin, James. *Systems Analysis for Data Transmission*. New Jersey: Prentice–Hall, 1972.

[9] Morin, Richard. "The Human Factor: Random Thoughts on UNIX System Performance." *UNIX Review*. March 1985.

Appendix B

[1] Haber, Audrey, and Richard P. Runyon. *General Statistics*. Massachusetts: Addison–Wesley, 1971.

[2] Miles, Martin J., and Neal B. Seitz. "Testing Network Performance: A Statistical Analysis." *Data Communications*. June 1985.

Appendix C

[1] Black, Uyless D. *Data Communications, Networks, and Distributed Processing*. Virginia: Reston Publishing Co., 1983.

[2] McNamara, John E. *Technical Aspects of Data Communication*. Digital Press, 1982.

GLOSSARY

Access bandwidth: A measure of disk capacity, expressed in time as megabytes per millisecond. This measures the speed at which you can access data on the disk.

Access time: The time waiting for access to a network. The data waits in a queue until the processor has time to transmit the data or until the network is ready to accept the data. Also called **admission delay.** See also **Connection time.**

Active concurrent user sessions: The number of users logged on and *actively* using CPU resources at the same time.

Adaptive routing (alternate routing): A routing system that uses an alternate communications path if the normal one is not available. There may be one or more possible alternative paths. See **Routing.**

Aggregate throughput: See **Throughput, network.**

Analog data signal: Signals that are smooth, electronic waves, a continuum of voltages, rather than discrete voltage changes. The signal can assume any of an infinite number of voltage or current values. In contrast, a digital signal can only assume a finite number of values. Thus, analog signals are more subject to error, since digital signals have more definite possibilities. Contrast with **Digital data signal.**

Application level: The highest protocol level in the network architecture, which actually performs services for users.

Application program: A program that performs a function specific to the needs of a particular user or class of users. Can be a program that is not part of the basic operating system of the CPU where it is located.

Applied load: See **Load, applied.**

Asynchronous transmission: Transmissions in which the time periods between character transmissions differ. Each character is surrounded by a start bit and a stop bit to help the receiving device recognize the beginning and end of each character. (Also called start-stop transmission.) Most commonly used over terminal lines and lines connecting other inexpensive devices, such as personal computers. Contrast with **Synchronous transmission.**

Availability: The proportion of time a specific piece of equipment, system, or network is usable, compared to the total time it is expected to be.

Bandwidth: The bandwidth of a transmission medium is the range between the highest and lowest frequencies at which signals can be passed through it without the signal being distorted beyond recognition by the receiver. The bandwidth reflects the medium's information-carrying capability. Usually expressed in terms of its signaling rate in Hertz or bits per second.

Baseband signaling: Transmission of a signal at its original frequency; that is, a signal not changed by modulation, sent over a single channel.

Baseband LAN: A LAN that transmits all data at the same frequency over a single channel (and still provides high-speed transfers). It is analogous to a very high-speed, single-lane road. Contrast with a **broadband LAN**, which is analogous to a multilane highway.

Batch processing: Processing in which transactions are collected and prepared as a unit for input to a computer or for transmission over a network. The processing of data accumulated over a period of time or designated for processing at a specified time. Contrast with **Interactive processing**, and **Real time**.

Baud: A unit of signaling rate, equal to the number of signal elements (pulses) per second. Depending on how the signals are sent, the value of the information rate in bits per second may differ from the value of the baud rate. If each signal element represents only one bit of information, then the baud rate and bit rate are equal. However, more than one bit can be coded into a signal element, in which case the bit rate would be greater than the baud rate.

Benchmark: A reference workload that can be used as the basis for performance comparisons and evaluation. It consists of a group of jobs or tasks that attempt to represent a typical workload for the component or network under evaluation. Benchmarks are often designed to exercise a particular feature. As a verb, benchmark refers to the effort to establish ratings of a component, system, or network using a standard input condition.

Bit: The smallest quantity of data processing information. A bit can assume the values of 0 or 1, only.

Bit transfer rate: The number of bits transferred in a unit of time, usually expressed in bits per second (bps). Contrast with **Baud**.

Block: Any contiguous unit of user information that is grouped together for transmission, such as the user data within a packet, excluding the protocol overhead.

Block transfer time: A measure of the network delay. The block transfer time consists of the time required to transfer data from the source to the destination and back, excluding the time taken by the destination node to generate a response.

Bottleneck: A point in the network where traffic is delayed or blocked. Usually a saturated device becomes a bottleneck. Bottlenecks are the limiting factors in network performance.

Bps: Bits per second.

Bridge: A device that expands the extent of a LAN by connecting it to another LAN or physical link. For example, Digital's LAN Bridge 100 manages inter-LAN traffic and selectively forwards packets to keep local traffic local. Only data destined for different LANs passes through the bridge and continues on to the appropriate remote destination. Thus, the bridge improves performance by reducing traffic between LANs.

Broadband: A communication channel having a great bandwidth, and therefore capable of multiple, parallel high-speed transmissions.

Broadband LAN: A LAN that transmits signals under different frequencies. The LAN transmission medium is subdivided into a number of independent frequency channels. These subchannels transmit different forms of information — voice, data, and video — simultaneously.

Broadcast: In networks, the capability of sending a single message simultaneously to many nodes, just as television programs are sent to a multiple of television sets.

Buffer: A device or an area of memory used for temporary storage to compensate for a difference in rate of data flow or in time of occurrence of events, when transmitting data from one device to another.

Buffering level: In network communications, this refers to the number of buffers provided at a time by the network software to handle data. Single-buffering tends to be less efficient than multibuffering, but uses less memory on the local system. Multibuffering provides better performance if sufficient memory is available. With multibuffering, a network can send or process several buffers of data in quick succession.

Bus: (1) A LAN topology where all nodes connect to a single transmission medium so that all nodes are equal and all nodes hear all transmissions on the medium. Bus topologies are reliable because failure of any node does not affect the ability of any other nodes to transmit and receive. (2) Digital-specific: A flat, flexible cable that consists of many transmission lines or wires, and that interconnects computer system components to provide communication paths for addresses, data, and control information.

Byte: A unit of data consisting of eight bits.

Caching: An efficient method of using memory, as used on almost all VAX/VMS systems. The most frequently used (most recently used) information is kept in a special area (called a cache) between main memory and the processor, where it can be accessed more quickly. Used to minimize the physical transfer of data between mass storage

devices (such as disks) or main (slower) memory and the CPU. This quickens access time to data and processing time.

Channel: The data path between two or more stations, including the communications control capability of the associated stations.

Character-interrupt device: A communications device that interrupts CPU processing for each character received from the communications line and for each character ready for transmission. This consumes considerable CPU time and limits character throughput. Contrast with **DMA device.**

Circuit: A logical communications path providing a communications connection between adjacent nodes. In DECnet, there is one circuit for each point-to-point line, one for each tributary station on a multipoint line, and one for each node on an Ethernet LAN.

Circuit cost: In DECnet, an arbitrary, positive integer assigned to a circuit by the network manager. Messages traveling over the network are routed over the path with the smallest total cost.

Circuit switching: A type of switching used by telephone companies and networks for establishing a physical path from your telephone (or modem) to the receiver's telephone. When you dial, one or more switching centers set up a temporary path which is reserved for the duration of the call. The circuits chosen for a path may vary from call to call, depending on their availability. Contrast with **Packet switching.**

Coaxial cable: An electrical cable in which a piece of wire is surrounded by insulation and then surrounded by a tubular piece of metal whose axis of curvature coincides with the center of the piece of wire, hence the term of "coaxial." Offers high transmission rates. Used by Ethernet.

Collision: The event when two nodes in a network try to transmit at the same time on the same channel, causing the transmitted data to be unusable.

Communications buffers: Buffers allocated at each node for temporary storage of communications data. (In VMS systems, they are also called executor buffers.)

Communications servers: Special-purpose, stand-alone systems dedicated to managing communications activities for other computer systems. See also **Front-end processor.**

Compute bound: Describes a task that requires a relatively large amount of CPU time to process. A small number of such tasks could saturate the CPU.

Concentrator: An intelligent multiplexer which has more storage capacity than most other types of multiplexers and performs more complex functions. A concentrator reduces the number of physical communications links between two points by logically multiplexing the links. Terminal servers can be considered as concentrators.

Concurrency: Simultaneous use of a resource, such as concurrent terminal sessions.

Conditioning: A technique that improves the capability of a line for higher data rates. Used on leased, private lines.

Confidence level: A specified boundary for performance results. Most applications specify a confidence level for certain performance parameters (for example, a confidence level for response times might be 1 second, plus or minus .5 second).

Congestion: A condition when the demands on the network exceed the network's capacity to handle them within a particular time. The condition that arises when arriving traffic exceeds the capability of servers handling the traffic.

Connection time: On certain networks, this is the time required to establish a link between the source and destination nodes. On dialup lines, it includes the time to dial and establish the connection.

Contention control: A scheme of access control used by many networks. Control is distributed among the nodes of the network. Any node that wishes to transmit may do so, accessing the network on a first-come, first-served basis. It does not have to wait its turn. However, it is possible that two nodes may be in contention, or start transmitting at the same time, in which case a collision occurs, and each node must back off and transmit again after waiting a random period of time. An improvement on this scheme is to have the device sense whether the channel is busy before transmitting. Another improvement is to have the device sense while transmitting, thereby reducing the time needed to detect a collision. The CSMA/CD method uses all these techniques. See **CSMA/CD**.

Control data: Control data is the information used by the network or communicating devices to transfer and process data. Distinguished from user data. For example, control data includes the destination and source address of a packet of user data. Error-control information is control data. Control data is considered part of the protocol overhead incurred in communications.

Control token: See **Token passing.**

Controller: See **Line controller.**

Cost: See **Circuit cost.**

CPU cycle: One tick of the CPU's internal clock. CPU utilization is often expressed in terms of the number of CPU cycles consumed by a process.

Criteria: The measures selected for evaluating a network. In discussions of performance criteria, this book includes only those criteria that are easily quantifiable.

Critical point: The point on the graph of a function where the increase or decrease takes a sharp turn.

Crosstalk: Noise passed between elements of a communications cable or device.

CSMA/CD: Carrier Sense Multiple Access with Collision Detection, a method of having multiple stations access a transmission medium (multiple access) by listening until no signals are detected (carrier sense), then transmitting and checking to see if more than one signal is present (collision detection).

Data compression: A technique used for cutting transmission costs or time, or for saving buffer space. It removes unnecessary gaps, redundancies, and empty fields from data streams.

Data integrity: Preservation of data for its intended purpose. A network with high data integrity is one that ensures that data is transferred safe from line noise and other conditions that can cause data to be received incorrectly.

Data link: A logical connection between two stations on the same circuit on which data integrity is maintained.

Data-signaling rate: See **Transfer rate, maximum.**

Deadlock: The point at which real throughput of a network becomes zero, such as when the amount of traffic saturates one or more components, causing the traffic flow to stop.

DECnet: Digital networking software that runs on nodes in both local and wide area networks. An implementation of DNA (Digital Network Architecture). See **DNA.**

Dedicated line: Synonymous with a leased or private line, it is reserved or committed to a specific user or purpose.

Dedicated network: One that consists of dedicated lines and computers. Each line or computer is dedicated to a specific application or user.

Destination user: The user to whom the network is delivering information.

Digital data signal: A signal that has discrete changes (such as 0-bit, 1-bit, or "on" and "off"). Transmitted by computers. Contrast with **Analog data signal.**

Disconnect time: The time required to disengage a connection; that is, the delay from when the disengagement is requested to when it is completed.

Distributed processing: A technology that enables the distribution of computing power and storage facilities to user work areas such as offices, labs, or desks on factory floors. Distributed processing grew out of the realization that users wasted valuable time by having to wait for answers to their questions from isolated and remotely located mainframe computers. Digital is a pioneer in distributed processing.

DMA device (direct memory access): A device that permits I/O transfers directly into and out of CPU memory. The device does not interrupt the CPU for each character, as do **character-interrupt devices.** Instead, it transfers I/O data in blocks. It requires CPU attention only when the entire message has been transferred. By this method, character throughput is greatly increased.

DNA (Digital Network Architecture): Digital's layered data communications protocols. A division of activities into layers, required to perform network communications. DNA is compatible with the internationally accepted OSI model.

Down time: The period of time during which a system or network is not operable because of component failure.

Duplex transmission: See **Full duplex.**

Dynamic bandwidth switching: For data links in a network, bandwidth is allocated automatically to links needing more bandwidth because of sudden rises in demands. Permits flexibility in assignment of bandwidths.

Echo: On a terminal, the terminal displays each character instantaneously after it is typed. A remote device can return each character to the screen as it receives it. Provides positive feedback to the terminal user.

Effective throughput: See **Throughput, net.**

End node: A nonrouting node. It can receive packets addressed to it and send packets to other nodes, but it cannot route packets to other nodes.

End user: (1) A human operator of a data terminal connected to a node in a network, who generates or uses information communicated over the network. (2) A computer application program accessing a network to generate or use communicated information.

Error detecting code: Code in which each data signal conforms to specific rules of construction so that departures from it in the received signals can be automatically detected. Data detected as received in error can be dropped, with or without informing the user(s), or it can be retransmitted.

Ethernet: A local area network that employs coaxial cable as a passive communications medium in a CSMA/CD system to interconnect different types of computers, server products, and office equipment at a local site. No switching logic or central computer is needed to establish or control communications.

Fiber optics: A transmission technology that transmits signals in the form of light through glass fibers (high transfer rates and immunity to electromagnetic interference).

Flow control: Hardware or software mechanisms employed in data communications to turn off transmission when the receiving station is unable to store the data it is receiving.

In DECnet, the NSP (Network Services Protocol) does flow control by coordinating the flow of data on a logical link in both directions, from transmit buffers to receive buffers. It ensures that data is not lost, prevents buffer deadlock, and minimizes communications overhead.

Forward-error correction: A mechanism for handling errors in data communications. It detects and corrects the errors, reconstructing the lost or damaged data. Thus, it eliminates the need for retransmissions. Forward-error correction requires that extra bits be sent with user data. See **Error detecting code.**

Frequency: The number of cycles per second at which an analog signal occurs, usually expressed in terms of Hertz (Hz). One Hz is one cycle per second.

Frequency division multiplexing (FDM): A form of multiplexing that divides the communications line into separate frequency channels. Each user transmits at a different frequency. Broadband is a form of FDM. See also **Statistical time division multiplexing**, and **Time division multiplexing.**

Front-end processor: A supplementary CPU that relieves a main CPU of one or more functions. For example, when the demands on a node in a network are great, another CPU can be configured to handle network processing exclusively. This is called **front-end communications processing**; the data communications control function is removed from the main CPU (the **host**) and delegated to another CPU (the **front end**).

Full duplex: Simultaneous two-way independent transmission in both directions. Provides more efficient use of communications lines than **half duplex.**

Gateway servers: Communications servers that provide access from one type of network to another (networks having different access protocols). For example, Digital's SNA Gateway provides access to IBM SNA systems, while the X.25 Gateway provides access to X.25 PSDNs.

Half duplex: Two-way transmission, but one way at a time. Contrast with **Full duplex.**

Hertz (Hz): A unit of frequency equal to one cycle per second.

Host: The primary or controlling computer in a multiple computer network. A node that provides services for another node. In a LAN, any primary processing nodes with multiple computer facilities. Distinguished from communications servers, which are dedicated units that offload the hosts of specific functions.

Idle time: Time that a computer is not being used while available for use.

Interactive processing: A mode of computer operation in which the commands and data that control the actions of the computer are entered by a person at a terminal rather than by a programmed script. Contrast with **Batch processing.**

Internal delays: The delays imposed by the components and internal functions of a network. For example, the communications lines between nodes impose delays that depend on the speed of each line. See **Network delay.**

Interrupt: A signal given to a computer that, when acknowledged, causes the computer to stop what it is doing (storing the details thereof) and to turn its attention to the device that sent the signal. After the device is serviced, the computer resumes where it left off.

Kbps: Kilobits per second. (One kilobit is 1000 bits.)

LAN (local area network): A privately owned data communications system that offers high-speed communications channels optimized for connecting information processing equipment. Geographical area is usually limited to a building or group of buildings.

Layer: In networks, layers pertain to the software protocol levels that make up the architecture. Each layer performs certain functions for the layers above it.

Leased line: See **Dedicated line.**

Line: A physical communications path.

Line controller: A hardware device (at each node) that manages communications over each line. (It handles the physical-link and data-link layers.)

Line speed: Maximum rate at which data can be reliably transmitted over a line. Rate varies with the modem or hardware device that does the transmitting. Generally, line speed is expressed in bits per second (bps). As a signaling rate of transmission facilities, it is often expressed in baud. Contrast with **Transfer rate, data.**

Line speed, effective: See **Transfer rate, maximum.**

Link: Any specified relationship between two nodes in a network. A communications path between two nodes.

Load: The total volume of work demanded of a component or network. In networks, the load is synonymous with traffic intensity.

Load, applied: The total data actually given to the network for transmission, including protocol overhead. Contrast with **Load, offered.**

Load balancing: Load balancing is the ability of a system to bring a user job or request to the least loaded resource (of a group of resources providing similar services).

Load, offered: The load offered to a component by the user application. Consists of the data that crosses the data link interface of a network to be prepared for transmission onto the network. It does not include protocol overhead.

Logical link: A temporary connection between source and destination nodes (or between two processes on the same node).

Mbps: Megabits per second. (One megabit is one million bits.)

Mean (arithmetic): The average of a group of numerical values, calculated by finding their sum and dividing by the number of values. See **Measure of central tendency.**

Median: The measure of central tendency that marks the middle point of a distribution, above and below which one-half of the distribution falls. (By definition, it is the 50th percentile in a distribution.)

Measure of central tendency: A statistic used to typify a distribution of data; it characterizes how the data clusters around some central point or value. Such measures include the arithmetic **mean, median,** and **mode.** Contrast with **Measure of dispersion.**

Measure of dispersion: A statistic used to typify the fluctuation or variability of a distribution of data (around some central point).

Message: A message block, or a series of message blocks, that constitute a logical grouping of information. Each message block is delimited by communications control characters.

Metric: A vector that has two dimensions (parameters) that can be displayed, for example, on an "x" and "y" axis in a graph, or in rows and columns in a table.

Microsecond: One-millionth of a second. Abbreviated us.

Millisecond: One-thousandth of a second. Abbreviated ms.

Mode: The measure of central tendency that corresponds to the most frequently occurring value or measurement in a distribution.

Modem: "Data sets" at each end of a communications line necessary to convert digital signals transmitted by a computer into analog signals for transmission over a voice-grade line. A modem at the other end converts the analog signal back into digital form for the receiving computer. MOdulator/DEModulator.

Monitor: A software or hardware tool that records events or activities of a system or network, and/or analyzes the data.

Multibuffering: Providing more than one buffer at a time to handle data. A network can send or process several buffers of data in quick succession. In single-buffering, it handles only one buffer at a time, so subsequent operations requiring buffers must wait until the single buffer is available. Multibuffering is a form of **pipelining.**

Multidrop line: See **Multipoint line.**

Multiplex, multiplexing: To use a common physical channel to transmit two or more data streams, simultaneously or nearly simultaneously. See **Frequency division multiplexing, Time division multiplexing,** and **Statistical time division multiplexing.**

Multiplexer: A communications device that allows several users to share a single circuit. A multiplexer funnels the data streams of several users into a single stream. Another multiplexer at the other end of the link splits the stream back into the original streams, forwarding them to their intended destinations (such as user terminals).

Multipoint line: A single line shared by several nodes. (An Ethernet is a form of a multipoint line, but is more correctly referred to as a multi-access medium.) Most multipoint configurations include a central node that controls access to the line shared by the other nodes (Ethernet LANs do not need a central node).

Multiprocessing: Multiprocessing consists of linking two or more CPUs together to share resources efficiently. Resources can be CPU cycles, specialized peripherals, or disk memory, for example. An application can be split into components that interact by sharing databases or exchanging messages. Also, these components can be designed to exchange messages or share data in the network environment. Digital's VAXclusters are a form of multiprocessing.

Network: In the context of this book, a group of computers interconnected to share resources and information.

Network delay: The time it takes to get a unit of data from the source of a transmission to the destination. Usually associated with the purely network-induced delays, which include **transmission time** and **propagation time**, and not application processing delays at source and destination nodes See also **Internal delays**, and **Queuing delays.**

Network management: The administrative services needed to manage a network, such as configuring and tuning the network software, monitoring network performance, maintaining network operation, and diagnosing and troubleshooting network problems. In the DNA architecture, the layer containing functions that enable operator control over, and observation of, network parameters and variables.

Network performance: The way a network performs, measured against the expectations or requirements of users, customers, designers, and/or implementors, or as claimed by sales and marketing personnel. The criteria for network performance include such parameters as throughput, response time, and resource utilization.

Network utilization: The ratio of total data flow to the theoretical maximum data flow.

Node: Any intelligent device that communicates with other devices in the network. A point in the network where service is provided or used, or communications channels are interconnected.

Noise: Undesirable disturbances in a communications system or component. Noise can generate errors in transmission.

Normal distribution: A distribution of data that, in graphical form, is symmetrical and bell-shaped. (Note that "normal" is not equivalent to "usual.") Contrast with **Skewed distribution.**

Offered load: See **Load, offered**

OSI (Open System Interconnection): The internationally accepted framework of standards for intersystem communication. Developed by the International Standards Organization (ISO). OSI is a seven-layer model that covers all aspects of information exchange between systems. Digital's **DNA** is compatible with OSI. See **DNA.**

Ogive: A graph or curve that constantly increases on the vertical scale as the value on the horizontal scale increases.

Overhead data: All information other than user information. Overhead data can include: (1) header information on a user message, giving such information as the source and destination address, number of bits included; (2) control information for data exchanges between two parties, such as hand-shaking information to initiate and synchronize an exchange, acknowledgment of data received; (3) information sent by the network to users for reporting system status or controlling user operations.

Packet: A group of bits, including user data and control data, that are transferred as a unit through a network. The data units transferred in DECnet are called packets. In packet-switched networks, a packet is the basic unit of transmission. Each packet sent by a user can take different routes to the destination. (DECnet does not employ packet switching, although it can provide access to packet-switching networks.)

Packet switching: A form of switching in which user data and the associated protocol information is formed into packets. The communications channel is occupied only for the duration of the message, rather than for the entire session. Thus, the channel is freed quickly for use by packets being sent by other users. Packets of the same user message can take separate routes to their destination, rather than all taking the same route as in circuit switching. When all the packets of a message reach their destination, they are reassembled in their proper sequence and delivered to the user there. Contrast with **Circuit switching,** and **Store-and-forward.**

Page: A group of bytes stored in memory or on disks. Memory can hold a limited number of pages at a time. When the demand for pages exceeds those available in memory, certain pages are swapped between memory and disks. When used as a verb (Digital-specific), to move parts of a user's program to or from memory selectively.

Paging: On VAX/VMS systems, the act of bringing pages of an executing process into physical memory when referenced. When a process executes, all of its pages are said to

reside in virtual memory. Only the actively used pages, however, need to reside in physical memory.

Parameters: Software-related variables that network managers, system managers, and applications programmers can manipulate to optimize performance. In networks, these include: the sizes of packets and buffers, the length of certain timers or timeout values, the sizes of certain quotas, and more.

Parity check: An error detection technique. Non-information bits are added to each character transmitted, making the number of 1's in each grouping of bits either always odd (for odd parity) or even (for even parity). The receiver checks that the parity is the same as it was when transmitted. If it differs, then at least one bit is in error (a 1 changed to a 0, or vice versa).

Path: A possible route for a unit of data from a source node to a destination node. The path can comprise a sequence of connected nodes between the source and destination.

Path cost: The sum of the circuit costs along a path between two nodes.

Path splitting: In DECnet Phase IV, the ability to split the transmission load for a single destination, using several paths of equal total path cost.

PBX: Private branch exchange. Many companies can save money by using their internal phone systems as LANs. However, PBXs provide slower data transmission and low capacity. They are also more susceptible to noise problems (static) than other types of LANs because they consist of twisted-pair cables.

Per-activation data: See Appendix D.

Per-second resource consumption rate: See Appendix D.

Percentage of total resource consumption: See Appendix D.

Percentile: A value on a scale of 100 that indicates the percent of a distribution that is equal to or below it. Thus, the 90th percentile is the response time below which 90% of the recorded response times fall.

Per CPU-second resource utilization: See Appendix D.

Performance: See **Network performance.**

Performance criterion: A characteristic or standard by which performance is judged. This book defines these criteria: response time, throughput, and resource utilization.

Performance analysis: Determines why the system or network performs the way it does. It is concerned with the internal measures, the internal "mechanics" or "processes" and how each affects the performance individually and in combination with others. Performance analysis is done for the purpose of improving performance.

Performance evaluation: Determines how well the network does its job. It is concerned with the external measures. Performance evaluation is driven by the needs of the users. The criteria must be those that they care most about. The expected values for these measures are used to judge the measured results.

Performance parameter: A statistical quantity whose numerical values characterize a particular aspect of network performance.

Piggybacking: A mechanism by which the data link layer of a network holds an acknowledgment (of data received) until a packet of user data headed for the same destination is transmitted. The acknowledgment is then embedded as part of the protocol header on the packet and "hitches" a free ride. More efficient than sending each protocol message separately, since it significantly reduces overhead.

Pipelining: A technique in which several messages or tasks are handled simultaneously through several stages of a process. This speeds up processing time — much more efficient than processing one message or task at a time through several stages, forcing subsequent messages or tasks to have to wait until the preceding are processed. At the data link layer of a network, pipelining refers to the mechanism by which the transmitting node sends several messages without waiting for individual acknowledgment of each successive message.

Pipeline quota: The maximum number of messages or packets that a transmitting node can send without waiting for individual acknowledgments for each successive message.

Polling: A scheme for control of access to a data channel or resource. For example, a central computer or some other device controls transmissions between nodes. The nodes cannot access the channel on their own. The central node polls each node consecutively, asking if it has anything to transmit. When a node says yes, it is given the right to send.

Point-to-point: A point-to-point link is a circuit that connects two nodes only. (Contrast with **Multipoint line**.) A point-to-point configuration requires a separate physical connection between each pair of nodes.

Power: A measure that expresses the relationship between throughput and response time. It is calculated by dividing the mean throughput by the mean response time.

Power tuning: Involves finding parameter values that bring optimal power.

Process: A scheduled entity on an operating system.

Propagation time: Part of the transmission time; it is the time necessary for a signal to travel from one point on a circuit or network to another. Its starting point is when the first byte of a message enters the transmission medium; its ending point is when the first byte of the message reaches the destination. See **Transmission time**.

Protocol: A basic procedure or set of rules that controls the communications between computers. Also, a set of conventions between communicating processes regarding the format and contents of messages to be exchanged. The rules of "etiquette" determine the behavior that the communicating devices can expect of each other.

Protocol overhead: That part of communications data or processing which is not directly "consumed" by the users, but which is necessary to successfully bring about the transfer of user information. See **Control data.**

Public switched network: Any switching system that provides circuit switching to many customers. In the U.S.A. there are four such networks: Telex, TWX, telephone, and Broadband Exchange.

Queue: A group of data items waiting for service from a device or processor. For example, if incoming data arrive at a rate faster than the device can process them, the data must enter a queue and wait until the device processes the data ahead of them in the queue. Items of data in a queue gain attention from the device according to the priority scheme imposed by the system (for example, first-come-first-served).

Queuing delays: The length of time that queued data has to wait its turn for service from a device (server). The busier the device, or the slower the device (in relation to the arrival rate of data), the longer the queues will become and, therefore, the longer the delays. Queues are usually where most delays in a network occur.

Queuing theory: A scientific discipline that studies queuing delays and operations. It is useful for predicting queuing delays. A study of the patterns involved and the time required for discrete units to move through channels, such as elapsed time for auto traffic at a toll booth, or for data packets waiting for access to a network.

Range: The simplest measure of dispersion, which states the distance between the smallest and largest recorded values.

Real time: A real-time system is one that performs computation during the actual time the related physical process transpires, so that the results of the computation can be used in guiding the process. Contrast with **Batch processing.**

Real-time applications: Applications such as factory manufacturing control processing, traffic control, environmental monitoring, medical monitoring, and other instrumentation and control systems. Quick response times for such operations are vital to their success.

Reliability: The amount of time a piece of equipment, a system, or a network is available, how often it fails, and how easy it is to maintain. Also, concerns the degree of data integrity maintained in data transmissions.

Resource: Any part of the network where traffic is processed, stored, or channeled, such as central processor units (CPUs) at each node, storage disks, communications lines, and

line interfaces. An application requires one or more network resources at each stage of execution, and if it does not have them all, it is delayed or blocked from execution until the necessary resources are available.

Resource utilization: A measure of how much time a resource is busy. Measured as a percentage. For example, 80% CPU utilization means that the CPU is busy 80% of the time it is operating. The remaining 20% is called idle time. See **Utilization.**

Response time: The time it takes a system or network to respond to a given input; the interval between an event and the response to that event. When an operator types a message at a terminal, sending the message to a remote location in the network, the response time is the interval between the operator's pressing the last key of the input message and the terminal's typing the first letter of the response or a prompt.

Repeater: A bi-directional device that amplifies or resynchronizes signals into standard voltages, currents, and timing.

Retransmission: A method of error control: stations that receive messages acknowledge the receipt of correct messages and, on receipt of incorrect messages, either do not acknowledge or acknowledge in the negative. The lack of acknowledgment or receipt of a negative acknowledgment causes the sending station to retransmit the failed message.

Ring: A network topology in which nodes are connected to one another in a closed, logical circle. Typically, access to the media passes sequentially from one node to the next by means of polling from a master station, or by passing an access token from one station to another.

Router servers: Communications servers which offload routing functions from other nodes in a network.

Routing: The routing protocol in a network determines the path or **route** along which the data travels to reach its destination. Where multiple paths to a destination are available, routing chooses the best path. See **Adaptive routing.**

Saturation: The point when the demand on a resource reaches its maximum capacity. A saturated resource cannot handle any further demands until previous demands are processed.

Segmentation: The mechanism by which network software divides normal data (handed to the network in buffers) into numbered segments for transmission over a logical link. At the receiving end, the data is concatenated into its original form.

Server: A hardware and software module or device that performs a specific, well-defined service for many users, such as terminal I/O handling (terminal server) or remote file access. A network node that manages the sharing of resources independently of a host processor

Signaling rate: The rate at which signals are transmitted over the transmission medium, irrespective of extrinsic factors that decrease the signaling capacity available for the user application. See also **Baud, Line speed.** Contrast with **Transfer rate, data.**

Simplex: A mode of communication in which data is transferred in one direction only. Contrast with **Full duplex.**

Simulation: A hypothetical situation constructed by analysts to measure the results of a test or study. A simulation usually consists of one or more computer programs designed to imitate the system under study for purposes of analysis.

Skewed distribution: A distribution of data that is not symmetrical. See Appendix B.

SNA (IBM System Network Architecture): A network for moving IBM mainframe data. Digital's SNA Gateway product allows a node not directly connected to an IBM SNA network to access the facilities of the SNA network for terminal access and remote job entry.

Specification limit: The performance limit specified for an application or component.

Standard deviation: A statistical measure of the dispersion of a distribution of data. It characterizes to what extent the data varies from the mean. See Appendix B.

Star: A network topology in which one central node is connected to two or more adjacent, end nodes. The central node makes all message routing decisions.

Station: A physical termination of a line, having both a hardware and software implementation (that is, a controller and/or a unit). A server, such as a workstation or print station, at which direct user interaction occurs.

Statistical time division multiplexing (STDM): A form of multiplexing that is more intelligent and efficient than **time division multiplexing** and **frequency division multiplexing.** The devices, also called "stat muxes," allocate time (for use of the line) for active users only. The user who has nothing to communicate does not get a time slot, so other users get more opportunities to use the line, and less bandwidth is wasted.

Stat mux: See **Statistical time division multiplexing.**

Store-and-forward: A method of switching messages in which a message from one location to another is received at a computer acting as a switch and is temporarily stored there. After the computer determines the destination address and an available communications circuit, it forwards the message toward its destination. This method is also called **message switching.**

Switched line: A communications link for which the physical path may vary with each use. The dialup telephone network is an example of a switched line.

Switching: In data communications, identifying and connecting independent transmission links to form a temporary, continuous path from the source to the destination. Contrast with **Dedicated line.** See also **Packet switching, Circuit switching.**

Synchronous transmission: Transmissions that send messages at a definite fixed rate with the transmitter and receiver synchronized. Synchronous devices store and send data in blocks, rather than a character at a time. Greater efficiency is gained by not having to use start and stop bits with each character as in asynchronous transmissions. Synchronous transmissions send a predetermined group of "sync" characters ahead of a long stream of data. The sync characters enable the communicating devices to synchronize with each other in accord with time clocks at each end.

Terminal servers: A system that handles terminal operations for host nodes on the LAN. Terminal servers can be used to connect terminal users to nodes on the same LAN. Some terminal servers connect users to nodes located off the local LAN. Terminal servers offload the terminal connection and I/O responsibilities from host nodes on the LAN. Terminal servers reduce the number of direct terminal connections to each host, saving substantial power, packaging, and cabling expenses.

Throughput: In general, throughput is the measure of how much data is sent, or can be sent, between two points in a specified unit of time.

Throughput efficiency: Achievable, user throughput of a component or network. A component with high throughput efficiency has a high ratio of real throughput to rated throughput.

Throughput, gross: (Also called **total throughput.**) Measures the total amount of data transferred in time, including the protocol overhead as well as the "consumable" user part of the data. Contrast with **Throughput, net.**

Throughput, net: (Also called **effective throughput** or **productive throughput**) Measures the amount of user data transferred, discounting network or protocol data. User data means that which the user directly uses or "consumes." Network or protocol control data is useful, but not directly used by the user. Contrast with **Throughput, gross.**

Throughput, network: The sum of the lengths of all terminating messages received on all nodes across a network per second, where a terminating message is any packet that has reached its destination node. Also called the aggregate throughput.

Throughput, rated: The theoretical or potential throughput of a component, based on its capacity and irrespective of actual, applied conditions that will detract from it in real life. Contrast with **Throughput, real.**

Throughput, real: The measured processing rate or transfer rate of the user information. This meaning of throughput is sometimes called the **transfer rate of information blocks (TRIB).** It is less than capacity (rated throughput) because of the time required by the application or network to prepare the data for transmission.

Throughput saturation point: The point when the throughput of a component or network stops increasing: further increases of input do not provide additional throughput. See **Saturation.**

Timesharing: A method of allowing many users to share a resource simultaneously by dividing its available time into slices, each devoted to a particular user's task. If time slices occur frequently enough, each user thinks he or she has sole use of the computer. Processing appears real-time. Contrast with **Real time** and **Batch processing.**

Time division multiplexing (TDM): A form of multiplexing that allocates time slots to users wishing to transmit over a channel. Each user is given a time slot whether there is something to send or not. See also **Frequency division multiplexing,** and **Statistical time division multiplexing.**

Token passing: A scheme used by networks for controlling access to the network. Usually used in ring networks. Permission to use the network is given in the form of a token, usually an 8-bit data unit that is passed from node to node around a ring. When a node has a message to send, it grabs the token. Possession of the token gives the node exclusive access to the network. When the node finishes transmitting, it reinserts the token into the ring so other nodes may have a turn.

Topology: The physical shape or arrangement of interconnected nodes in a network. Where possible, the topology should be based on the nature of the traffic flow in the network; otherwise, performance can be hampered significantly.

Total throughput: See **Throughput, gross.**

Traffic: The measurement of data flow, volume, and velocity over a communications link.

Transceiver: A device required in baseband networks that takes the digital signal from a computer or terminal and imposes it on the baseband medium.

Transfer rate, data: The actual measured rate at which data is transferred, such as bits per second. It does not take into account intervals between transmissions of bits. Therefore, the measured transfer rate is less than the maximum transfer rate. See **Throughput, real.**

Transfer rate, maximum: The signaling capacity of a component, or its theoretical throughput, disregarding any limiting factors of real applications. Synonymous with **line speed** or **signaling rate** of a communications line. Contrast with **Transfer rate, data.** See **Throughput, rated.**

Transmission time: The time required to transmit a unit of data from the source node to the destination. The starting point for measuring the transmission time of a packet is the moment the first byte enters the network medium. The ending point is the moment when

the last byte of that packet is received by the destination. Thus, transmission time includes the time taken to transmit the packet out onto the network. In contrast, propagation time only consists of the time for the data to travel across the network. The size of the unit of data affects the transmission time, but not the propagation time. See **Propagation time.**

Transparency: A property of a transmission medium to pass within specified limits a range of signals having one or more defined properties; for example, a channel may be code transparent, or a device may be bit-pattern transparent.

Transparent: Network access is transparent to users when commands that access operations on local and remote systems are very similar; that is, the command syntax does not reveal whether the command involves the network.

Transparent data: With this feature, recognition of most control characters in a data transmission is suppressed. It is the ability to let any bit pattern appear in the data portion of the message without being interpreted as a control message. Without transparency, if user data should include bit patterns that can be interpreted as a control message, the data will be acted upon as a control message rather than as user information. DDCMP provides data transparency because it can recognize, without misinterpretation, data containing bit patterns that resemble DDCMP control characters.

TRIB: Transfer rate of information blocks. See **Throughput, real.**

Tributary station: In DECnet multipoint links, a station on a multipoint line that is not a control station.

Trimmed mean: A statistic that tests the calculated arithmetic mean's sensitivity to extremes. The trimmed mean is found by discarding an equal number of extreme values, and repeating the process until the trimmed mean is constant.

Tuning: The process of optimizing the performance of a program or component. Tuning may involve changing the way space is allocated in memory, changing the message or buffer size, and so forth.

Turnaround time: The elapsed time between submitting a job and receiving the output. A form of response time, often used to measure the performance of batch applications. Also denotes the time taken to change the direction of transmission (from sending to receiving, or vice versa) over a half-duplex line. Time is required by line propagation effects, modem timing, and computer reaction. Lengthy turnaround times can degrade performance considerably.

Twisted-pair cable: An inexpensive cable used in low-speed communications, similar to a telephone cable. Because of its narrow bandwidth, it is used only in a baseband network. It is susceptible to noise.

T1 Carrier: A time-division-multiplex transmission system introduced by the Bell System in 1962. Eight thousand times per second, an 8-bit sample from each of 24 voice channels is transmitted, along with a framing bit. The transmission of 193 bits 8,000 times per second produces a 1,544-megabit transmission rate.

Up time: The amount of time that a system or network is operable, as opposed to **down time.**

User: A user can be a human operator at a terminal, an unattended device, or an application program. Network communications usually consist of two operators at interactive terminals, two programs swapping messages, or an operator who wants to run a remote program or open a remote file.

User information: The information (message) that the source user wishes to convey to the destination user.

Utilization: A measure that describes the percentage of time that a resource is busy. It indicates how much use a resource is getting.

Utilization factor: For a component, this measures the ratio of the average arrival rate (incoming traffic) to the average service rate. This measure is important for studying queuing delays within a device or network.

Utilization, gross: Measures the percentage of a component's capacity used by all data, including the protocol bits. Contrast with **Utilization, net.**

Utilization, net: Reflects the percentage of a resource's capacity that is devoted to, or occupied by, user data only. Contrast with **Utilization, gross.**

Virtual terminal: A terminal which is physically connected to one node in a network but which, by use of the network software, may access other nodes and perform the same sorts of activities on them as a directly connected terminal. (On VAX/VMS systems, this is done by the SET HOST command.)

Voice-grade: Referring to a communications link: the telephone line, the most common medium used for data communications. A communications line optimized for voice transmission; it has not had its capabilities to transmit data enhanced to remove noises for the lines. A telephone line with enhanced data communications capabilities (noise reduction) is called a conditioned line.

Volatile information: Application data that quickly loses value as delay increases.

WAN (wide area network): A public or private data communications system that spans a large geographical area, with nodes distributed in several cities or countries.

Window: A range of data packets authorized to be in transit across a network.

Window size: The maximum number of data packets that can be in transit at a time.

Window flow control: A form of flow control based on the window size. When the window size is reached, further transmission stops until the number of outstanding packets drops below the window. The optimum window size brings maximum throughput. If the window size is too small, throughput is restricted. However, if the window is too large, throughput will start to decrease when the network becomes congested.

X.25 Gateway Access Protocol: In DECnet, a protocol which allows a node that is not directly connected to a public data network to access facilities of that network through an intermediary gateway node. X.25 is the protocol standard for communication in packet-switched data networks.

INDEX

U

W