“The Medium is the Message” because it is the medium that shapes and controls the search and form of human associations and actions.

Marshall McLuhan
*Understanding Media* (1964)

The marvels—of film, radio, and television—are marvels of one-way communication, which is not communication at all.

Milton Mayer
*On the Remote Possibility of Communication* (1967)
8.1 Introduction

Thus far we have covered the components of a single computer, which has been the traditional focus of computer architecture. In this chapter we see how to connect computers together, forming a community of computers. Figure 8.1 shows the generic components of this community: computer nodes, hardware and software interfaces, links to the interconnection network, and the interconnection network. Interconnection networks are also called networks or communication subnets, and nodes are sometimes called end systems or hosts. The connection of two or more interconnection networks is called internetworking, which relies on communication standards to convert information from one kind of network to another. The Internet is the most famous example of internetworking.

There are two reasons that computer architects should devote attention to networking. In addition to providing external connectivity, Moore’s Law shrunk networks so much that they connect the components within a single computer. Using a network to connect autonomous systems within a computer has long been found in mainframes, but today such designs can be found in PCs too. Switches are replacing buses as the normal communication technique: between
computers, between I/O devices, between boards, between chips, and even between modules inside chips. As a result, computer architects must understand networking terminology, problems and solutions in order to design and evaluate modern computers.

The second reason architects should study networking is that today almost all computers are--or will be--networked to other devices. Thus, understanding networking is critical; any device without a network is somehow flawed. Just as a modern computer without a memory hierarchy “broken”—hence a chapter just for it—a modern computer without a network is “broken” too. Hence this chapter.

This topic is vast, with portions of Figure 8.1 the subject of whole books and college courses. Networking is also a buzzword-rich environment, where many simple ideas are obscured behind acronyms and unusual definitions. To help you breakthrough the buzzword barrier, Figure 8.2 is a glossary of about 80 networking terms. The goal of this chapter is to provide computer architects a gentle, qualitative introduction to networking. It defines terms, helps you understand the architectural implications of interconnection network technology, provides introductory explanations of the key ideas, and give references to more detailed descriptions.

Most of this chapter is on networking, but the final quarter of this chapter focuses on clusters. A *cluster* is the coordinated use of interconnected computers in a machine room. In contrast to the qualitative network introduction, these sections give a more quantitative description of clusters, including many examples. It ends with a guided tour of the Google clusters.
### Term | Definition
--- | ---
**adaptive routing** | Router picks best path based upon measure of delay on outgoing links
**ATM** | Asynchronous Transfer Mode is a WAN designed for real-time traffic such as digital voice
**attenuation** | Loss of signal strength as signal passes through the medium over a long distance
**backpressure flow control** | When the receiver cannot accept another message, separate wires between adjacent senders and receivers tell the sender to stop immediately. It causes links between two end points to freeze until the receiver makes room for the next message.
**bandwidth** | Maximum rate the network can propagate information once the message enters the it
**base station** | A network architecture that uses boxes connected via land lines to communicate to wireless handsets
**bisection bandwidth** | Sum of the bandwidth of lines that cross that imaginary dividing line between two roughly equal parts of the network, each with half the nodes
**bit error rate** | BER, the error rate of a network, typically in errors per million bits transferred
**blade** | A removable computer component that fits vertically into a box in a standard VME rack
**blocking** | Contention that prevents a message from making progress along a link of a switch
**bridge** | OSI layer 2 networking device that connects multiple LANs, which can operate in parallel; in contrast, a router connects networks with incompatible addresses at OSI layer 3
**category 5 wire** | “Cat 5” twisted-pair, copper wire used for 10, 100, and 1000 Mbits/sec LANs
**carrier sensing** | “Listening” to the medium to be sure it is unused before trying to send a message
**channel** | In wireless networks, it is a pair of frequency bands that allow 2-way communication
**checksum** | A field of a message for a error correction code
**circuit switching** | A circuit is established from source to destination, reserving bandwidth along a path until the circuit is broken
**cluster** | Coordinated use of interconnected computers in a machine room
**coaxial cable** | A single stiff copper wire is surrounded by insulating material and a shield; historically faster and longer distance than twisted pair copper wire
**collision** | Two nodes (or more) on a shared medium try to send at the same time
**collision detection** | “Listening” to shared medium after sending to see if a message collided with another

**FIGURE 8.2** Networking terms in this chapter and their definitions
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>collocation site</td>
<td>A warehouse for remote hosting of servers with expansible networking, space, cooling, and security</td>
</tr>
<tr>
<td>communication</td>
<td>Another name for interconnection network</td>
</tr>
<tr>
<td>subnets</td>
<td></td>
</tr>
<tr>
<td>credit-based flow</td>
<td>To reduce overhead for flow control, a sender is given a credit to send up to N packets, and only checks for network delays when the credit is spent</td>
</tr>
<tr>
<td>control</td>
<td></td>
</tr>
<tr>
<td>cut-through routing</td>
<td>The switch examines the header, decides where to send the message, and then start transmitting it immediately without waiting for the rest of the message. When the head of the message blocks, the message stays strung out over the network.</td>
</tr>
<tr>
<td>destination-based routing</td>
<td>The message contains a destination address, and the switch picks a path to deliver the message, often by table lookup</td>
</tr>
<tr>
<td>deterministic routing</td>
<td>Router always picks the same path for the message</td>
</tr>
<tr>
<td>end systems</td>
<td>Another name for interconnection network node as opposed to the intermediate switches</td>
</tr>
<tr>
<td>end-to-end argument</td>
<td>Intermediate functions (error checking, performance optimization, and so on) may be incomplete as compared to performing the function end-to-end</td>
</tr>
<tr>
<td>Ethernet</td>
<td>The most popular LAN, it has scaled from its original 3 Mbits/second rate using shared media in 1975 to switched media at 1000 Mbits/second in 2001; it shows no signs of stopping</td>
</tr>
<tr>
<td>fat tree</td>
<td>A network topology with extra links at each level enhancing a simple tree, so bandwidth between each level is normally constant (see Figure 8.14 on page 595)</td>
</tr>
<tr>
<td>FC-AL</td>
<td>Fibre Channel Arbitrated Loop; a SAN for storage devices</td>
</tr>
<tr>
<td>frequency-division multiplexing</td>
<td>Divide the bandwidth of the transmission line into a fixed number of frequencies, and assign each frequency to a conversation.</td>
</tr>
<tr>
<td>full duplex</td>
<td>Two-way communication on a network segment</td>
</tr>
<tr>
<td>header</td>
<td>The first part of a message that contains no user information, but contents helps that network, such as providing the destination address</td>
</tr>
<tr>
<td>host</td>
<td>Another name for interconnection network node</td>
</tr>
<tr>
<td>hub</td>
<td>An OSI layer 1 networking device that connects multiples LANs to act as one</td>
</tr>
<tr>
<td>Infiniband</td>
<td>An emerging standard SAN for both storage and systems in a machine room</td>
</tr>
</tbody>
</table>

**FIGURE 8.2** Networking terms in this chapter and their definitions
### 8.1 Introduction

**interference**
In wireless networks, reduction of signal due to frequency reuse; frequency is reused to try to increase the number of simultaneous conversations over a large area.

**internetworking**
Connection of two or more interconnection networks.

**IP**
Internet Protocol is an OSI layer 3 protocol, at the network layer.

**iSCSI**
SCSI over IP networks, it is a competitor to SANs using IP and Ethernet switches.

**LAN**
Local Area Network, for machines in a building or campus, such as Ethernet.

**message**
The smallest piece of electronic “mail” sent over a network.

**multimode fiber**
An inexpensive optical fiber that reduces bandwidth and distance for cost.

**multipath fading**
In wireless networks, interference between multiple versions of signal that arrive at different times, determined by time between fastest signal and slowest signal relative to signal bandwidth.

**multistage switch**
a switch containing many smaller switches that perform a portion of routing.

**OSI layer**
Open System Interconnect models the network as seven layers (see 8.25 on page 612).

**overhead**
In this chapter, networking overhead is sender overhead + receiver overhead + time of flight.

**packet switching**
In contrast to circuit switching, information is broken into packets (usually fixed or maximum size), each with its own destination address, and they are routed independently.

**payload**
The middle part of the message that contains user information.

**peer-to-peer protocol**
Communication between two nodes occurs logically at the same level of the protocol.

**peer-to-peer wireless**
Instead of communicating to base stations, peer-to-peer wireless networks communicate between handsets.

**protocol**
The sequence of steps that network software follows to communicate.

**rack unit**
An R.U. is 1.7 inches, the height of a single slot in a standard 19-inch VME rack; there are 44 R.U. in standard 6-foot rack.

**receiver overhead**
The time for the processor to pull the message from the interconnection network.

**router**
OSI layer 3 networking device that connects multiples LANs with incompatible addresses.

**SAN**
Originally System Area Network but more recently Storage Area Network, it connects computers and/or storage devices in a machine room. FC-AL or Infiniband are SANs.

---

**FIGURE 8.2** Networking terms in this chapter and their definitions.
**Term** | **Definition**
--- | ---
sender overhead | The time for the processor to inject the message into the network; the processor is busy for the entire time
shadow fading | In wireless networks, when the received signal is blocked by objects; buildings outdoors or walls indoors
signal-to-noise ratio | SNR, the ratio of the strength of the signal carrying information to the background noise
simplex | One-way communication on a network segment
single-mode fiber | “Single-wavelength” fiber is narrower and more expensive than multimode fiber but it offers greater bandwidth and distance
source-based routing | The message specifies the path to the destination at each switch
store-and-forward | Each switch waits for the full message to arrive before it is sent on to the next switch
TCP | Transmission Control Protocol, it is an OSI layer 4 protocol (transport layer)
throughput | In networking, measured speed of the medium or network bandwidth delivered to an application; i.e., does not give credit for headers and trailers
time of flight | The time for the first bit of the message to arrive at the receiver
trailer | The last part of a message that has no user information but helps the network, such as error correction code
transmission time | The time for the message to pass through the network (not including time of flight)
transport latency | Time that the message spends in the interconnection network (including time of flight)
twisted pairs | Two wires twisted together to reduce electrical interference
virtual circuit | A logical circuit is established between source and destination for a message to follow
WAN | Wide Area Network, a network across a continent, such as ATM
wavelength division multiplexing | WDM sends different streams simultaneously on the same fiber using different wavelengths of light and then demultiplexes the different wavelengths at the receiver
window | In TCP, the number of TCP datagrams that can be sent without waiting for approval
wireless network | A network that communicates without physical connections, such as radio
wormhole routing | The switch examines the header, decides where to send the message, and then starts transmitting it immediately without waiting for the rest of the message. The tail continues when the head blocks, potentially compressing the strung-out message into a single switch

**FIGURE 8.2** Networking terms in this chapter and their definitions
8.1 Introduction

Let’s start with the generic types of interconnections. Depending on the number of nodes and their proximity, these interconnections are given different names:

- **Wide area network (WAN)**—Also called *long haul network*, the WAN connects computers distributed throughout the world. WANs include thousands of computers, and the maximum distance is thousands of kilometers. ATM is a current example of a WAN.

- **Local area network (LAN)**—This device connects hundreds of computers, and the distance is up to a few kilometers. Unlike a WAN, a LAN connects computers distributed throughout a building or on a campus. The most popular and enduring LAN is Ethernet.

- **Storage or System area network (SAN)**—This interconnection network is for a machine room, so the maximum distance of a link is typically less than 100 meters, and it can connect hundreds of nodes. Today SAN usually means Storage area network as it connects computers to storage devices, such as disk arrays. Originally SAN meant a System area network to connect computers together, such as PCs in a cluster. A recent SAN trying to network both storage and system is Infiniband.

Figure 8.3 shows the rough relationship of these systems in terms of number autonomous systems connected, including a bus for comparison. Note the area of overlap between buses, SANs, and LANs, which lead to product competition.

![Figure 8.3](image-url)  

**FIGURE 8.3** Relationship of four types of interconnects in terms of number of autonomous systems connected: bus, system or storage area network, local area network, and wide area network/Internet. Note that there are overlapping ranges where buses, SANs, and LANs compete. Some supercomputers have a switch-based custom network to interconnect up to thousands of computers; such interconnects are basically custom SANs.
These three types of interconnection networks have been designed and sustained by several different cultures—Internet, telecommunications, workgroup/enterprise, storage, and high performance computing—each using its own dialects and its own favorite approaches to the goal of interconnecting autonomous computers.

This chapter gives a common framework for evaluating all interconnection networks, using a single set of terms to describe the basic alternatives. Figure 8.22 in section 8.7 gives several other examples of each of these interconnection networks. As we shall see, some components are common to all types and some are quite different.

We begin the chapter in section 8.2 by exploring the design and performance of a simple network to introduce the ideas. We then consider the following problems: which media to use as the interconnect (8.3), how to connect many computers together (8.4 and 8.5), and what are the practical issues for commercial networks (8.6). We follow with examples illustrating the trade-offs for each type of network (8.7), explore internetworking (8.8), and cross cutting issues for networks (8.9). With this gentle introduction to networks in sections 8.2 to 8.9, readers interested in more depth should try the suggested reading in section 8.16. Sections 8.10 to 8.12 switch to clusters, and give a more quantitative description with designs and examples. Section 8.13 gives a view of networks from the embedded perspective, using a cell phone and wireless networks as the example. We conclude in sections 8.14 to 8.16 with the traditional ending of the chapters.

As we shall see, networking shares more characteristics with storage than with processors and memory. Like storage, the operating system controls what features of the network are used. Again like storage, performance includes both latency and bandwidth, and queueing theory is a valuable tool. Like RAID, networking assumes failures occur, and thus dependability in the presence of errors is the norm.

8.2 A Simple Network

There is an old network saying: Bandwidth problems can be cured with money. Latency problems are harder because the speed of light is fixed—you can’t bribe God.

Anonymous

To explain the complexities and concepts of networks, this section describes a simple network of two computers. We then describe the software steps for these two machines to communicate. The remainder of the section gives a detailed and then a simple performance model, including several examples to see the implications of key network parameters.
Suppose we want to connect two computers together. Figure 8.4 shows a simple model with a unidirectional wire from machine A to machine B and vice versa. At the end of each wire is a first-in-first-out (FIFO) queue to hold the data. In this simple example, each machine wants to read a word from the other’s memory. A *message* is the information sent between machines over an interconnection network.

For one machine to get data from the other, it must first send a request containing the address of the data it desires from the other node. When a request arrives, the machine must send a reply with the data. Hence, each message must have at least 1 bit in addition to the data to determine whether the message is a new request or a reply to an earlier request. The network must distinguish between information needed to deliver the message, typically called the *header* or the *trailer* depending on where it is relative to the data, and the *payload*, which contains the data. Figure 8.5 shows the format of messages in our simple network. This example shows a single-word payload, but messages in some interconnection networks can include hundreds of words.

Interconnection networks involve normally software. Even this simple example invokes software to translate requests and replies into messages with the appropriate headers. An application program must usually cooperate with the operating system to send a message to another machine, since the network will be shared with all the processes running on the two machines, and the operating system cannot allow messages for one process to be received by another. Thus, the messaging software must have some way to distinguish between processes; this distinction may be included in an expanded header. Although hardware support can reduce the amount of work, most is done by software.

In addition to protection, network software is often responsible for ensuring reliable delivery of messages. The twin responsibilities are ensuring that the message is neither garbled nor lost in transit.
Adding a checksum field (or some other error detection code) to the message format meets the first responsibility. This redundant information is calculated when the message is first sent and checked upon receipt. The receiver then sends an acknowledgment if the message passes the test.

One way to meet the second responsibility is to have a timer record the time each message is sent and to presume the message is lost if the timer expires before an acknowledgment arrives. The message is then re-sent.

The software steps to send a message are as follows:

1. The application copies data to be sent into an operating system buffer.
2. The operating system calculates the checksum, includes it in the header or trailer of the message, and then starts the timer.
3. The operating system sends the data to the network interface hardware and tells the hardware to send the message.

Message reception is in just the reverse order:

1. If the data pass the test, the system copies the data to the user’s address space and signals the application to continue.
2. The system calculates the checksum over the data. If the checksum matches the sender’s checksum, the receiver sends an acknowledgment back to the sender. If not, it deletes the message, assuming that the sender will resend the message when the associated timer expires.
3. The system copies the data from the network interface hardware into the operating system buffer.

The sender must still react to the acknowledgment:
When the sender gets the acknowledgment, it releases the copy of the message from the system buffer.

If the sender gets the time-out instead of an acknowledgment, it resends the data and restarts the timer.

Here we assume that the operating system keeps the message in its buffer to support retransmission in case of failure. Figure 8.6 shows how the message format looks now.

The sequence of steps that software follows to communicate is called a protocol and generally has the symmetric but reversed steps between sending and receiving.

Note that this example protocol above is for sending a single message. When an application does not require a response before sending the next message, the sender can overlap the time to send with the transmission delays and the time to receive.

A protocol must handle many more issues than reliability. For example, if two machines are from different manufacturers, they might order bytes differently within a word (see section 2.3 of Chapter 2). The software must reverse the order of bytes in each word as part of the delivery system. It must also guard against the possibility of duplicate messages if a delayed message were to become unstuck. It is often necessary to deliver the messages to the application in the order they are sent, and so sequence numbers may be added to the header to enable assembly. Finally, it must work when the receiver’s FIFO becomes full, suggesting feedback to control the flow of messages from the sender (see section 8.4).

Now that we have covered the steps in sending and receiving a message, we can discuss performance.

Figure 8.7 shows the many performance parameters of interconnection networks. This figure is critical to understanding network performance, so study it
well! Note that the parameters in Figure 8.7 apply to the interconnect in many levels of the system: inside a chip, between chips on a board, between computers in a cluster, and so on. The units change, but the principles remain the same, as does the bandwidth that results.

These terms are often used loosely, leading to confusion, so we define them here precisely:

- **Bandwidth**—We use this most widely used term to refer to the maximum rate at which the network can propagate information once the message enters the network. Unlike disks, bandwidth includes the headers and trailers as well as the payload, and the units are traditionally bits/second rather than bytes/second. The term bandwidth is also used to mean the measured speed of the medium or network bandwidth delivered to an application. *Throughput* is sometimes used for this latter term.

- **Time of flight**—The time for the first bit of the message to arrive at the receiver, including the delays due to repeaters or other hardware in the network. Time of flight can be milliseconds for a WAN or nanoseconds for an SAN.

- **Transmission time**—The time for the message to pass through the network, not including time of flight. One way to measure it is the difference in time between when the first bit of the message arrives at the receiver and when the last bit of the message arrives at the receiver. Note that by definition transmission time is equal to the size of the message divided by the bandwidth. This measure assumes there are no other messages to contend for the network.
Transport latency—The sum of time of flight and transmission time. Transport latency is the time that the message spends in the interconnection network. Stated alternatively, it is the time between when the first bit of the message is injected into the network and when the last bit of the message arrives at the receiver. It does not include the overhead of injecting the message into the network nor pulling it out when it arrives.

Sender overhead—The time for the processor to inject the message into the network, including both hardware and software components. Note that the processor is busy for the entire time, hence the use of the term overhead. Once the processor is free, any subsequent delays are considered part of the transport latency. For pedagogic reasons, we assume overhead is not dependent on message size. (Typically, only very large messages have larger overhead.)

Receiver overhead—The time for the processor to pull the message from the interconnection network, including both hardware and software components. In general, the receiver overhead is larger than the sender overhead: for example, the receiver may pay the cost of an interrupt.

The total latency of a message can be expressed algebraically:

\[
\text{Total latency} = \text{Sender overhead} + \text{Time of flight} + \frac{\text{Message size}}{\text{Bandwidth}} + \text{Receiver overhead}
\]

Let’s look at how the time of flight and overhead parameters change in importance as we go from SAN to LAN to WAN.

**Example** Assume a network with a bandwidth of 1000 Mbits/second has a sending overhead of 80 microseconds and a receiving overhead of 100 microseconds. Assume two machines. One wants to send a 10000-byte message to the other (including the header), and the message format allows 10000 bytes in a single message. Let’s compare SAN, LAN, and WAN by changing the distance between the machines. Calculate the total latency to send the message from one machine to another in a SAN assuming they are 10 meters apart. Next, perform the same calculation but assume the machines are now 500 meters apart, as in a LAN. Finally, assume they are 1000 kilometers apart, as in a WAN.

**Answer** The speed of light is 299,792.5 kilometers per second in a vacuum, and signals propagate at about 63% to 66% of the speed of light in a conductor. Since this is an estimate, in this chapter we’ll round the speed of light to 300,000 kilometers per second, and assume we can achieve two-thirds of that in a conductor. Hence, we can estimate time of flight. Let’s plug the parameters for the short distance of a SAN into the formula above:
Converting all terms into microseconds (μsecs) leads to

\[
\text{Total latency} = \text{Sender overhead} + \text{Time of flight} + \frac{\text{Message size}}{\frac{\text{Bandwidth}}{2}} + \text{Receiver overhead} \\
= 80\mu\text{secs} + \frac{0.01\text{km}}{\frac{2\times3}{300,000\text{ km/sec}}} + \frac{10000\text{ bytes}}{1000\text{ Mbits/sec}} + 100\mu\text{secs}
\]

Substituting an example LAN distance into the third equation yields

\[
\begin{align*}
\text{Total latency} &= 80\mu\text{secs} + \frac{0.01\times10^6\mu\text{secs}}{2\times3\times300,000\text{ km/sec}} + \frac{10000\times8\mu\text{secs}}{1000\text{ Mbits/sec}} + 100\mu\text{secs} \\
&= 80\mu\text{secs} + 0.05\mu\text{secs} + 80\mu\text{secs} + 100\mu\text{secs} = 260 + 0.05\mu\text{secs} \\
&= 260\mu\text{secs}
\end{align*}
\]

Substituting the WAN distance into the equation yields

\[
\begin{align*}
\text{Total latency} &= 80\mu\text{secs} + \frac{0.5\text{km}}{2\times3\times300,000\text{ km/sec}} + \frac{10000\text{ bytes}}{1000\text{ Mbits/sec}} + 100\mu\text{secs} \\
&= 80\mu\text{secs} + 2.50\mu\text{secs} + 80\mu\text{secs} + 100\mu\text{secs} = 260 + 2.5\mu\text{secs} \\
&= 262\mu\text{secs}
\end{align*}
\]

The increased fraction of the latency required by time of flight for long distances, as well as the greater likelihood of errors over long distances, are why wide area networks use more sophisticated and time-consuming protocols. Complexity increases from protocols used on a bus versus a LAN versus the Internet as we go from tens to hundreds to thousands of nodes.

Note that messages in LANs and WANs go through switches which add to the latency, which we neglected above. Generally, switch latency is small compared to overhead in LANs or time of flight in SANs.

As mentioned above, when an application does not require a response before sending the next message, the sender can overlap the sending overhead with the transport latency and receiver overhead. Increased latency affects the structure of programs that try to hide this latency, requiring quite different solutions if the latency is 1, 100, or 10,000 microseconds.
Note that the example above shows that time of flight for SANs is so short relative to overhead that it can be ignored, yet in WANs, time of flight is so long that sender and receiver overheads can be ignored. Thus, we can simplify the performance equation by combining sender overhead, receiver overhead, and time of flight into a single term called Overhead:

\[
\text{Total latency} = \text{Overhead} + \frac{\text{Message size}}{\text{Bandwidth}}
\]

We can use this formula to calculate the effective bandwidth delivered by the network as message size varies:

\[
\text{Effective bandwidth} = \frac{\text{Message size}}{\text{Total latency}}
\]

Let’s use this simpler equation to explore the impact of overhead and message size on effective bandwidth.

**Example**

Plot the effective bandwidth versus message size for overheads of 25 and 250 microseconds and for network bandwidths of 100, 1000, and 10000 Mbits/second. Vary message size from 16 bytes to 4 megabytes. For what message sizes is the effective bandwidth virtually the same as the raw network bandwidth? If overhead is 250 microseconds, for what message sizes is the effective bandwidth always less than 100 Mbits/second?

**Answer**

Figure 8.8 plots effective bandwidth versus message size using the simplified equation above. The notation “Ox,bwY” means an overhead of X microseconds and a network bandwidth of Y Mbits/second. To amortize the cost of high overhead, message sizes must be four megabytes for effective bandwidth to be about the same as network bandwidth. Assuming the high overhead, message sizes about 3K bytes or less will not break the 100 Mbits/second barrier no matter what the actual network bandwidth.

Thus, we must lower overhead as well as increase network bandwidth unless messages are very large.

Hence, message size is important in getting full benefit of fast networks. What is the natural size of messages? Figure 8.9 above shows the size of Network File System (NFS) messages for 239 machines at Berkeley collected over a period of one week. One plot is cumulative in messages sent, and the other is cumulative in data bytes sent. The maximum NFS message size is just over 8 KB, yet 95% of the messages are less than 192 bytes long. Figure 8.10 below shows the similar results for Internet traffic, where the maximum transfer unit was 1500 bytes.
Again, 60% of the messages are less than 192 bytes long, and 1500-byte messages represented 50% of the bytes transferred. Many applications send far more small messages than large messages, since requests and acknowledgements are more frequent than data.

Summarizing this section, even this simple network has brought up the issues of protection, reliability, heterogeneity, software protocols, and a more sophisticated performance model. The next four sections address other key questions:

- Which media are available to connect computers together?
- What issues arise if you want to connect more than two computers?
- What practical issues arise for commercial networks?
FIGURE 8.9  Cumulative percentage of messages and data transferred as message size varies for NFS traffic. Each x-axis entry includes all bytes up to the next one; e.g., 32 represents 32 bytes to 63 bytes. More than half the bytes are sent in 8-KB messages, but 95% of the messages are less than 192 bytes. Figure 8.50 (page 651) shows details of this measurement. Collected at the University of California at Berkeley.

FIGURE 8.10  Cumulative percentage of messages and data transferred as message size varies for Internet traffic. About 40% of the messages were 40 bytes long, and 50% of the data transfer was in messages 1500 bytes long. The maximum transfer unit of most switches was 1500 bytes. Collect by Vern Paxton on MCI Internet traffic in 1998.
8.3 Interconnection Network Media

Just as there is a memory hierarchy, there is a hierarchy of media to interconnect computers that varies in cost, performance, and reliability. Network media have another figure of merit, the maximum distance between nodes. This section covers three popular examples, and Figure 8.11 illustrates them.

The first medium is twisted pairs of copper wires. These are two insulated wires, each about 1 mm thick. They are twisted together to reduce electrical interference, since two parallel lines form an antenna but a twisted pair does not. As they can transfer a few megabits per second over several kilometers without amplification, twisted pair were the mainstay of the telephone system. Telephone companies bundled together (and sheathed) many pairs coming into a building. Twisted pairs can also offer tens of megabits per second of bandwidth over shorter distances, making them plausible for LANs.

FIGURE 8.11 Three network media. (From a presentation by David Culler of U.C. Berkeley.)
The desire to go at higher speeds with the less expensive copper led to improvements in the quality of unshielded twisted-pair copper cabling systems. The original telephone-line quality was called Level 1. Level 3 was good enough for 10 Mbits/second Ethernet. The desire for even greater bandwidth lead to the Level 5 or Category 5, which is sufficient for 100 Mbits/second Ethernet. By limiting the length to 100 meters, “Cat5” wiring can be used for 1000 Mbits/second Ethernet links today. It uses the RJ-45 connector, which is similar to the connector found on telephone lines.

*Coaxial cable* was deployed by cable television companies to deliver a higher rate over a few kilometers. To offer high bandwidth and good noise immunity, insulating material surrounds a single stiff copper wire, and then cylindrical conductor surrounds the insulator, often woven as a braided mesh. A 50-ohm baseband coaxial cable delivers 10 megabits per second over a kilometer.

Connecting to this heavily insulated media is more challenging. The original technique was a T junction: the cable is cut in two and a connector is inserted that reconnects the cable and adds a third wire to a computer. A less invasive solution is a vampire tap: a hole of precise depth and width is first drilled into the cable, terminating in the copper core. A connector is then screwed in without having to cut the cable.

To keep up with the demands of bandwidth and distance, it became clear that the telephone company would need to find new media. The solution could be more expensive provided that it offered much higher bandwidth and that supplies were plentiful. The answer was to replace copper with glass and electrons with photons. *Fiber optics* transmits digital data as pulses of light.

A fiber optic network has three components:

1. the transmission medium, a fiber optic cable;
2. the light source, an LED or laser diode;
3. the light detector, a photodiode.

First, cladding surrounds the glass fiber core to confine the light. A buffer then surrounds the cladding to protect the core and cladding. Note that unlike twisted pairs or coax, fibers are one-way, or *simplex*, media. A two-way, or *full duplex*, connection between two nodes requires two fibers.

Since light bends or refracts at interfaces, it can slowly spread as it travels down the cable unless the diameter of the cable is limited to one wavelength of light; then it transfers in a straight line. Thus, fiber optic cables are of two forms:

1. *Multimode fiber*—It uses inexpensive LEDs as a light source. It is typically much larger than the wavelength of light: typically 62.5 microns in diameter vs. the 1.3-micron wavelength of infrared light. Since it is wider it has more dispersion problems, where some wave frequencies have different propagation velocities. The LEDs and dispersion limit it to up to a few hundred meters at 1000 Mbits/second or a few kilometers at 100 Mbits/second. It is older and less expensive than single mode fiber.
2. Single-mode fiber—This “single-wavelength” fiber (typically 8 to 9 microns in diameter) requires more expensive laser diodes for light sources and currently transmits gigabits per second for hundreds of kilometers, making it the medium of choice for telephone companies. The loss of signal strength as it passes through a medium, called attenuation, limits the length of the fiber.

Although single-mode fiber is a better transmitter, it is much more difficult to attach connectors to single-mode; it is less reliable and more expensive, and the cable itself has restrictions on the degree it can be bent. The cost, bandwidth, and distance of single-mode fiber is affected by the power of the light source, the sensitivity of the light detector, and the attenuation rate per kilometer of the fiber cable. Typically, glass fiber has better characteristics than the less expensive plastic fiber, and so is more widely used.

Connecting fiber optics to a computer is more challenging than connecting cable. The vampire tap solution of cable fails because it loses light. There are two forms of T-boxes:

1. Taps are fused onto the optical fiber. Each tap is passive, so a failure cuts off just a single computer.

2. In an active repeater, light is converted to electrical signals, sent to the computer, converted back to light, and then sent down the cable. If an active repeater fails, it blocks the network.

These taps and repeaters also reduce optical signal strength, reducing the useful distance of a single piece of fiber.

In both cases, fiber optics has the additional cost of optical-to-electrical and electrical-to-optical conversion as part of the computer interface. Hence, the network interface cards for fiber optics are considerably more expensive than for Cat5 copper wire. In 2001, most switches for fiber involve such a conversion to allow switching, although expensive all optical switches are beginning to be available.

To achieve even more bandwidth from a fiber, wavelength division multiplexing (WDM) sends different streams simultaneously on the same fiber using different wavelengths of light, and then demultiplexes the different wavelengths at the receiver. In 2001, WDM can deliver a combined 40 Gbits/second using about 8 wavelengths, with plans to go to 80 wavelengths and deliver 400 Gbits/second.

The product of the bandwidth and maximum distance forms a single figure of merit: gigabit-kilometers per second. According to Desurvire [1992], since 1975 optical fibers have increased transmission capacity by tenfold every four years by this measure.

Let’s compare media in an example.
**Example** Suppose you have 25 magnetic tapes, each containing 40 GB. Assume that you have enough tape readers to keep any network busy. How long will it take to transmit the data over a distance of one kilometer? Assume the choices are Category 5 twisted pair wires at 100 Mbits/second, multimode fiber at 1000 Mbits/second, and single mode fiber at 2500 Mbits/second. How do they compare to delivering the tapes by car?

**Answer** The amount of data is 1000 GB. The time for each medium is given below:

- Twisted pair: \[ \frac{1000 \times 1024 \times 8 \text{ Mb}}{100 \text{ Mb/sec}} = 81,920 \text{ secs} = 22.8 \text{ hours} \]
- Multimode fiber: \[ \frac{1000 \times 1024 \times 8 \text{ Mb}}{1000 \text{ Mb/sec}} = 8192 \text{ secs} = 2.3 \text{ hours} \]
- Single-mode fiber: \[ \frac{1000 \times 1024 \times 8 \text{ Mb}}{2500 \text{ Mb/sec}} = 3277 \text{ secs} = 0.9 \text{ hours} \]

Car = Time to load car + Transport time + Time to unload car
= 300 secs + \[ \frac{1 \text{ km}}{30 \text{ kph}} \times 300 \text{ secs} = 300 \text{ secs} + 120 \text{ secs} + 300 \text{ secs} \]
= 720 secs = 0.3 hours

A car filled with high-density tapes is a high-bandwidth medium!

---

8.4 Connecting More Than Two Computers

*Computer power increases by the square of the number of nodes on the network.*

Robert Metcalf ("Metcalf’s Law")

Thus far, we have discussed two computers communicating over private lines, but what makes interconnection networks interesting is the ability to connect hundreds of computers together. And what makes them more interesting also makes them more challenging to build.

**Shared versus Switched Media**

Certainly the simplest way to connect multiple computers is to have them share a single interconnection medium, just as I/O devices share a single I/O bus. The most popular LAN, Ethernet, originally was simply a bus shared by a hundred of computers.

Given that the medium is shared, there must be a mechanism to coordinate and arbitrate the use of the shared medium so that only one message is sent at a time.
If the network is small, it may be possible to have an additional central arbiter to give permission to send a message. (Of course, this leaves open the question of how the nodes talk to the arbiter.)

Centralized arbitration is impractical for networks with a large number of nodes spread out over a kilometer, so we must distribute arbitration. A first step towards arbitration is looking before you leap. A node first checks the network to avoid trying to send a message while another message is already on the network. If the interconnection is idle, the node tries to send. Looking first is not a guarantee of success, of course, as some other node may decide to send at the same instant. When two nodes send at the same time, it is called a collision. Let’s assume that the network interface can detect any resulting collisions by listening to hear if the data were garbled by other data appearing on the line. Listening to avoid and detect collisions is called carrier sensing and collision detection. This is the second step of arbitration.

The problem is not solved. If every node on the network waited exactly the same amount of time, listened to be sure there was no traffic, and then tried to send again, we could still have synchronized nodes that would repeatedly bump heads. To avoid repeated head-on collisions, each node whose message was garbled waits (or “backs off”) a random time before resending. Note that randomization breaks the synchronization. Subsequent collisions result in exponentially increasing time between attempts to retransmit, so as not to tax the network.

Although this approach is not guaranteed to be fair—some subsequent node may transmit while those that collided are waiting—it does control congestion on the shared medium. If the network does not have high demand from many nodes, this simple approach works well. Under high utilization, performance degrades since the medium is shared.

Another approach to arbitration is to pass a token between the nodes, with the token giving the node the right to use the network. If the shared medium is connected in a ring, then the token can rotate through all the nodes on the ring.

Shared media have some of the same advantages and disadvantages as buses: they are inexpensive, but they have limited bandwidth. And like buses, they must have an arbitration scheme to solve conflicting demands.

The alternative to sharing the media is to have a dedicated line to a switch that in turn provides a dedicated line to all destinations. Figure 8.12 shows the potential bandwidth improvement of switches: Aggregate bandwidth is many times that of a single shared medium.

Switches allow communication directly from source to destination, without intermediate nodes to interfere with these signals. Such point-to-point communication is faster than a line shared between many nodes because there is no arbitration and the interface is simpler electrically. Of course, it does pay the added latency of going through the switch, trading off arbitration overhead for switching overhead.

Given the obvious advantages, why weren’t switches always used? Earlier computers were much slower and so could share media. In addition, applications
such as the World Wide Web rely on the network much more than older applications. Finally, earlier switches would take several large boards, and be as large as a computer. In 2001, a single chip contains a full 64-by-64 switch, or at least a large slice of it. Moore’s Law is making switches more attractive, and so technology trends favor switches today.

Every node of a shared line will see every message, even if it is just to check to see whether or not the message is for that node, so this style of communication is sometimes called broadcast to contrast it with point-to-point. The shared medium makes it easy to broadcast a message to every node, and even to broadcast to subsets of nodes, called multicasting.

Switches allow multiple pairs of nodes to communicate simultaneously, giving these interconnections much higher aggregate bandwidth than the speed of a shared link to a node. Switches also allow the interconnection network to scale to a very large number of nodes. Switches are also called data switching exchanges, multistage interconnection networks, or even interface message processors (IMPs). Depending on the distance of the node to the switch and desired bandwidth, the network medium is either copper wire or optical fiber.
EXAMPLE Compare 16 nodes connected three ways: a single 100 Mb/sec shared media; a switch connected via Cat5, each segment running at 100 Mb/sec; and a switch connected via optical fibers, each running at 1000 Mb/sec. The shared media is 500 meters long, and the average length of each segment to a switch is 50 meters. Both switches can support the full bandwidth. Assume each switch adds 5 microseconds to the latency. Calculate the aggregate bandwidth and transport latency. Assume the average message size is 125 bytes, and ignore the overhead of sending or receiving a message and contention for the network.

ANSWER The aggregate bandwidth of each example is the simplest calculation: 100 Mb/sec for the shared media; 16 × 100, or 1600 Mb/sec for the switched twisted pairs; and 16 × 1000, or 16000 Mb/sec for the switched optical fibers.

The transport time is

\[
\text{Transport time} = \frac{\text{Time of flight} \times \text{Message size}}{\text{Bandwidth}} + 5 \text{ µsecs}
\]

For coax we just plug in the distance, bandwidth, and message size:

\[
\text{Transport time}_{\text{shared}} = \frac{500/1000 \times 10^6}{2/3 \times 300,000} \mu\text{secs} + \frac{125 \times 8}{100} \mu\text{secs}
\]

\[
= 2.5 \mu\text{secs} + 0.8 \mu\text{secs}
\]

\[
= 12.5 \mu\text{secs}
\]

For the switches, the distance is twice the average segment, since there is one segment from the sender to the switch and one from the switch to the receiver. We must also add the latency for the switch.

\[
\text{Transport time}_{\text{switch}} = 2 \times \left( \frac{50/1000 \times 10^6}{2/3 \times 300,000} \mu\text{secs} + \frac{125 \times 8}{100} \mu\text{secs} \right) + 5 \mu\text{secs} + 10 \mu\text{secs}
\]

\[
= 0.5 \mu\text{secs} + 0.8 \mu\text{secs} + 5 \mu\text{secs} + 10 \mu\text{secs}
\]

\[
= 15.5 \mu\text{secs}
\]

\[
\text{Transport time}_{\text{fiber}} = 2 \times \left( \frac{50/1000 \times 10^6}{2/3 \times 300,000} \mu\text{secs} + \frac{125 \times 8}{1000} \mu\text{secs} \right) + 5 \mu\text{secs} + 1 \mu\text{secs}
\]

\[
= 0.5 \mu\text{secs} + 0.8 \mu\text{secs} + 5 \mu\text{secs} + 1 \mu\text{secs}
\]

\[
= 6.5 \mu\text{secs}
\]

Although the bandwidth of the switch is many times the shared media, the latency for unloaded networks is comparable.
Switches allow communication to harvest the same rapid advance from silicon as have processors and main memory. Whereas the switches from telecommunications companies were once the size of mainframe computers, today we see single-chip switches. Just as single-chip processors led to processors replacing logic in a surprising number of places, single-chip switches are increasingly replacing buses and shared media networks.

**Connection-Oriented versus Connectionless Communication**

Before computers arrived on the scene, the telecommunications industry allowed communication around the world. An operator set up a *connection* between a caller and a callee, and once the connection is established, a conversation can continue for hours. To share transmission lines over long distances, the telecommunications industry used switches to multiplex several conversations on the same lines. Since audio transmissions have relatively low bandwidth, the solution was to divide the bandwidth of the transmission line into a fixed number of frequencies, with each frequency assigned to a conversation. This technique is called *frequency-division multiplexing*.

Although a good match for voice, frequency-division multiplexing is inefficient for sending data. The problem is that the frequency channel is dedicated to the conversation whether or not there is anything being said. Hence, the long distance lines are “busy” based on the *number* of conversations, and not on the *amount* of information being sent at a particular time. An alternative style of communication is called *connectionless*, where each package is routed to the destination by looking at its address. The postal system is a good example of connectionless communication.

Closely related to the idea of connection versus connectionless communication are the terms *circuit switching* and *packet switching*. Circuit switching is the traditional way to offer a connection-based service. A circuit is established from source to destination to carry the conversation, reserving bandwidth until the circuit is broken. The alternative to circuit-switched transmission is to divide the information into *packets*, or *frames*, with each packet including the destination of the packet plus a portion of the information. Queuing theory in section 6.4 tells us that packets cannot use all of the bandwidth, but in general, this *packet-switched* approach allows more use of the bandwidth of the medium and is the traditional way to support connectionless communication.

**Example**

Let’s compare a single 1000 Mbits/sec packet switched network with ten 100 Mbits/sec packet-switched networks. Assume that the mean size of a packet is 250 bytes, the arrival rate is 250,000 packets per second, and the interarrival times are exponentially distributed. What is the mean response time for each alternative? What is the intuitive reason behind the difference?
From section 6.4 in the prior chapter, we can use an M/M/1 queue to calculate the mean response time for the single fast network:

\[ \text{Service rate} = \frac{\text{Bandwidth}}{\text{Message size}} = \frac{1000 \times 10^6}{250 \times 8} = \frac{1000 \times 10^6}{2000} = 500,000 \text{ packets per second} \]

\[ \text{Time}_{\text{server}} = \frac{1}{500,000} = 2 \mu\text{secs} \]

\[ \text{Utilization} = \frac{\text{Arrival rate}}{\text{Service rate}} = \frac{250,000}{500,000} = 0.5 \]

\[ \text{Time}_{\text{queue}} = \text{Time}_{\text{server}} \times \frac{\text{Server utilization}}{1 - \text{Server utilization}} = 2 \mu\text{secs} \times \frac{0.5}{1 - 0.5} = 2 \times 0.5 = 2 \mu\text{secs} \]

\[ \text{Mean response time} = \text{Time}_{\text{queue}} + \text{Time}_{\text{server}} = 2 + 2 = 4 \mu\text{secs} \]

The 10 slow networks can be modeled by an M/M/m queue, and the appropriate formulas are found in section 6.7:

\[ \text{Service rate} = \frac{100 \times 10^6}{250 \times 8} = \frac{100 \times 10^6}{2000} = 50,000 \text{ packets per second} \]

\[ \text{Time}_{\text{server}} = \frac{1}{50,000} = 0.00002 \text{ secs} = 20 \mu\text{secs} \]

\[ \text{Utilization} = \frac{\text{Arrival rate}}{\text{Service rate}} = \frac{250,000}{10 \times 50000} = \frac{250,000}{500,000} = 0.5 \]

\[ \text{Time}_{\text{queue}} = \frac{\text{Time}_{\text{server}} \times \text{Server utilization}}{\text{Server utilization}} = 20 \mu\text{secs} \times \frac{0.5}{0.5} = 2 \times 0.5 = 2 \mu\text{secs} \]

\[ \text{Mean response time} = \text{Time}_{\text{queue}} + \text{Time}_{\text{server}} = 2 + 20 = 22 \mu\text{secs} \]

The intuition is clear from the results: the service time is much less for the faster network even though the queuing times are the same. This intuition is the argument for “statistical multiplexing” using packets; queuing times are not worse for a single faster network, and the latency for a single packet is much less. Stated alternatively, you get better latency when you use an unloaded fast network, and data traffic is bursty so it works.

Although connections traditionally align with circuit switching, providing the user with the appearance of a logical connection on top of a packet-switched network is certainly possible. TCP/IP, as we shall see in section 8.8, is a connection-oriented service that operates over packet-switched networks.
Routing: Delivering Messages

Given that the path between nodes may be difficult to navigate depending upon the topology, the system must be able to route the message to the desired node. Shared media has a simple solution: The message is broadcast to all nodes that share the media, and each node looks at an address within the message to see whether the message is for that node. This routing also made it easy to broadcast one message to all nodes by reserving one address for everyone; broadcast is much harder to support in switch-based networks.

Switched media use three solutions for routing. In source-based routing, the message specifies the path to the destination. Since the network merely follows directions, it can be simpler. A second alternative is the virtual circuit, whereby a circuit is established between source and destination, and the message simply names the circuit to follow. ATM uses virtual circuits. The third approach is a destination-based routing, where the message merely contains a destination address, and the switch must pick a path to deliver the message. IP uses destination routing. Hence, ATM switches are simpler conceptually; once a virtual circuit is established, packet switching is very fast. On the other hand, IP routers must decide how to route every packet it receives by doing a routing table lookup on every packet.

Destination-based routing may be deterministic and always follow the same path, or it may be adaptive, allowing the network to pick different routes to avoid failures or congestion. Closely related to adaptive routing is randomized routing, whereby the network will randomly pick between several equally good paths to spread the traffic throughout the network, thereby avoiding hot spots.

Switches in WANs route messages using a store-and-forward policy; each switch waits for the full message to arrive in the switch before it is sent on to the next switch. Generally store-and-forward can retry a message within the network in case of failure. The alternative to store-and-forward, available in some SANs, is for the switch to examine the header, decide where to send the message, and then start transmitting it immediately without waiting for the rest of the message. It requires retransmission from the source on a failure within the network.

This alternative is called either cut-through routing or wormhole routing. In wormhole routing, when the head of the message is blocked, the message stays strung out over the network, potentially blocking other messages. Cut-through routing lets the tail continue when the head is blocked, compressing the strung-out message into a single switch. Clearly, cut-through routing requires a buffer large enough to hold the largest packet, while wormhole routing needs only to buffer the piece of the packet sent between switches.

The advantage of both cut-through and wormhole routing over store-and-forward is that latency reduces from a function of the number of intermediate switches multiplied by the size of the packet to the time for the first part of the packet to negotiate the switches plus the transmission time.
EXAMPLE

The CM-5 supercomputer used wormhole routing, with each switch buffer being just 4 bits per port. Compare efficiency of store-and-forward versus wormhole routing for a 128-node machine using a CM-5 interconnection sending a 16-byte payload. Assume each switch takes 0.25 microseconds and that the transfer rate is 20 MBytes/sec.

ANSWER

The CM-5 interconnection for 128 nodes is hierarchy (see Figure 8.14 on page 595), and a message goes through seven intermediate switches. Each CM-5 packet has four bytes of header information, so the length of this packet is 20 bytes. The time to transfer 20 bytes over one CM-5 link is

\[
\frac{20}{20 \text{ MB/sec}} = 1 \mu\text{sec}
\]

Then the time for store and forward is

\[
(\text{Switches} \times \text{Switch delay}) + ((\text{Switches} + 1) \times \text{Transfer time}) = (7 \times 0.25) + (8 \times 1) = 9.75 \mu\text{secs}
\]

while wormhole routing is

\[
(\text{Switches} \times \text{Switch delay}) + \text{Transfer time} = (7 \times 0.25) + 1 = 2.75 \mu\text{secs}
\]

For this example, wormhole routing improves latency by more than a factor of three.

A final routing issue is the order in which packets arrive. Some networks require that packets arrive in the order sent. The alternative removes this restriction, requiring software to reassemble the packets in proper order.

Congestion Control

One advantage of a circuit-switched network is that once a circuit is established, it ensures there is sufficient bandwidth to deliver all the information sent along that circuit. Moreover, switches along a path can be requested to give specific quality of service guarantees. Thus, interconnection bandwidth is reserved as circuits are established rather than consumed as data are sent, and if the network is full, no more circuits can be established. You may have encountered this blockage when trying to place a long distance phone call on a popular holiday or to a television show, as the telephone system tells you that “all circuits are busy” and asks you to please call back at a later time.

Packet-switched networks generally do not reserve interconnect bandwidth in advance, so the interconnection network can become clogged with too many packets. Just as with rush hour traffic, a traffic jam of packets increases packet latency. Packets take longer to arrive, and in extreme cases fewer packets per second are delivered by the interconnect, just as is the case for the poor rush-hour commuters. There is even the computer equivalent of gridlock: deadlock is
achieved when packets in the interconnect can make no forward progress no mat-

er what sequence of events happens. Chapter 6 addresses how to avoid this ulti-

mate congestion in the context of a multiprocessor.

Higher bandwidth and longer distance networks exacerbate these problems, as 

this example illustrates.

**EXAMPLE**

Assume a 155 Mbits/sec network stretching from San Francisco to New 

York City. How many bytes will be in flight? What is the number if the net-

work is upgraded to 1000 Mbits/sec?

**ANSWER**

Use the prior assumptions and speed of light. The distance between San 

Francisco and New York City is 4120 km. Calculating time of flight:

\[
\text{Time of flight} = \frac{4120 \text{ km}}{\frac{2}{3} \times 300,000 \text{ km/sec}} = 0.0206 \text{ secs}
\]

Let’s assume the network delivers 50% of the peak bandwidth. The num-

ber of bytes in transit on a 155 Mbits/sec network is

\[
\text{Bytes in transit} = \frac{0.5 \times 155 \text{ Mbits/sec}}{8} \times 0.0206 \text{ secs} = 9.7 \text{ MB/sec} \times 0.0206 \text{ secs}
\]

\[
= 0.200\text{MB}
\]

At 1000 Mbits/sec the number is

\[
\text{Bytes in transit} = \frac{0.5 \times 1000 \text{ Mbits/sec}}{8} \times 0.0206 \text{ secs} = 62.5 \text{ MB/sec} \times 0.0206 \text{ secs}
\]

\[
= 1.718\text{MB}
\]

More than a megabyte of messages will be a challenge to control and to 

store in the network.

The solution to congestion is to prevent new packets from entering the net-

work until traffic reduces, just as metering lights guarding on-ramps control the 

rate of cars entering a freeway. There are three basic schemes used for congestion 

control in computer interconnection networks, each with its own weaknesses: 

packet discarding, flow control, and choke packets.

The simplest, and most callous, is *packet discarding*. If a packet arrives at a 

switch and there is no room in the buffer, the packet is discarded. This scheme re-

lies on higher-level software that handles errors in transmission to resend lost 

packets. Internetworking protocols such as UDP discard packets.
The second scheme is to rely on flow control between pairs of receivers and senders. The idea is to use feedback to tell the sender when it is allowed to send the next packet. One version of feedback is via separate wires between adjacent senders and receivers that tell the sender to stop immediately when the receiver cannot accept another message. This backpressure feedback is rapidly sent back to the original sender over dedicated lines, causing all links between the two end points to be frozen until the receiver can make room for the next message. Backpressure flow control is common in supercomputer networks, SANs and even some gigabit Ethernet switches which send fake collision signal to control flow.

A more sophisticated variation of feedback is for the ultimate destination to give the original sender a credit to send \( n \) packets before getting permission to send more. These are generically called credit-based flow control. A window is one version of credit-based flow control. The window’s size determines the minimum frequency of communication from receiver to sender. The goal of the window is to send enough packets to overlap the latency of the interconnection with the overhead to send and receive a packet. The TCP protocol uses a window.

This brings us to a point of confusion on terminology in many papers and textbooks. Note that flow control describes just two nodes of the interconnection and not the total interconnection network between all end systems. Congestion control refers to schemes that reduce traffic when the collective traffic of all nodes is too large for the network to handle. Hence, flow control helps congestion control, but it is not a universal solution.

Choke packets are basis of the third scheme. The observation is that you only want to limit traffic when the network is congested. The idea is for each switch to see how busy it is, entering a warning state when it passes a threshold. Each packet received by the switch in a warning state are sent back to the source via a choke packet that includes the intended destination. The source is expected to reduce traffic to that destination by a fixed percentage. Since it likely will have already sent many packets along that path, it waits for all the packets in transit to be returned before taking choke packets seriously.

8.5 Network Topology

The number of topologies described in publications would be difficult to count, but the number that have been used commercially is just a handful, with designers of parallel supercomputers being the most visible and imaginative. They have used regular topologies to simplify packaging and scalability. The topologies of SANS, LANs and WANs are more haphazard, having more to do with the challenges of long distance or simply the connection of equipment purchased over several years. Topology matters less today than it did in the past. You don’t want to rewrite your application for each new topology, but you would like the system to take advantage of locality that naturally occurs in programs.
Centralized Switch

Figure 8.13 illustrates two of the popular switch organizations, with the path from node $P_0$ to node $P_6$ shown in gray in each topology. A fully connected, or crossbar, interconnection allows any node to communicate with any other node in one pass through the interconnection. Routing depends on the style of addressing. In source-based routing, the message includes a sequence of out-bound arcs to reach a destination. Once an outgoing arc is picked, that portion of the routing

![Diagram of popular switch topologies for eight nodes.](image)

**FIGURE 8.13 Popular switch topologies for eight nodes.** The links are unidirectional; data come in at the left and exit out the right link. The switch box in (c) can pass A to C and B to D or B to C and A to D. The crossbar uses $n^2$ switches, where $n$ is the number of processors, while the Omega network uses $n^2 \log_2 n$ of the large switch boxes, each of which is logically composed of four of the smaller switches. In this case the crossbar uses 64 switches versus 12 switch boxes or 48 switches in the Omega network. The crossbar, however, can simultaneously route any permutation of traffic pattern between processors. The Omega network cannot.
sequence may be dropped from the packet. In destination-based routing, a table decides which port to take for a given address. Some networks will run programs in the switches (“spanning tree protocols”) to generate the routing table on the fly once the network is connected. The Internet does something similar for routing.

An Omega interconnection uses less hardware than the crossbar interconnection \( (n/2 \log_2 n) \) vs. \( n^2 \) switches), but contention is more likely to occur between messages. The amount of contention depends on the pattern of communication. This form of contention is called \textit{blocking}. For example, in the Omega interconnection in Figure 8.13 a message from \( P_1 \) to \( P_7 \) blocks while waiting for a message from \( P_0 \) to \( P_6 \). Of course, if two nodes try to send to the same destination—both \( P_0 \) and \( P_1 \) send to \( P_6 \)—there will be contention for that link, even in the crossbar. Routing in an Omega net can use the same techniques as in a full-crossbar.

A tree is the basis of another switch, with bandwidth added higher in the tree to match the requirements of common communications patterns. Figure 8.14 shows this topology, called a \textit{fat tree}. Interconnections are normally drawn as graphs, with each arc of the graph representing a link of the communication interconnection, with nodes shown as black squares and switches shown as shaded circles.

To double the number of nodes in a fat tree, we just add another level to the top of the tree. Notice that this also increases the bandwidth at the top of the tree, which is an advantage of a fat tree.

This figure shows that there are multiple paths between any two nodes in a fat tree. For example, between node 0 and node 8 there are four paths. Such redundancy can help with fault tolerance. In addition, if messages are randomly assigned to different paths, then this should spread the load throughout the switch and result in fewer congestion problems.

Thus far, the switch is separate from the processor and memory, and assumed to be located in a central location. Looking inside this switch, we see many smaller switches. The term \textit{multistage switch} is sometimes used to refer to centralized units to reflect the multiple steps that a message may travel before it reaches a computer.

**Distributed Switch**

Instead of centralizing these small switching elements, an alternative is to place one small switch at every computer, yielding a distributed switching function.

Given a distributed switch, the question is how to connect the switches together. Figure 8.15 shows that a low-cost alternative to full interconnection is a network that connects a sequence of nodes together. This topology is called a \textit{ring}. Since some nodes are not directly connected, some messages will have to hop along intermediate nodes until they arrive at the final destination. Unlike shared lines, a ring is capable of many simultaneous transfers: the first node can send to the second at the same time as the third node can send to the fourth, for example. Rings
A fat-tree topology for 16 nodes. The shaded circles are switches, and the squares at the bottom are processor-memory nodes. A simple 4-ary tree would only have the links at the front of the figure; that is, the tree with the root labeled 0,0. This three-dimensional view suggests the increase in bandwidth via extra links at each level over a simple tree, so bandwidth between each level of a fat tree is normally constant rather than reduced by a factor of four as in a 4-ary tree. Multiple paths and random routing give it the ability to route common patterns well, which ensures no single pattern from a broad class of communication patterns will do badly. In the CM-5 fat-tree implementation, the switches have four downward connections and two or four upward connections; in this figure the switches have two upward connections.

A ring network topology.
are not quite as good as this sounds because the average message must travel through \( n/2 \) switches, where \( n \) is the number of nodes. To first order, a ring is like a pipelined bus: on the plus side are point-to-point links, and on the minus side are “bus repeater” delays.

One variation of rings used in local area networks is the token ring. To simplify arbitration, a single slot, or token, goes around the ring to determine which node is allowed to send a message. A node can send only when it gets the token. (A token is simply a special bit pattern.) In this section we evaluate the ring as a topology with more bandwidth than a bus, neglecting its advantages in arbitration.

A straightforward but expensive alternative to a ring is to have a dedicated communication link between every element of a distributed switch. The tremendous improvement in performance of fully connected switches is offset by the enormous increase in cost, typically going up with the square of the number of nodes. This cost inspires designers to invent new topologies that are between the cost of rings and the performance of fully connected networks. The evaluation of success depends in large part on the nature of the communication in the interconnection network. Real machines frequently add extra links to these simple topologies to improve performance and reliability. Figure 8.16 illustrates three popular topologies for high performance computers with distributed switches.

One popular measure for interconnections, in addition to the ones covered in section 8.2, is the bisection bandwidth. This measure is calculated by dividing the interconnect into two roughly equal parts, each with half the nodes. You then sum the bandwidth of the lines that cross that imaginary dividing line. For fully connected interconnections the bisection bandwidth is proportional to \((n/2)^2\), where \( n \) is the number of nodes. For a bus, bisection bandwidth is just the speed of one link.

Since some interconnections are not symmetric, the question arises as to where to draw the imaginary line when bisecting the interconnect. Bisection bandwidth is a worst-case metric, so the answer is to choose the division that makes interconnection performance worst. Stated alternatively, calculate bisection bandwidths for all pairs of equal-sized parts, and pick the smallest. Figure 8.17 summarizes these different topologies using bisection bandwidth and the number of links for 64 nodes.

**Example** A common communication pattern in scientific programs is to consider the nodes as elements of a two-dimensional array and then have communication to the nearest neighbor in a given direction. (This pattern is sometimes called NEWS communication, standing for north, east, west, and south, the directions on the compass.) Map an eight-by-eight array onto the 64 nodes in each topology, and assume every link of every interconnect is the same speed. How long does it take each node to send one message to its northern neighbor and one to its eastern neighbor? Ignore nodes that have no northern or eastern neighbors.
In this case, we want to send $2 \times (64 - 8)$, or 112, messages. Here are the cases, again in increasing order of difficulty of explanation:

- **Bus**—The placement of the eight-by-eight array makes no difference for the bus, since all nodes are equally distant. The 112 transfers are done sequentially, taking 112 time units.

- **Fully connected**—Again the nodes are equally distant; all transfers are done in parallel, taking one time unit.

- **Ring**—Here the nodes are differing distances. Assume the first row

![Network topologies that have appeared in commercial supercomputers.](image-url)
of the array is placed on nodes 0 to 7, the second row on nodes 8 to 15, and so on. It takes just one time unit to send to the eastern neighbor, for this is a send from node \(n\) to node \(n + 1\). In this scheme the northern neighbor is exactly eight nodes away, so it takes eight time units for each node to send to its northern neighbor. The ring total is nine time units.

- **2D torus**—There are eight rows and eight columns in our grid of 64 nodes, which is a perfect match to the NEWS communication. It takes just two time units to send to the northern and eastern neighbors.

- **6-cube**—It is possible to place the array so that it will take just two time units for this communication pattern, as in the case of the torus.

This simple analysis of interconnection networks in this section ignores several important practical considerations in the construction of an interconnection network. First, these three-dimensional drawings must be mapped onto chips, boards, and cabinets that are essentially two-dimensional media, often tree-like. For example, due to the fixed height of cabinets, an \(n\)-node Intel Paragon used an \(n/16 \times 16\) rectangular grid rather than the ideal of \(\sqrt{n} \times \sqrt{n}\). Another consideration is the internal speed of the switch: if it is fixed, then more links per switch means lower bandwidth per link, potentially affecting the desirability of different topologies. Yet another consideration is that the latency through a switch depends on the complexity of the routing pattern, which in turn depends on the topology. Finally, the bandwidth from the processor is often the limiting factor: if there is only one port in and out of the processor, then it can only send or receive one message per time unit regardless of the technology.

Topologies that appear elegant when sketched on the blackboard may look awkward when constructed from chips, cables, boards, and boxes. The bottom
line is that quality of implementation matters more than topology. To put these topologies in perspective, Figure 8.18 lists those used in commercial high performance computers.

<table>
<thead>
<tr>
<th>Institution</th>
<th>Name</th>
<th>Number of nodes</th>
<th>Basic topology</th>
<th>Data bits/link</th>
<th>Network clock rate</th>
<th>Peak BW/link (MB/sec)</th>
<th>Bisection (MB/sec)</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thinking Machines</td>
<td>CM-2</td>
<td>1024 to 4096</td>
<td>12-cube</td>
<td>1</td>
<td>7 MHz</td>
<td>1</td>
<td>1024</td>
<td>1987</td>
</tr>
<tr>
<td>Intel</td>
<td>Delta</td>
<td>540</td>
<td>2D grid</td>
<td>16</td>
<td>40 MHz</td>
<td>40</td>
<td>640</td>
<td>1991</td>
</tr>
<tr>
<td>Thinking Machines</td>
<td>CM-5</td>
<td>32 to 2048</td>
<td>Multistage fat tree</td>
<td>4</td>
<td>40 MHz</td>
<td>20</td>
<td>10,240</td>
<td>1991</td>
</tr>
<tr>
<td>Intel</td>
<td>Paragon</td>
<td>4 to 2048</td>
<td>2D grid</td>
<td>16</td>
<td>100 MHz</td>
<td>175</td>
<td>6400</td>
<td>1992</td>
</tr>
<tr>
<td>IBM</td>
<td>SP-2</td>
<td>2 to 512</td>
<td>Multistage fat tree</td>
<td>8</td>
<td>40 MHz</td>
<td>40</td>
<td>20,480</td>
<td>1993</td>
</tr>
<tr>
<td>Cray Research</td>
<td>T3E</td>
<td>16 to 2048</td>
<td>3D torus</td>
<td>16</td>
<td>300 MHz</td>
<td>600</td>
<td>122,000</td>
<td>1997</td>
</tr>
<tr>
<td>Intel</td>
<td>ASCI Red</td>
<td>4536 (x 2 CPUS)</td>
<td>2D Grid</td>
<td></td>
<td></td>
<td>800</td>
<td>51,600</td>
<td>1996</td>
</tr>
<tr>
<td>IBM</td>
<td>ASCI Blue Pacific</td>
<td>1336 (x 4 CPUS)</td>
<td></td>
<td></td>
<td></td>
<td>150</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SGI</td>
<td>ASCI Blue Mountain</td>
<td>1464 (x 2 CPUS)</td>
<td>Fat Hypercube</td>
<td></td>
<td></td>
<td>800</td>
<td>200 x nodes</td>
<td>1998</td>
</tr>
<tr>
<td>IBM</td>
<td>ASCI Blue Horizon</td>
<td>144 (x 8 CPUS)</td>
<td>Multistage Omega</td>
<td></td>
<td></td>
<td>115</td>
<td></td>
<td>1999</td>
</tr>
<tr>
<td>IBM</td>
<td>SP</td>
<td>1 to 512 (x 2 to 16 CPUs)</td>
<td>Multistage Omega</td>
<td></td>
<td></td>
<td>500</td>
<td></td>
<td>2000</td>
</tr>
<tr>
<td>IBM</td>
<td>ASCI White</td>
<td>484 (x 16 CPUS)</td>
<td>Multistage Omega</td>
<td></td>
<td></td>
<td>500</td>
<td></td>
<td>2001</td>
</tr>
</tbody>
</table>

FIGURE 8.18 Characteristics of interconnections of some commercial supercomputers. The bisection bandwidth is for the largest machine. The 2D grid of the Intel Delta is 16 rows by 35 columns and the ASCI Red is 38 rows by 32 columns. The fat-tree topology of the CM-5 is restricted in the lower two levels, hence the lower bandwidth in the bisection. Note that the Cray T3D has two processors per node and the Intel Paragon has from two to four processors per node.

Once again the issues discussed in this section apply at many levels, from inside a chip to a country-sized WAN. The redundancy of a topology matter so that the network can survive despite failures. This is true within a switch as well, so that a single chip failure need not lead to switch failure. It also must be true for a WAN, so that a single backhoe cannot take down the network of a country. The switch then depends on the implementation technology and the demands of the application: it is a multistage network whose topology can be anything from a bus to Omega network.
There are practical issues in addition to the technical issues described so far that are important considerations for some interconnection networks: connectivity, standardization, and fault tolerance.

Connectivity

The number of machines that communication affects the complexity of the network and its protocols. The protocols must target the largest size of the network, and handle the types of anomalous events that occur. Hundreds of machines communicating are a much easier than millions.

Connecting the Network to the Computer

Where the network attaches to the computer affects both the network interface hardware and software. Questions include whether to use the memory bus or the I/O bus, whether to use polling or interrupts, and how to avoid invoking the operating system. The network interface is often the network bottleneck.

Computers have a hierarchy of buses with different cost/performance. For example, a personal computer in 2001 has a memory bus, a PCI bus for fast I/O devices, and an USB bus for slow I/O devices. I/O buses follow open standards and have less stringent electrical requirements. Memory buses, on the other hand, provide higher bandwidth and lower latency than I/O buses. Where to connect the network to the machine depends on the performance goals and whether you hope to buy a standard network interface card or are willing to design or buy one that only works with the memory bus on your model of computer. A few SANs plug into the memory bus, but most SANs and all LANs and WANs plug into the I/O bus.

The location of the network connection significantly affects the software interface to the network as well as the hardware. As mentioned in section 6.6, one key is whether the interface is coherent with the processor’s caches: the sender may have to flush the cache before each send, and the receiver may have to flush its cache before each receive to prevent the stale data problem. Such flushes increase send and receive overhead. A memory bus is more likely to be cache-coherent than an I/O bus and therefore more likely to avoid these extra cache flushes.

A related question of where to connect to the computer is how to connect to the software: Do you use programmed I/O or direct memory access (DMA) to send a message? (See section 6.6.) In general, DMA is the best way to send large
messages. Whether to use DMA to send small messages depends on the efficiency of the interface to the DMA. The DMA interface is usually memory-mapped, and so each interaction is typically at the speed of main memory rather than of a cache access. If DMA setup takes many accesses, each running at uncached memory speeds, then the sender overhead may be so high that it is faster to simply send the data directly to the interface.

Standardization: Cross-Company Interoperability

Standards are useful in many places in computer design, but with interconnection networks they are often critical. Advantages of successful standards include low cost and stability. The customer has many vendors to choose from, which both keeps price close to cost due to competition. It makes the viability of the interconnection independent of the stability of a single company. Components designed for a standard interconnection may also have a larger market, and this higher volume can lower the vendor’s costs, further benefitting the customer. Finally, a standard allows many companies to build products with interfaces to the standard, so the customer does not have to wait for a single company to develop interfaces to all the products the customer might be interested in.

One drawback of standards is the time it takes for committees to agree on the definition of standards, which is a problem when technology is changing quickly. Another problem is when to standardize: on one hand, designers would like to have a standard before anything is built; on the other, it would be better if something is built before standardization to avoid legislating useless features or omitting important ones. When done too early, it is often done entirely by committee, which is like asking all of the chefs in France to prepare a single dish of food; masterpieces are rarely served. Standards can also suppress innovation at that level, since the standard fixes interfaces.

LANs and WANs use standards and interoperate effectively. WANs involve many types of companies and must connect to many brands of computers, so it is difficult to imagine a proprietary WAN ever being successful. The ubiquitous nature of the Ethernet shows the popularity of standards for LANs as well as WANs, and it seems unlikely that many customers would tie the viability of their LAN to the stability of a single company.

Alas, some SANs are standardized yet switches from different companies do not interoperate, and some interoperate as well as LANs and WANs.

Message Failure Tolerance

Although some hardware designers try to build fault free networks, in practice it is only a question of the rate of faults, not whether you can prevent them. Thus, the communication system must have mechanisms for retransmission of a message in case of failure. Often it is handled in higher layers of the software protocol at the end points, requiring retransmission at the source. Given the long time of flight for WANs, often they can retransmit from hop to hop rather relying only on retransmission from the source.
Node Failure Tolerance

The second practical issue refers to whether or not the interconnection relies on all the nodes being operational in order for the interconnection to work properly. Since software failures are generally much more frequent than hardware failures, the question is whether a software crash on a single node can prevent the rest of the nodes from communicating.

Clearly, WANs would be useless if they demanded that thousands of computers spread across a continent be continuously available, and so they all tolerate the failures of individual nodes. LANs connect dozens to hundreds of computers together, and again it would be impractical to require that no computer ever fail. All successful LANs normally survive node failures.

Although most SANs have the ability to work around failed nodes and switches, it is not clear that all communication layer software supports this feature. Typically, low latency schemes sacrifice fault tolerance.

**Example**

Figure 8.19 shows the number of failures of 58 desktop computers on a local area network for a period of just over one year. Suppose that one local area network is based on a network that requires all machines to be operational for the interconnection network to send data; if a node crashes, it cannot accept messages, so the interconnection becomes choked with data waiting to be delivered. An alternative is the traditional local area network, which can operate in the presence of node failures; the interconnection simply discards messages for a node that decides not to accept them. Assuming that you need to have both your workstation and the connecting LAN to get your work done, how much greater are your chances of being prevented from getting your work done using the failure-intolerant LAN versus traditional LANs? Assume the down time for a crash is less than 30 minutes. Calculate using the one-hour intervals from this figure.

**Answer**

Assuming the numbers for Figure 8.19, the percentage of hours that you can’t get your work done using the failure-intolerant network is

\[
\text{Intervals with failures} = \frac{\text{Total intervals} - \text{Intervals no failures}}{\text{Total intervals}}
\]

\[
= \frac{8974 - 8605}{8974} = \frac{369}{8974} = 4.1\%
\]

The percentage of hours that you can’t get your work done using the traditional network is just the time your workstation has crashed. If these failures are equally distributed among workstations, the percentage is

\[
\text{Failures/Machines} = \frac{654/58}{8974} = \frac{11.28}{8974} = 0.13\%
\]
Hence, you are more than 30 times more likely to be prevented from getting your work done with the failure-intolerant LAN than with the traditional LAN, according to the failure statistics in Figure 8.19. Stated alternatively, the person responsible for maintaining the LAN would receive a thirtyfold increase in phone calls from irate users!

<table>
<thead>
<tr>
<th>Failed machines per time interval</th>
<th>One-hour intervals with number of failed machines in first column</th>
<th>Total failures per one-hour interval</th>
<th>One-day intervals with number of failed machines in first column</th>
<th>Total failures per one-day interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8605</td>
<td>0</td>
<td>184</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>264</td>
<td>264</td>
<td>105</td>
<td>105</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>100</td>
<td>35</td>
<td>70</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td>75</td>
<td>11</td>
<td>33</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>40</td>
<td>6</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>35</td>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>18</td>
<td>6</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>28</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>8</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>18</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>11</td>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>1</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>21</td>
<td>1</td>
<td>21</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td>1</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td></td>
<td>1</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td></td>
<td>1</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>8974</strong></td>
<td><strong>654</strong></td>
<td><strong>373</strong></td>
<td><strong>573</strong></td>
</tr>
</tbody>
</table>

FIGURE 8.19 Measurement of reboots of 58 DECstation 5000s running Ultrix over a 373-day period. These reboots are distributed into time intervals of one hour and one day. The first column sorts the intervals according to the number of machines that failed in that interval. The next two columns concern one-hour intervals, and the last two columns concern one-day intervals. The second and fourth columns show the number of intervals for each number of failed machines. The third and fifth columns are just the product of the number of failed machines and the number of intervals. For example, there were 50 occurrences of one-hour intervals with two failed machines, for a total of 100 failed machines, and there were 35 days with two failed machines, for a total of 70 failures. As we would expect, the number of failures per interval changes with the size of the interval. For example, the day with 31 failures might include one hour with 11 failures and one hour with 20 failures. The last row shows the total number of each column: the number of failures doesn’t agree because multiple reboots of the same machine in the same interval do not result in separate entries. (Randy Wang of U.C. Berkeley collected these data.)
One practical issue ties to node failure tolerance: If the interconnection can survive a failure, can it also continue operation while a new node is added to the interconnection? If not, each addition of a new node disables the interconnection network. Disabling is impractical for both WANs and LANs.

Finally, we have been discussing the ability of the network to operate in the presence of failed nodes. Clearly as important to the happiness of the network administrator is the reliability of the network media and switches themselves, for their failure is certain to frustrate much of the user community.

8.7 Examples of Interconnection Networks

To further understand these issues, we look at ten design decisions on the topics we covered so far using examples from LAN, SAN, and WAN:

- What is the target bandwidth?
- What is the message format?
- Which media are used?
- Is the network shared or switched?
- Is it connection-oriented or connectionless?
- Does it use store-and-forward or cut-through routing?
- Is routing use source-based, destination-based, or virtual-circuit based?
- What is used for congestion control?
- What topologies are supported?
- Does it follow a standard?

Ethernet: The Local Area Network

The first example is the Ethernet. It has been extraordinarily successful, with the 10 Mbits/second standard proposed in 1978 used practically everywhere. In 2001, the 100 Mbits/second standard proposed in 1994 is closing in popularity. Many classes of computers include Ethernet as a standard interface. This packet-switched network is connectionless, and it routes using the destination address. Figure 8.20 shows the packet formats for Ethernet, as well as the other two examples. Ethernet is codified as IEEE standard 802.3.

Designed originally for coaxial cable, today Ethernets are primarily Cat5 copper wire, with optical fiber reserved for longer distances and higher bandwidths. There is even a wireless version, which is testimony to its ubiquity.

Computers became thousands of times faster than they were in 1978 and the shared interconnection was no faster for almost 20 years. Hence, past engineers
FIGURE 8.20  Packet format for Infiniband, Ethernet, and ATM. ATM calls their messages “cells” instead of packets, so the proper name is ATM cell format. The width of each drawing is 32 bits. All three formats have destination addressing fields, encoded differently for each situation. All three also have a checksum field to catch transmission errors, although the ATM checksum field is calculated only over the header; ATM relies on higher-level protocols to catch errors in the data. Both Infiniband and Ethernet have a length field, since the packets hold a variable amount of data, with the former counted in 32-bit words and the latter in bytes. Infiniband and ATM headers have a type field (T) that gives the type of packet. The remaining Ethernet fields are a preamble to allow the receiver to recover the clock from the self-clocking code used on the Ethernet, the source address, and a pad field to make sure the smallest packet is 64 bytes (including the header). Infiniband includes a version field for protocol version, a sequence number to allow in-order delivery, a field to select the destination queue, and a partition key field. Infiniband has many more small fields not shown and many other packet formats; above is a simplified view. ATM’s short packet, fixed is a good match to real-time demand of digital voice.
invented temporary solutions until a faster Ethernet was available. One solution was to use multiple Ethernets to connect machines, and to connect these smaller Ethernets with devices that can take traffic from one Ethernet and pass it on to another as needed. These devices allow individual Ethernets to operate in parallel, thereby increasing the aggregate interconnection bandwidth of a collection of computers. In effect these devices provide similar functionality to the switches described above for point-to-point networks.

Figure 8.21 shows the potential parallelism. Depending on how they pass traffic and what kinds of interconnections they can put together, these devices have different names:

- **Bridges**—These devices connect LANs together, passing traffic from one side to another depending on the addresses in the packet. Bridges operate at the Ethernet protocol level and are usually simpler and cheaper than routers, discussed next. Using the notation of the OSI model described in the next section (see Figure 8.25 on page 612), bridges operate at layer 2, the data link layer.

- **Routers** or **gateways**—These devices connect LANs to WANs or WANs to WANs, and resolve incompatible addressing. Generally slower than bridges, they operate at OSI layer 3, the network layer. Routers divide the interconnect
into separate smaller subnets, which simplifies manageability and improves security.

The final network devices are *hubs*, but they merely extend multiple segments into a single LAN. Thus, hubs do not help with performance, as only one message can transmit at a time. Hubs operate at OSI layer 1, the physical layer.

Since these devices were not planned as part of the Ethernet standard, their ad hoc nature has added to the difficulty and cost of maintaining LANs.

In 2001, Ethernet link speed is available at 10, 100, and 1000 Mbits/second, with 10000 Mbits per second likely available in 2002 to 2003. Although 10 and 100 Mbits/sec can share the media with multiple devices, 1000 Mbits/second and above relies on point-to-point links and switches. Ethernet switches normally use cut-through routing.

Due to its age, Ethernet has no real flow control. It originally used carrier sensing with exponential back-off (see page 584) to arbitrate for the shared media. Some switches try to use that interface to retrofit their version of flow control, but flow control is not part of an Ethernet standard.

**Storage Area Network: Infiniband**

A SAN that tries to optimize based on shorter distances is Infiniband. This new standard has clock rates of 2.5 GHz and can transmit data at a peak speed of 2000 Mbits/second per link. These point-to-point links can be bundled together in groups of 4 to 12 to give 4 to 12 times the bandwidth per link. Like Ethernet, it is a packet switched, connectionless network. It also relies only on switches, as does gigabit Ethernet, and also uses cut-through routing and destination-based addressing. The distances are much shorter than Ethernet, with category 5 wire limited to 17 meters and optical fiber limited to 100 meters. It uses backpressure for flow control (see page 592). When going to storage, it relies on the SCSI command set. Although it is not a traditional standard, a trade organization of cooperating companies is responsible for Infiniband.

Given the similarities, why does one need a separate standard for a storage area network versus a local area network? The storage community believes a SAN has different emphasis from a LAN. First, protocol overhead is much lower for a SAN. A gigabit per second LAN can fully occupy a 0.8 to 1.0 GHz CPU when running TCP/IP (see page 653). The Infiniband protocol, on the other hand, places a very light load on the host computer. The reason is a controller on the Infiniband network interface card that offloads the processing from the host computer. Second, protection is much more important in the LAN than the SAN. The SAN is for data only, and is behind the server. From a SAN perspective, the server is like a firewall for the SAN, and hence the SAN is not required to provide protection. Third, storage designers think that graceful behavior under congestion is critical for SANs. The lack of flow control in Ethernet can lead to a lack of grace under pressure. TCP/IP copes with congestion by dropping packets, but storage applications do not appreciate dropped packets.
Not surprisingly, the LAN advocates have a response. First, Ethernet switches are less costly than SAN switches due to greater competition in the marketplace. Second, since Internet Protocol (IP) networks are naturally large, they enable replication of data to geographically diverse sites of the Internet. This geographical advantage both protects against disasters and offers an alternative to tape backup. Thus far, SANs have been relatively small, both in number of nodes and physical distance. Finally, although TCP/IP does have overhead, to try to preserve server utilization, TCP/IP off-loading engines are appearing in the marketplace.

Some LAN advocates are embracing a standard called iSCSI, which exports native SCSI commands over IP networks. The operating system intercepts SCSI commands, and repackages and sends them in a TCP/IP message. At the receiving end, it unpacks messages into SCSI commands and issues them locally. iSCSI allows a company to send SCSI commands and data over its internal WAN or, if transmitted over the Internet, to locations with Internet access.

Wide Area Network: ATM

*Asynchronous Transfer Mode (ATM)* is latest of the ongoing standards set by the telecommunications industry. Although it flirted as competition to Ethernet as a LAN in the 1990s, today ATM has retreated to its WAN stronghold.

The telecommunications standard has scalable bandwidth built in. It starts at 155 Mbits/second, and scales by factors of four to 620 Mbits/second, 2480 Mbits per second, and so on. Since it is a WAN, ATM’s media is fiber, both single mode and multimode. Although it is a switched media, unlike the other examples, it relies on connections for communication. ATM uses virtual channels for routing to multiplex different connections on a single network segment, thereby avoiding the inefficiencies of conventional connection-based networking. The WAN focus also leads to store-and-forward routing. Unlike the other protocols, Figure 8.20 shows ATM has a small, fixed sized packet. (For those curious to the selection of a 48-byte payload, see Section 8.16.) It uses a credit-based flow control scheme (see page 592).

The reason for connections and small packets is quality of service. Since the telecommunications industry is concern about voice traffic, predictability matters as well as bandwidth. Establishing a connection has less variability than connectionless networking, and it simplifies store and forward routing. The small, fixed packet also makes it simpler to have fast routers and switches. Towards that goal, ATM even offers its own protocol stack to compete with TCP/IP. Surprisingly, even though the switches are simple, the ATM suite of protocols is large and complex. The dream was a seamless infrastructure from LAN to WAN, avoiding the hodge-podge of routers common today. That dream has faded from inspiration to nostalgia.
### Summary

Figure 8.22 summarizes answers to the ten questions from the start of this section. It covers three example networks covered here, plus a few other. This section shows how similar technology gets different spins for different concerns of LAN, SAN, and WAN. Nevertheless, the inherent similarity leads to marketplace competition. ATM tried (and failed) to usurp the LAN championship from Ethernet, and in 2001 Ethernet/iSCSI is trying to compete with Fibre Channel Arbitrated Loop (FC-AL) and Infiniband for the SAN markets.
8.8 Internetworking

Undoubtedly one of the most important innovations in the communications community has been internetworking. It allows computers on independent and incompatible networks to communicate reliably and efficiently. Figure 8.23 illustrates the need to cross networks. It shows the networks and machines involved in transferring a file from Stanford University to the University of California at Berkeley, a distance of about 75 km.

**FIGURE 8.23** The connection established between mojave.stanford.edu and mammoth.berkeley.edu. (1995) FDDI is a 100 Mbits/sec LAN, while a T1 line is a 1.5 Mbits/sec telecommunications line and a T3 is a 45 Mbits/sec telecommunications line. BARRNet stands for Bay Area Research Network. Note that inr-111-cs2.Berkeley.edu is a router with two Internet addresses, one for each port.
The low cost of internetworking is remarkable. For example, it is vastly less expensive to send electronic mail than to make a coast-to-coast telephone call and leave a message on an answering machine. This dramatic cost improvement is achieved using the same long-haul communication lines as the telephone call, which makes the improvement even more impressive.

The enabling technologies for internetworking are software standards that allow reliable communication without demanding reliable networks. The underlying principle of these successful standards is that they were composed as a hierarchy of layers, each layer taking responsibility for a portion of the overall communication task. Each computer, network, and switch implements its layer of the standards, relying on the other components to faithfully fulfill their responsibilities. These layered software standards are called protocol families or protocol suites. They enable applications to work with any interconnection without extra work by the application programmer. Figure 8.24 suggests the hierarchical model of communication.

![FIGURE 8.24 The role of internetworking. The width indicates the relative number of items at each level.](image)

The most popular internetworking standard is TCP/IP, which stands for transmission control protocol/internet protocol. This protocol family is the basis of the humbly named Internet, which connects tens of millions of computers around the world. This popularity means TCP/IP is used even when communicating locally across compatible networks; for example, the network file system NFS uses IP even though it is very likely to be communicating across a homogenous LAN such as Ethernet.

We use TCP/IP as our protocol family example; other protocol families follow similar lines. Section 8.16 gives the history of TCP/IP.

The goal of a family of protocols is to simplify the standard by dividing responsibilities hierarchically among layers, with each layer offering services needed by the layer above. The application program is at the top, and at the bottom is the physical communication medium, which sends the bits. Just as abstract data types simplify the programmer’s task by shielding the programmer from details of the implementation of the data type, this layered strategy makes the standard easier to understand.
There were many efforts at network protocols, which led to confusion in terms. Hence, Open Systems Interconnect (OSI) developed a model that popularized describing networks as a series of layers. Figure 8.25 shows the model. Although all protocols do not exactly follow this layering, the nomenclature for the different layers is widely used. Thus, you can hear discussions about a simple layer 3 switch versus a layer 7 smart switch.

<table>
<thead>
<tr>
<th>Layer number</th>
<th>Layer name</th>
<th>Main Function</th>
<th>Example Protocol</th>
<th>Network component</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Application</td>
<td>Used for applications specifically written to run over the network</td>
<td>FTP, DNS, NFS, http</td>
<td>Gateway, smart switch</td>
</tr>
<tr>
<td>6</td>
<td>Presentation</td>
<td>Translates from application to network format, and vice-versa</td>
<td></td>
<td>Gateway</td>
</tr>
<tr>
<td>5</td>
<td>Session</td>
<td>Establishes, maintains and ends sessions across the network</td>
<td>Named pipes, RPC</td>
<td>Gateway</td>
</tr>
<tr>
<td>4</td>
<td>Transport</td>
<td>Additional connection below the session layer</td>
<td>TCP</td>
<td>Gateway</td>
</tr>
<tr>
<td>3</td>
<td>Network</td>
<td>Translates logical network address and names to their physical address (e.g., computer name to MAC address)</td>
<td>IP</td>
<td>Router, ATM switch</td>
</tr>
<tr>
<td>2</td>
<td>Data Link</td>
<td>Turns packets into raw bits and at the receiving end turns bits into packets</td>
<td>Ethernet</td>
<td>Bridge, Network Interface Card</td>
</tr>
<tr>
<td>1</td>
<td>Physical</td>
<td>Transmits raw bit stream over physical cable</td>
<td>IEEE 802</td>
<td>Hub</td>
</tr>
</tbody>
</table>

The key to protocol families is that communication occurs logically at the same level of the protocol in both sender and receiver, but services of the lower level implement it. This style of communication is called peer-to-peer. As an analogy, imagine that General A needs to send a message to General B on the battlefield. General A writes the message, puts it in an envelope addressed to General B, and gives it to a colonel with orders to deliver it. This colonel puts it in an envelope and writes the name of the corresponding colonel who reports to General B, and gives it to a major with instructions for delivery. The major does the same thing and gives it to a captain, who gives it to a lieutenant, who gives it to a sergeant. The sergeant takes the envelope from the lieutenant, puts it into an envelope with the name of a sergeant who is in General B’s division, and finds a private with orders to take the large envelope. The private borrows a motorcycle and
delivers the envelope to the other sergeant. Once it arrives, it is passed up the chain of command, with each person removing an outer envelope with his name on it and passing on the inner envelope to his superior. As far as General B can tell, the note is from another general. Neither general knows who was involved in transmitting the envelope, nor how it was transported from one division to the other.

Protocol families follow this analogy more closely than you might think, as Figure 8.26 shows. The original message includes a header and possibly a trailer sent by the lower-level protocol. The next-lower protocol in turn adds its own header to the message, possibly breaking it up into smaller messages if it is too large for this layer. Reusing our analogy, a long message from the general is divided and placed in several envelopes if it could not fit in one. This division of the message and appending of headers and trailers continues until the message descends to the physical transmission medium. The message is then sent to the destination. Each level of the protocol family on the receiving end will check the message at its level and peel off its headers and trailers, passing it on to the next higher level and putting the pieces back together. This nesting of protocol layers for a specific message is called a protocol stack, reflecting the last-in-first-out nature of the addition and removal of headers and trailers.

As in our analogy, the danger in this layered approach is the considerable latency added to message delivery. Clearly, one way to reduce latency is to reduce the number of layers. But keep in mind that protocol families define a standard, but do not force how the to implement the standard. Just as there are many ways to implement an instruction set architecture, there are many ways to implement a protocol family.

FIGURE 8.26 A generic protocol stack with two layers. Note that communication is peer-to-peer, with headers and trailers for the peer added at each sending layer and removed by each receiving layer. Each layer offers services to the one above to shield it from unnecessary details.
FIGURE 8.27 The headers for IP and TCP. This drawing is 32 bits wide. The standard headers for both are 20 bytes, but both allow the headers to optionally lengthen for rarely transmitted information. Both headers have a length of header field (L) to accommodate the optional fields, as well as source and destination fields. The length field of the whole datagram is in a separate length field in IP, while TCP combines the length of the datagram with the sequence number of the datagram by giving the sequence number in bytes. TCP uses the checksum field to be sure that the datagram is not corrupted, and the sequence number field to be sure the datagrams are assembled into the proper order when they arrive. IP provides checksum error detection only for the header, since TCP has protected the rest of the packet. One optimization is that TCP can send a sequence of datagrams before waiting for permission to send more. The number of datagrams that can be sent without waiting for approval is called the window, and the window field tells how many bytes may be sent beyond the byte being acknowledged by this datagram. TCP will adjust the size of the window depending on the success of the IP layer in sending datagrams: the more reliable and faster it is, the larger TCP makes the window. Since the window slides forward as the data arrives and is acknowledged, this technique is called a sliding window protocol. The piggyback acknowledgment field of TCP is another optimization. Since some applications send data back and forth over the same connection, it seems wasteful to send a datagram containing only an acknowledgment. This piggyback field allows a datagram carrying data to also carry the acknowledgment for a previous transmission, “piggybacking” on top of a data transmission. The urgent pointer field of TCP gives the address within the datagram of an important byte, such as a break character. This pointer allows the application software to skip over data so that the user doesn’t have to wait for all prior data to be processed before seeing a character.
that tells the software to stop. The identifier field and fragment field of IP allow intermediary machines to break the original datagram into many smaller datagrams. A unique identifier is associated with the original datagram and placed in every fragment, with the fragment field saying which piece is which. The time-to-live field allows a datagram to be killed off after going through a maximum number of intermediate switches no matter where it is in the network. Knowing the maximum number of hops that it will take for a datagram to arrive—if it ever arrives—simplifies the protocol software. The protocol field identifies which possible upper layer protocol sent the IP datagram; in our case, it is TCP. The V (for version) and type fields allow different versions of the IP protocol software for the network. Explicit version numbering is included so that software can be upgraded gracefully machine by machine, without shutting down the entire network.

Our protocol stack example is TCP/IP. Let’s assume that the bottom protocol layer is Ethernet. The next level up is the Internet Protocol or IP layer; the official term for an IP packet is datagram. The IP layer routes the datagram to the destination machine, which may involve many intermediate machines or switches. IP makes a best effort to deliver the packets, but does not guarantee delivery, content, or order of datagrams. The TCP layer above IP makes the guarantee of reliable, in-order delivery and prevents corruption of datagrams.

Following the example in Figure 8.26, assume an application program wants to send a message to a machine via an Ethernet. It starts with TCP. The largest number of bytes that can be sent at once is 64 KB. Since the data may be much larger than 64 KB, TCP must divide it into smaller segments and reassemble them in proper order upon arrival. TCP adds a 20-byte header (Figure 8.27) to every datagram, and passes them down to IP. The IP layer above the physical layer adds a 20-byte header, also shown in Figure 8.27. The data sent down from the IP level to the Ethernet is sent in packets with the format shown in Figure 8.20 on page 605. Note that the TCP packet appears inside the data portion of the IP datagram, just as Figure 8.26 suggests.

8.9 Crosscutting Issues for Interconnection Networks

This section describes four topics discussed in other chapters that are fundamental to interconnections.

Density-Optimized Processors versus SPEC-optimized Processors

Given that people all over the world are accessing WWW sites, it doesn’t really matter where your servers are located. Hence, many servers are kept at collocation sites, which charge by network bandwidth reserved and used, and by space occupied and power consumed.

Desktop microprocessors in the past have been designed to be as fast as possible at whatever heat could be dissipated, with little regard to the size of the package and surrounding chips. One microprocessor in 2001 burns 135 watts! Floor space efficiency was also largely ignored. As a result of these priorities, power is a major cost for collocation sites, and density of processors is limited by the power consumed and dissipated.
With portable computers making different demands on power consumption and cooling for processors and disks, the opportunity exists for using this technology to create considerably denser computation. In such a case performance per watt or performance per cubic foot could replace performance per microprocessor as the important figure of merit.

The key is that many applications already work with large clusters (see section 8.10), so it's possible that replacing 64 power hungry processors with, say, 256 efficient processors could be cheaper to run yet be software compatible.

**Smart Switches vs. Smart Interface Cards**

Figure 8.28 shows a trade-off is where intelligence is located in the network. Generally the question is whether to have smarter network interfaces or smarter switches. Making one side smarter generally makes the other side easier and less expensive.

By having an inexpensive interface it was possible for Ethernet to become standard as part of most desktop and server computers. Lower cost switches were made available for people with small configurations, not needing sophisticated routing tables and spanning tree protocols of larger Ethernet switches.

Infiniband is trying a hybrid approach by offering lower cost interface cards for less demanding devices, such as disks, in the hopes that it will be included with some I/O devices. As Infiniband is planned as the successor to PCI bus, computers may come with an Host Channel Adapter built in.

**Protection and User Access to the Network**

A challenge is to ensure safe communication across a network without invoking the operating system in the common case. The Cray Research T3D supercomputer offers an interesting case study. It supports a global address space, so loads and stores can access memory across the network. Protection is ensured because each access is checked by the TLB.

To support transfer of larger objects, a block transfer engine (BLT) was added to the hardware. Protection of access requires invoking the operating system before using the BLT, to check the range of accesses to be sure there will be no protection violations.

Figure 8.29 compares the bandwidth delivered as the size of the object varies for reads and writes. For very large reads, 512 KB, the BLT does achieve the highest performance: 140 MBytes/sec. But simple loads get higher performance for 8 KB or less. For the write case, both achieve a peak of 90 MBytes/sec, presumably because of the limitations of the memory bus. But for writes, BLT can only match the performance of simple stores for transfers of 2 MB; anything smaller and it's faster to send stores. Clearly, a BLT that avoided invoking the operating system in the common case would be more useful.
8.9 Crosscutting Issues for Interconnection Networks

Efficient Interface to Memory Hierarchy versus Interconnection Network

Traditional evaluations of processor performance, such as SPECint and SPECfp, encourage integration of the memory hierarchy with the processor, as the efficiency of the memory hierarchy translates directly into processor performance. Hence, microprocessors have first-level caches on chips along with buffers for writes, and usually have second-level caches on-chip or immediately next to the chip.

Benchmarks such as SPECint and SPECfp do not reward good interfaces to interconnection networks, and hence many machines make the access time to the network delayed by the full memory hierarchy. Writes must lumber their way through full write buffers, and reads must go through the cycles of first- and second-level cache misses before reaching the interconnection. This hierarchy results in newer systems having higher latencies to interconnections than older machines.

Let’s compare three machines from the past. A 40-MHz SPARCstation-2, a 50-MHz SPARCstation-20 without an external cache, and a 50-MHz SPARCstation-20 with an external cache. According to SPECint95, this list is in order of in-
creasing performance. The time to access the I/O bus (S-bus), however, increases in this sequence: 200 ns, 500 ns, and 1000 ns. The SPARCstation-2 is fastest because it has a single bus for memory and I/O, and there is only one level to the cache. The SPARCstation-20 memory access must first go over the memory bus (M-bus) and then to the I/O bus, adding 300 ns. Machines with a second-level cache pay an extra penalty of 500 ns before accessing the I/O bus.

On the other hand, recent computers have dramatically improved memory bandwidth, which is helpful to network bandwidth.

Compute-Optimized Processors versus Receiver Overhead

The overhead to receive a message likely involves an interrupt, which bears the cost of flushing and then restarting the processor pipeline. As mentioned earlier, to read the network status and to receive the data from the network interface likely operates at cache miss speeds. As microprocessors become more superscalar and go to faster clock rates, the number of missed instruction issue opportunities per message reception is likely to rise quickly over time.
8.10 Clusters

...do-it-yourself Beowulf clusters built from commodity hardware and software...has mobilized a community around a standard architecture and tools. Beowulf's economics and sociology are poised to kill off the other architectural lines—and will likely affect traditional supercomputer centers as well.

Gordon Bell and Jim Gray [2001]

Instead of relying on custom machines and custom networks to build massively parallel machines, the introduction of switches as part of LAN technology meant that high network bandwidth and scaling was available from off-the-shelf components. When combined with using desktop computers and disks as the computing and storage devices, a much less expensive computing infrastructure could be created that could tackle very large problems. And by their component nature, clusters are much easier to scale and more easily isolate failures.

There are many mainframe applications—such as databases, file servers, Web servers, simulations, and multiprogramming/batch processing—amenable to running on more loosely coupled machines than the cache-coherent NUMA machines of Chapter 6. These applications often need to be highly available, requiring some form of fault tolerance and repairability. Such applications—plus the similarity of the multiprocessor nodes to desktop computers and the emergence of high-bandwidth, switch-based local area networks—lead to clusters of off-the-shelf, whole computers for large-scale processing.

Performance Challenges of Clusters

One drawback is that clusters are usually connected using the I/O bus of the computer, whereas multiprocessors are usually connected on the memory bus of the computer. The memory bus has higher bandwidth and much lower latency, allowing multiprocessors to drive the network link at higher speed and to have fewer conflicts with I/O traffic on I/O-intensive applications. This connection point also means that clusters generally use software-based communication while multiprocessors use hardware for communication. However, it makes connections non-standard and hence more expensive.

A second weakness is the division of memory: a cluster of \( N \) machines has \( N \) independent memories and \( N \) copies of the operating system, but a shared address multiprocessor allows a single program to use almost all the memory in the computer. Thus, a sequential program in a cluster has \( 1/N \)th the memory available compared to a sequential program in a shared memory multiprocessor. Interest-
ingly, the drop in DRAM prices has made memory costs so low that this multi-
processor advantage is much less important in 2001 than it was in 1995. The
primary issue in 2001 is whether the maximum memory per cluster node is suffi-
cient for the application.

Dependability and Scalability Advantage of Clusters

The weakness of separate memories for program size turns out to be a strength in
system availability and expansibility. Since a cluster consists of independent
computers are connected through a local area network, it is much easier to re-
place a machine without bringing down the system in a cluster than in an shared
memory multiprocessor. Fundamentally, the shared address means that it is diffi-
cult to isolate a processor and replace a processor without significant work by the
operating system and hardware designer. Since the cluster software is a layer that
runs on top of local operating systems running on each computer, it is much easi-
er to disconnect and replace a broken machine.

Given that clusters are constructed from whole computers and independent,
scalable networks, this isolation also makes it easier to expand the system with-
out bringing down the application that runs on top of the cluster. High availability
and rapid, incremental extensibility make clusters attractive to service providers

Pros and Cons of Cost of Clusters

One drawback of clusters has been that the cost of ownership. Administering a
cluster of \( N \) machines is close to the cost of administering \( N \) independent ma-
chines, while the cost of administering a shared address space multiprocessor
with \( N \) processors is close to the cost of administering a single, big machine.

Another difference between the two tends to be the price for equivalent com-
puting power for large-scale machines. Since large-scale multiprocessors have
small volumes, the extra development costs of large machines must be amortized
over few systems, resulting in higher cost to the customer. As we shall see, even
prices for components common to small machines are increased, possibly to re-
cover development. In addition, the manufacturer learning curve (see 573 in the
prior chapter) brings down the price of components used in the high volume PC
market. Since the same switches sold in high volume for small systems can be
composed to construct large networks for large clusters, local area network
switches have the same economy-of-scale advantages as small computers.

Originally, the partitioning of memory into separate modules in each node was
a significant disadvantage to clusters, as division means memory is used less effi-
ciently than on a shared address computer. The incredible drop in price of memo-
ry has mitigated this weakness, dramatically changed the trade-offs in favor of
clusters.
Shooting for the Best of Both Worlds

As is often the case with two competing solutions, each side tries to borrow ideas from the other to become more attractive.

On one side of the battle, to combat the high-availability weakness of multiprocessors, hardware designers and operating system developers are trying to offer the ability to run multiple operating systems on portions of the full machine. The goal is that a node can fail or be upgraded without bringing down the whole machine. For example, the Sun Fire 6800 server has these features (see section 5.15).

On the other side of the battle, since both system administration and memory size limits are approximately linear in the number of independent machines, some are reducing the cluster problems by constructing clusters from small-scale shared memory multiprocessors.

A more radical approach is to keep storage outside of the cluster, possibly over a SAN, so that all computers inside can be treated as clones of one another. As the nodes may cost on the order of a few thousand dollars, it can be cheaper to simply discard a flaky node than spend the labor costs to try hard to repair it. The tasks of the failed node are then handed off to another clone. Clusters are also benefiting from faster SANs and from network interface cards that offer lower-overhead communication.

Popularity of Clusters

Low cost, scaling and fault isolation proved a perfect match to the companies providing services over the Internet since the mid 1990s. Internet applications such as search engines and email servers are amenable to more loosely coupled computers, as the parallelism consists of millions of independent tasks. Hence, companies like Amazon, AOL, Google, Hotmail, Inktomi, WebTV, and Yahoo rely on clusters of PCs or workstations to provide services used by millions of people every day. We delve into Google in section 8.11.

Clusters are growing in popularity in the scientific computing market as well. Figure 8.30 shows the mix of architecture styles between 1993 and 2000 for the top 500 fastest scientific computers. One attraction is that individual scientists can afford to construct clusters themselves, allowing them to dedicate their cluster to their problem. Shared supercomputers are placed on monthly allocation of CPU time, so its plausible for a scientist to get more work done from a private cluster than from a shared supercomputer. It is also relatively easy for the scientist to scale his computing over time as he gets more money for computing.

Clusters are also growing in popularity in the database community. Figure 8.31 plots the cost-performance and the cost-performance per processor of the different architecture styles running the TPC-C benchmark. Note in the top graph
that not only are clusters fastest, they achieve good cost performance. For example, five SMPs with just 6 to 8 processors have worse cost-performance than the 280-processor cluster! Only small SMPs with two to four processors have much better cost performance than clusters. This combination of high performance and cost-effectiveness is rare. Figure 8.32 shows similar results for TPC-H.

The bottom half of Figure 8.31 shows the scalability of clusters for TPC-C. They scale by about a factor of eight in price or processors while maintaining respectable cost performance.

Now that we have covered the pros and cons of clusters and showed their successes in several fields, the next step is to design some clusters.

FIGURE 8.30 Plot of Top 500 supercomputer sites between 1993 and 2000. Note that clusters of various kinds grew from 2% to almost 30% in the last three years, while uniprocessors and SMPs have almost disappeared. In fact, most of the MPPs in the list look are similar to clusters. In 2001, the top 500 collectively has a performance of about 100 Teraflops [Bell 2001]. Performance is measured as speed of running Linpack, which solves a dense system of linear equations. This list at www.top500.org is updated twice a year.
FIGURE 8.31 Performance, Cost, and Cost-Performance per Processor for TPC-C. Not only do clusters have the highest tpmC rating, they have better cost performance ($/tpmC) for any SMP with a total cost over $1M. The bottom graph shows that clusters get high performance by scaling. They can sustain 40 to 50 transactions per minute per $1000 of cost from 32 to 280 processors. Figure 8.40 on page 636 describes the leftmost cluster, and Figure 8.41 on page 637 shows the cost model of TPC-C in more detail. These plots are for all computers that have run version 5 of the TPC-C benchmark as of August 2001.
8.11 Designing a Cluster

To take the discussion of clusters from the abstract to the concrete, this section goes through four examples of cluster design. Like section 7.11 in the prior chapter, the examples evolve in realism. The examples of the last chapter which examined performance and availability apply to clusters as well. Instead, we show cost trade-offs, a topic rarely found in computer architecture.

In each case we are designing a system with about 32 processors, 32 GB of DRAM, and 32 or 64 disks. Figure 8.33 lists the components we use to construct the cluster, including their prices.

Before starting the examples, Figure 8.33 confirms some of the philosophical points of the prior section. Note that difference in cost and speed processor is in the smaller systems versus the larger multiprocessor. In addition, the price per DRAM DIMM goes up with the size of the computers.

Regarding the processors, the server chip includes a much larger L2 cache, increasing from 0.25 MB to 1 MB. Due to its much larger die size, the price of 1-MB-cache chip is more than double the 0.25-MB-cache. The purpose of the larger L2 cache is to reduce memory bandwidth to allow eight processors to share a memory system. Not only are these large caches chips much more expensive, its...
### FIGURE 8.33 Prices of options for three rack-mounted servers from IBM and 1-Gbit Ethernet switches from Emulex in August 2001.

Note the higher price for processors and DRAM DIMMs with larger computers. The base price of these computers includes 256 MB of DRAM (512 MB for 8-way server), two slots for disks, an UltraSCSI 160 adapter, two 100 Mbit Ethernets, a CD-ROM drive, a floppy drive, six to eight fans, and SVGA graphics. The power supply for the Emulex switches is 200 watts and is 500 watts for the EXP300. n the xSeries 370 you must add an accelerator costing $1249 to go over 4 CPUs.
has also been hard for Intel to achieve the similar clock rates to the small-cache chips: 700 MHz vs. 1000 MHz in August 2001.

The higher price of the DRAM is harder too explain based on cost. For example, all include ECC. The uniprocessor uses 133 MHz SDRAM and the 2-way and 8-way both use registered DIMM modules (RDIMM) SDRAM. There might a slightly higher cost for the buffered DRAM between the uniprocessor and 2-way boxes, but it is hard to explain increasing price 1.5 times for the 8-way SMP vs. the 2-way SMP. In fact, the 8-way SDRAM operates at just 100 MHz. Presumably, customers willing to pay a premium for processors for an 8-way SMP are also willing to pay more for memory.

Reasons for higher price matters little to the designer of a cluster. The task is to minimize cost for a given performance target. To motivate this section, here is an overview of the four examples:

1. **Cost of Cluster Hardware Alternatives with Local Disk**: The first example compares the cost of building from a uniprocessor, a 2-way SMP, and an 8-way SMP. In this example, the disks are directly attached to the computers in the cluster.

2. **Cost of Cluster Hardware Alternatives with Disks over SAN**: The second example moves the disk storage behind a RAID controller on a Storage Area Network.

3. **Cost of Cluster Options that is more realistic**: The third example includes the cost of software, the cost of space, some maintenance costs, and operator costs.

4. **Cost and Performance of a Cluster for Transaction Processing**: This final example describes a similar cluster tailored by IBM to run the TPC-C benchmark. (It is one of the cluster results in Figure 8.31.) This example has more memory and many more disks to achieve a high TPC-C result, and at the time of this writing, it the 13th fastest TPC-C system. In fact, the machine with the fastest TPC-C is just a replicated version of this cluster with a bigger LAN switch. This section highlights the differences between this database-oriented cluster and the prior examples.

**First Example: Cost of Cluster Hardware Alternatives with Local Disk**

This first example looks only at hardware cost of the three alternatives using the IBM pricing information. We’ll look at the cost of software and space later.

---

**Example**

Using the information in Figure 8.33, compare the cost of the hardware for three clusters built from the three options in the figure. In addition, calculate the rack space. The goal for this example is to construct a cluster with 32 processors, 32 GB of memory protected by ECC, and more than 2 TB of disk. Connect the clusters with 1 gigabit, switched Ethernet.
Figure 8.34 shows the logical organization of the three clusters.

Let's start with the 1-processor option (IBM xSeries model 300). First, we need 32 processors and thus 32 computers. The maximum memory for this computer is 1.5 GB, allowing $1GB \times 32 = 32$ GB. Each computer can hold two disks and the largest disk available in the model 300 is 36.4 GB, yielding $32 \times 2 \times 36.4$ GB or 2330 GB. Using the built-in slots for storage is the least expensive solution, so we'll take this option. Each computer needs its own Gbit Host Adapter, but 32 cables are more than a 30-port switch can handle. Thus, we use two Emulex cLAN5300 switches. We connect the two switches together with four cables, leaving plenty for of ports for the 32 computers.

A standard VME rack is 19 inches wide and about 6 feet tall, with a typical depth of 30 inches. This size is so popular that it has its own units: 1 VME rack unit (RU) is about 1.75 inches high, so a rack can hold objects up to 44 RU. The 32 uniprocessor computers each use 1 rack unit of space, plus 2 rack units for each switch, for a total of 36 rack units. This fits snugly in one standard rack.

For the 2-processor case (model 330), everything is halved. The 32 processors need only 16 computers. The maximum memory is 4 GB, but we need just 2 GB per computer to hit our target of 32 GB total. This model allows 73.4 GB disks, so we need only $16 \times 2 \times 73.4$ GB to reach 2.3 TB, and these disks fit in the slots in the computers. A single 30-port switch has more ports than we need. The total space demand is 18 rack units ($16 \times 1 + 1 \times 2$), or less than half a standard rack.

The 8-processor case (model 370) needs only 4 computers to hold 32 processors. The maximum memory is 32 GB, but we need just 8 GB per computer to reach our target. Since there are only 4 computers, the 8-port switch is fine. The shortcoming is in disks. At 2 disks per computer, these 4 computers can hold at most 8 disks, and the maximum capacity per disk is still 73.4 GB. The solution is to add a storage expansion box (EXP300) to each computer, which can hold up to 14 disks. This solution requires adding an external UltraSCSI controller to each computer as well. The rack space is 8 RU for the computer, 3 RU for the disk enclosure, and 1 RU for the switch. Alas, the total is $4 \times (8 + 3) + 1$ or 45 rack units, which just misses the maximum of a standard rack. Hence, this option occupies two racks.

Figure 8.35 shows the total cost of each option. This example shows some issues for clusters:

* **Expansibility incurs high prices.** For example, just 4 of the base 8-way SMPs--each with just one processor and 0.5 GB of DRAM--costs more than 32 uniprocessor computers, each with 1 processor and 0.25 GB of DRAM. The only hope of cost competitiveness is to occupy all the options of a large SMP.
FIGURE 8.34 Three cluster organizations based on uniprocessors (top), 2-way SMPs (middle), and 8-way SMPs (bottom). P stands for processor, M for memory (1, 2, and 8 GB), and D for disk (36.4, 73.4, 73.4 GB).
8.11  Designing a Cluster

* Network vs. local bus trade-off. Figure 8.35 shows how the larger SMPs need less to spend less on networking, as the memory buses carry more of the communication workload.

The uniprocessor cluster costs 1.1 times the 2-way SMP option, and the 8-way SMP cluster cost 1.6 times the 2-way SMP. The 2-way SMP wins the cost competition because the components are relatively cost-effective and it needs fewer systems and network components.

FIGURE 8.35  Price of three clusters with a total of 32 processors, 32 GB memory, and 2.3 TB disk. Note the reduction in network costs as the size of the SMP increases, since the memory buses supply more of the interprocessor communication. Rack prices are included in the total price, but are too small to show in the bars. They account for $1725 in the first two cases and $3450 in the third case.

Second Example: Using a SAN for disks.

The previous example uses disks local to the computer. Although this can reduce costs and space, the problem for the operator is that 1) there is no protection
against a single disk failure, and 2) there is state in each computer that must be
managed separately. Hence, the system is down on a disk failures until the opera-
tor arrives, and there is no separate visibility or access to storage.

This second example centralizes the disks behind a RAID controller in each
case using FC-AL as the Storage Area Network. To keep comparisons fair, we
continue use of IBM components. Figure 8.36 lists the costs of the components in
this option. Note that this IBM RAID controller requires FC-AL disks.

**EXAMPLE** Using the information in Figure 8.36, calculate the cost of the hardware
for three clusters above but now use the SAN and RAID controller.

**ANSWER** The change from the clusters in the first example is that we remove all in-
ternal SCSI disks and replace them with FC-AL disks behind the RAID
storage server. To connect to the RAID box, we add a FC-AL host bus
adapter per computer to the uniprocessor and 2-way SMP clusters and
replace the SCSI host bus adapter in the 8-way SMP cluster.

FC-AL can be connected in a loop with up to 127 devices, so there is
no problem in connecting the computers to the RAID box. The RAID box
has a separate FC-AL loop for the disks. It has room for 10 FC-AL disks,
so we need three EXP500 enclosures for the remaining 22 FC-AL disks.
(The FC-AL disks are half-height, which are taller than the low profile
SCSI disks, so we can fit only 10 FC-AL disks per enclosure.) We just
need to add cables for each segment of the loop.

Since the RAID box needs 3 rack units as do each of the 3 enclo-
ures, we need 12 additional rack units of space. This adds a second rack
to the uniprocessor cluster, but there is sufficient space in the racks of the
other clusters. If we use RAID-5 and have a parity group size of 8 disks,
we still have 28 disks of data or 28 x 73.4 or 2.05 TB of user data, which
is sufficient for our goals.

Figure 8.37 shows the hardware costs of this solution. Since there

---

**FIGURE 8.36 Components for Storage Area Network cluster.**

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM FC-AL High Availability RAID storage server</td>
<td>$15,999</td>
</tr>
<tr>
<td>IBM 73.4 GB 10K RPM FC-AL disk</td>
<td>$1,699</td>
</tr>
<tr>
<td>IBM EXP500 FC-AL storage enclosure (up to 10 disks)</td>
<td>$3,815</td>
</tr>
<tr>
<td>FC-AL 10-meter cables</td>
<td>$100</td>
</tr>
<tr>
<td>IBM PCI FC-AL Host Bus adaptor</td>
<td>$1,485</td>
</tr>
<tr>
<td>IBM FC-AL RAID server rack space (VME rack units)</td>
<td>3</td>
</tr>
<tr>
<td>IBM EXP500 FC-AL rack space (VME rack units)</td>
<td>3</td>
</tr>
</tbody>
</table>
must be one FC-AL host bus adapter per computer, they cost enough to bring the prices of the uniprocessor and 8-way SMP clusters to parity. The 2-way SMP is still substantially cheaper. Notice that again the cost of both the LAN network and the SAN network decrease as the number of computers in the cluster decrease.

The SAN adds about $40,000 to $100,000 to the price of the hardware for the clusters. We’ll see in the next example whether we justify such costs.

FIGURE 8.37 Prices for hardware for three clusters using SAN for storage. As in Figure 8.35, the cost of the SAN network also shrinks as the servers increase in number of processors per computer. They share the FC-AL host bus adapters and also have fewer cables. Rack prices are too small to see in the columns, but they account for $3450, $1725, and $3450, respectively.
Third Example: Accounting for Other Costs

The first and second examples only calculated the cost of the hardware (which is what you might expect from book on computer architecture). There are two other obvious costs not included: software and the cost of a maintenance agreement for the hardware. Figure 8.38 lists the costs covered in this example.

Notice that Microsoft quadruples the price when the operating system runs on a computer with 5 to 8 processors versus a computer with 1 to 4 processors. Moreover, the database cost is primarily a linear function of the number of processors. Once again, software pricing appears to be based on value to the customer versus cost of development.

Another significant cost is the cost of the operators to keep the machine running, upgrade software, perform backup and restore, and so on. In 2001, the cost (including overhead) is about $100,000 per year for an operator.

In addition to labor costs, backup uses up tapes to act as the long-term storage for system. A typical backup policy is daily incremental dumps and weekly full dumps. A common practice is to save four weekly tapes and then one full dump per month for the last six months. The total is 10 full dumps, plus a week of incremental dumps.

There are other costs, however. One is the cost of the space to house the server. Thus, collocation sites have been created to provide virtual machine rooms for companies. They provide scalable space, power, cooling, and network bandwidth plus provide physical security. They make money by charging rent for space, for network bandwidth, and for optional services from on-site administrators.

| Software: Windows 2000 1-4 CPUs + IBM Director | $799 |
| Software: Windows 2000 1-8 CPUs + IBM Director | $3,295 |
| Software: SQL Server Database (per processor!) | $16,541 |
| 3-year HW maintenance: LAN switches + HBA       | $45,000 |
| 3-year HW maintenance: IBM xSeries computers  | 7.5% |
| Rack space rental (monthly per rack)           | $800 to $1200 |
| Extra 20 amp circuit per rack (monthly)        | $200 to $400 |
| Bandwidth charges per megabit (monthly)        | $500 to $2000 |
| Operator costs (yearly)                        | $100,000 |
| DLT tapes (40 GB raw, 80 GB compressed)        | $70 |

FIGURE 8.38 Components for Storage Area Network cluster in 2001. Notice the higher cost of the operating system in the larger server. (Redhat Linix 7.1, however, is $49 for all three.)
Collocation rates are negotiated and much cheaper per unit as space requirements increase. A rough guideline in 2001 is that rack space, which includes one 20-amp circuit, costs $800 to $1200 per month. It drops by 20% if you use more than 75 to 100 racks. Each additional 20 amp circuit per rack costs another $200 to $400 per month. Although we are not calculating these costs in this case, they also charge for network bandwidth: $1500 to $2000 per Mbits/sec per month, if your continuous use is just 1-10 Mbits/second, drops to $500 to $750 per Mbits/sec per month, if your continuous use measures 1 - 2 Gbits/second.

Pacific Gas and Electric in Silicon Valley limits a single building to have no more than 12 megawatts of power and the typical size of a building is no more than 100,000 square feet. Thus, a guideline is that collocation sites are designed assuming no more than 100 watts per square foot. If you include the space for people to get access to a rack to repair and replace components, a rack needs about 10 square feet. Thus, collocation sites expect at most 1000 watts per rack.

**Example**
Using the information in Figure 8.38, calculate the total cost of ownership for three years: purchase prices, operator costs, and maintenance costs.

**Answer**
Figure 8.39 shows the total cost of ownership for the six clusters.

To keep things simple, we assume each system with local disks needs a full-time operator, but the clusters that access their disks over a SAN with RAID need only a half-time operator. Thus, operator cost is 3 x $100,000 = $300,000 or 3 x $50,000 = $150,000.

For backup, let’s assume we need enough tapes to store 2 TB for a full dump. We need four sets for the weekly dumps plus six more sets so that we can have a six-month archive. Tape units normally compress their data to get a factor of two in density, so we’ll assume compression successfully turns 40 GB drives into 80 GB drives. The cost of these tapes is:

\[10 \times \frac{2000GB}{tape} \times $70 = 10 \times 25 \times 70 = 17,500\]

The daily backups depend on the amount of data changed. If 2 tapes per day are sufficient (up to 8% changes per day), we need to spend another

\[7 \times 2 \times 70 = 14 \times 70 = 980\]

The figure lists maintenance costs for the computers and the LAN. The disks come with a 3-year warranty, so there is no extra maintenance cost for them.

The cost per rack of rental space for three years is 3 x 12 x $1000 or $36,000.

Figure 8.39 shows the 2-way SMP using SAN is the winner. Note that hardware costs are only a half to a third of the cost of ownership. Over
three years the operator costs can be more than the cost of purchase of the hardware, so reducing those costs significantly reduces total cost of ownership.

Our results depend on some critical assumptions, but surveys of the total cost of ownership for items with storage go up to factors five to ten over purchase price.
Fourth Example: Cost and Performance of a Cluster for Transaction Processing

The August 2001 TPC-C report includes a cluster built from similar building blocks to the examples above. This cluster also has 32 processors, uses the same IBM computers as building blocks, and it uses the same switch to connect computers together. Figure 8.40 shows its organization. It achieves 121,319 queries for hour for $2.2M.

Here are the key differences:

- **Disk size:** since TPC-C cares more about I/Os per second (IOPS) than disk capacity, this cluster uses many small fast disks. The use of small disks gives many more IOPS for the same capacity. These disks also rotate at 15000 RPM vs. 10000 RPM, delivering more IOPS per disk. The 9.1-GB disk costs $405 and the 18.2-GB disk costs $549, or an increase in dollars per GB of factor of 1.7 to 2.5. The totals are 560 9.1-GB disks and 160 18.2-GB disks, yielding a total capacity of 8 TB. (Presumably the reason for the mix of sizes is get sufficient capacity and IOPS to run the benchmark.) These 720 disks need $14 or 52 enclosures, which is 13 enclosures per computer. In contrast, earlier 8-way clusters achieved 2 TB with 32 disks, as we cared more about cost per GB than IOPS.

- **RAID:** Since the TPC-C benchmark does not factor in human costs for running a system, there is little incentive to use a SAN. TPC-C does require a RAID protection of disks, however. IBM used a RAID product that plugs into a PCI card and provides four SCSI strings. To get higher availability and performance, each enclosure attaches to two SCSI buses. Thus, there are 52 x 2 or 104 SCSI cables attached to the 28 RAID controllers which support up to 28 x 4 or 106 strings.

- **Memory:** Conventional wisdom for TPC-C is to pack as much DRAM as possible into the servers. Hence, each of the four 8-way SMPs is stuffed with the maximum of 32 GB, yielding a total of 128 GB.

- **Processor:** This benchmark uses 900 MHz Pentium III with a 2MB L2 cache. The price is $6599 as compared to prior 8-way clusters for $1799 for the 700 MHz Pentium III with a 1 MB L2 cache.

- **PCI slots:** This cluster uses 7 of the 12 available PCI bus slots for the RAID controllers compared to 1 PCI bus slot for an external SCSI or FC-AL controller in the prior 8-way clusters. This greater utilization follows the guideline of trying to use all resources of a large SMP.

- **Tape Reader, Monitor, Uninterruptable Power Supply:** To make the system easier to come up and to keep running for the benchmark, IBM includes one DLT tape reader, four monitors, and four UPSs.
FIGURE 8.40 IBM Cluster for TPC-C. This cluster has 32 Pentium III processors, each running at 900 MHz with a 2MB L2 cache. The total of DRAM memory is 128 GB. Seven PCI slots in each computer contain RAID controllers (R for RAID), and each has four Ultra160 SCSI strings. These strings connect to 13 storage enclosures per computer, giving 52 total. Each enclosure has 14 SCSI disks, either 9.1 GB or 18.2 GB. The total is 560 9.1 GB disk and 140 18.2 GB disks. There are also two 9.1 GB disks inside each computer that are used for paging and rebooting.
8.11 Designing a Cluster

Maintenance and spares: TPC-C allows use of spares to reduce maintenance costs, which is a minimum of two spares or 10% of the items. Hence, there are two spare Ethernet switches, host adapters, and cables for TPC-C.

Figure 8.41 compares the 8-way cluster from before to this TPC-C cluster. Note that almost half of the cost is in software, installation, and maintenance for the TPC-C cluster. At the time of this writing, the computer with the fastest TPC-C result basically scales this cluster from 4 to 35 xSeries 370 servers and uses bigger Ethernet switches.

<table>
<thead>
<tr>
<th></th>
<th>8-way SAN Cluster</th>
<th>TPC-C Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Systems (700 MHz/1MB v. 900 MHz/2MB)</td>
<td>$58 17%</td>
<td>$76 3%</td>
</tr>
<tr>
<td>28 Extra processors (700 MHz/1MB v. 900 MHz/2MB)</td>
<td>$55 16%</td>
<td>$190 8%</td>
</tr>
<tr>
<td>Extra memory (8 GB v. 32 GB)</td>
<td>$71 20%</td>
<td>$306 14%</td>
</tr>
<tr>
<td>Disk drives (2TB/73.4GB v. 8TB/9.1,18.2 GB)</td>
<td>$54 15%</td>
<td>$316 14%</td>
</tr>
<tr>
<td>Disk enclosures (3 v. 52)</td>
<td>$11 3%</td>
<td>$165 7%</td>
</tr>
<tr>
<td>RAID controller (1 v. 28)</td>
<td>$16 4%</td>
<td>$69 3%</td>
</tr>
<tr>
<td>LAN network (1 switch/4 HBAs v. 3 switches/6 HBAs)</td>
<td>$10 3%</td>
<td>$24 1%</td>
</tr>
<tr>
<td>SAN network (4 NICs, cables v. 0)</td>
<td>$10 3%</td>
<td>n.a. 0%</td>
</tr>
<tr>
<td>Software (Windows v. Windows + SQL server + installation)</td>
<td>$13 4%</td>
<td>$951 42%</td>
</tr>
<tr>
<td>Maintenance + hardware setup costs</td>
<td>$51 14%</td>
<td>$115 5%</td>
</tr>
<tr>
<td>Racks, UPS, backup (2 racks vs. 7 racks + 4 UPS +1 tape unit)</td>
<td>$3 1%</td>
<td>$40 2%</td>
</tr>
<tr>
<td>Total</td>
<td>$352 100%</td>
<td>$2,252 100%</td>
</tr>
</tbody>
</table>

FIGURE 8.41 Comparing 8-way SAN cluster and TPC-C cluster in price (in $1000) and percentage. The higher cost of the system and extra processors is due to using the faster chips with the larger caches. Memory costs are higher due to more total memory and using the more expensive 1 GB DIMMs. The increased disk costs and disk enclosure costs are due to higher capacity and using smaller drives. Software costs increase due to adding SQL server database plus IBM charges for software installation of this cluster. Similarly, although hardware maintenance costs are close, IBM charged to setup seven racks of hardware, whereas we assumed the customer assembled two racks of hardware “for free.” Finally, SAN costs are higher due to TPC-C policy of buying spares to lower maintenance costs.

Summary of Examples

With completion of the cluster tour, you’ve seen a variety of cluster designs, including one representative of the state-of-the-art cost-performance cluster in 2001. Note that we concentrated on cost in constructing these clusters, but only book length prevents us from evaluating the performance and availability bottlenecks in these designs. Given the similarity to performance analysis of storage systems in the last chapter, we leave that to the reader in the exercises.
Having completed the tour of cluster examples, a few things standout. First, the cost of purchase is less than half the cost of ownership. Thus, inventions that only help with hardware costs can solve only a part of the problem. For example, despite the higher costs of SAN, they may lower cost of ownership sufficiently to justify the investment. Second, the smaller computers are generally cheaper and faster for a given function compared to the larger computers. In this case, for the larger cache required to allow several processors to share a bus means a much larger die, which increases cost and limits clock rate. Third, space and power matter for both ends of the computing spectrum: clusters at the high end and embedded computers at the low end.

### 8.12 Putting It All Together: The Google Cluster of PCs

Figure 8.42 shows the rapid growth of the World Wide Web and the corresponding demand for searching it. The number of pages indexed grew by a factor of 1000 between 1994 and 1997, but people were still only interested in the top 10 answers, which was a problem for search engines. In 1997, only one quarter of the search engines would find themselves in their top 10 queries.

<table>
<thead>
<tr>
<th>Date</th>
<th>WWW pages indexed (Million)</th>
<th>Queries per day (Million)</th>
<th>Search Engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 1994</td>
<td>0.11</td>
<td>0.0015</td>
<td>World Wide Web Worm</td>
</tr>
<tr>
<td>November 1997</td>
<td>100</td>
<td>20</td>
<td>Alta Vista</td>
</tr>
<tr>
<td>December 2000</td>
<td>1327</td>
<td>70</td>
<td>Google</td>
</tr>
</tbody>
</table>

FIGURE 8.42 Growth in pages indexed and search queries performed by several search engines. [Brin and Page, 1998] Searches have been growing about 20% per month at Google, or about 8.9 times per year. Most of the 1.3 billion pages are fully indexed and cached at Google. Google also indexes pages based only on the URLs in cached and indexed pages, so about 40% of the 1.3 billion are just URLs without cached copies of the page at Google.

Searched the web for Hennessy Patterson. Results 1 - 10 of about 13,300. Search took 0.23 seconds.

**Computer Architecture: A Quantitative Approach**

... on currently predominant and emerging commercial systems, the Hennessy and Patterson have prepared entirely new chapters covering additional advanced topics: ...

www.mkp.com/books_catalog/1-55860-329-8.asp - 13k - [Cached](#) - [Similar pages](#)
Google was designed first to be a search engine that could scale at that growth rate. In addition to keeping up with the demand, Google improved the relevance of the top queries produced so that user would likely get what the desired result. For example, Figure 8.43 shows the first Google result for the query “Hennessy Patterson,” which from your authors’ perspective is the right answer. Techniques to improve search relevance include ranking pages by popularity, examining the text at the anchor sites of the URLs, and proximity of keyword text within a page.

Search engines also have a major reliability requirement, as people are using it at all times of the day and from all over the world. Google must essentially be continuously available.

Since a search engine is normally interacting with a person, its latency must not exceed its users’ patience. Google’s goal is that no search takes more than 0.5 seconds, including network delays.

As the figures above show, bandwidth is also vital. In 2000, Google served an average of almost 1000 queries per second as well as searched and indexed more than a billion pages.

In addition, a search engine must crawl the WWW regularly to have up-to-date information to search. Google crawls the entire WWW and updates its index every 4 weeks, so that every WWW page is visited once a month. Google also keeps a local copy of the text of most pages so that it can provide the snippet text as well as offer a cached copy of the page, as shown in Figure 8.43.

**Description of the Google Infrastructure**

To keep up with such demand, in December 2000 Google uses more than 6000 processors and 12000 disks, giving Google a total of about one petabyte of disk storage. At the time, the Google site was likely the single system with the largest storage capacity in the private sector.

Rather than achieving availability by using RAID storage, Google relies on redundant sites each with thousands of disks and processors: two sites are in Silicon Valley and one in Virginia. The search index, which is a small number of terabytes, plus the repository of cached pages, which is on the order of the same size, are replicated across the three sites. Thus, if a single site fails, there are still two more that can retain the service. In addition, the index and repository are replicated within a site to help share the workload as well as to continue to provide service within a site even if components fail.

Each site is connected to the Internet via OC48 (2488 Mbits/sec) links of the collocation site. To provide against failure of the collocation link, there is a separate OC12 link connecting the two Silicon Valley sites so that in an emergency both sites can use the Internet link at one site. The external link is unlikely to fail at both sites since different network providers supply the OC48 lines. (The Virginia site now has a sister site to provide so as to provide the same benefits.)

Figure 8.44 shows the floor plan of a typical site. The OC48 link connects to two Foundry BigIron 8000 switches via a large Cisco 12000 switch. Note that
this link is also connected to the rest of the servers in the site. These two switches are redundant so that a switch failure does not disconnect the site. There is also an OC12 link from the Foundry switches to the sister site for emergencies. Each switch can connect to 128 1-Gbit/sec Ethernet lines and each rack has 2 1-Gbit Ethernet lines per switch, so the maximum number of racks for the site is 64. The two racks near the Foundry switches contain a few PCs to act as front ends and help with tasks such as html service, load balancing, monitoring, and UPS to keep the switch and fronts up in case of a short power failure. It would seem that a facility that has redundant diesel engines to provide independent power for the whole site would make UPS redundant. A survey of data center users suggests power failures still happen yearly.

Figure 8.45 shows Google’s rack of PCs. Google uses PCs that are only 1 VME rack unit. To connect these PCs to the Foundry switches, it uses an HP Ethernet switch. It is 4 RU high, leaving room in the rack for 40 PCs. This switch has modular network interfaces, which are organized as removable blades. Each blade can contain 8 100-Mbits/sec Ethernet interfaces or a single 1-Gbit Ethernet interface. Thus, 5 blades are used to connect 100 Mbits/sec Cat5 cables to each of the 40 PCs in the rack, and 2 blades are used to connect 1-Gbit/sec copper cables to the two Foundry switches.

As Figure 8.45 shows, to pack even more PCs in a rack Google uses the same configuration in the front and back of the rack, yielding 80 PCs and 2 switches per rack. There is about a 3-inch gap in the middle between the columns of PCs for the hot air to exit, which is drawn out of the “chimney” via exhaust fans at the top of the rack.

**FIGURE 8.44** Floor plan of a Google cluster, from a God’s eye view. There are 40 racks, each connected via 4 copper Gbit Ethernet links to 2 redundant Foundry 128 by 128 switches (Fnd switch). Figure 8.45 shows a rack contains 80 PCs, so this facility has about 3200 PCs. (For clarity, the links are only shown for the top and bottom rack in each row.) These racks are on a raised floor so that the cables can be hidden and protected. Each Foundry switch in turn is connected to the collocation site network via an OC48 (2.4 Gbit) to the Internet. There are two Foundry switches so that the cluster is still connected even if one switch fails. There is also a separate OC12 (622 Mbit) link to a separate nearby collocation site in case the OC48 network of one collocation site fails; it can still serve traffic over the OC12 to the other sites network. Each Foundry switch can handle 128 1-Gbit Ethernet lines and each rack has 2 1-Gbit Ethernet lines per switch, so the maximum number of racks for the site is 64. The two racks near the Foundry switches contain a few PCs to act as front ends and help with tasks such as html service, load balancing, monitoring, and UPS to keep the switch and fronts up in case of a short power failure. It would seem that a facility that has redundant diesel engines to provide independent power for the whole site would make UPS redundant. A survey of data center users suggests power failures still happen yearly.
FIGURE 8.45  Front view, side view, and close-up of a rack of PCs used by Google. The photograph on the left shows the HP Procurve 4000M Ethernet switch in the middle, with 20 PCs above and 20 PCs below. Each PC connects via a Cat5 cable on the left side to the switch in the middle, running 100 Mbit Ethernet. Each “blade” of the switch can hold 8 100 Mbit Ethernet interfaces or 1 1Gbit interface. There are also two 1 Gbit Ethernet links leaving the switch on the right. Thus, each PC has only 2 cables: 1 Ethernet and 1 power cord. The far right of the photo shows a power strip, with each of the 40 PCs and the switch connected to it. Each PC is 1 VME rack unit (RU) high. The switch in the middle is 4 RU high. The photo on the middle is a close up of rack, showing contents of a 1 RU PC. This unit contains 2 Maxtor DiamondMax 5400 RPM IDE drives on the right of the box, 256 MB of 100 MHz SDRAM, a PC motherboard, a single power supply, and an Intel microprocessor. Each PC runs versions 2.2.16 or 2.2.17 Linix kernels on a slightly modified RedHat release. Between March 2000 and November 2000, over the period the Google site was populated, the microprocessor varied in performance from a 533 MHz Celeron to an 800 MHz Pentium III. The goal was selecting good cost performance, which was often close to $200 per chip. Disk capacity varied from 40 to 80 GB. You can see the Ethernet cables on the left, power cords on the right, and table Ethernet cables connected to the switch at the top of the figure. In December 2000 the unassembled parts costs are about $500 for the two drives, $200 for the microprocessor, $100 for the motherboard, and $100 for the DRAM. Including the enclosure, power supply, fans, cabling and so on, an assembled PC might cost $1300 to $1700. The drawing on the right shows that PCs are kept in two columns, front and back, so that a single rack holds 80 PCs and 2 switches. The typical power per PC is about 55 watts and about 70 watts per switch, so a rack uses about 4500 watts. Heat is exhausted into a 3-inch vent between the two columns, and the hot air is drawn out the top using fans. (The drawing shows uses 22 PCs per side each 2 RU high instead of the Google configuration of 40 1 RU PCs plus a switch per side.) (Photos and figure from Rackable Systems: http://www.rackable.com/advantage.htm).
The PC itself is a fairly standard: 2 Maxtor ATA/IDE drives, 256 MB of SDRAM, a modest Intel microprocessor, a PC motherboard, one power supply and a few fans. Each PC runs the Linux operating system. To get the best value per dollar, every 2-3 months Google increases the capacity of the drives or the speed of the processor. Thus, the 40 rack site shown above was populated between March and November 2000 has microprocessors that are from a 533 MHz Celeron to an 800 MHz Pentium III, disks that vary in capacity between 40 and 80 GB and in speed at 5400 to 7200 RPM, and memory bus speed is either 100 or 133 MHz.

Performance

Each collocation site connects to the Internet via OC48 (2488 Mbits/sec) links, which is shared by Google and the other Internet service providers. If a typical response to a query is, say, 4000 bytes, then the average bandwidth demand is

\[
\frac{70,000,000 \text{ queries/day} \times 4000 \text{ B/query} \times 8 \text{ bits/B}}{24 \times 60 \times 60 \text{ seconds/day}} = \frac{2,240,000 \text{ Mbits}}{86,400 \text{ seconds}} = 26 \text{ Mbit/s}
\]

which is just 1% of the link speed of each site. Even if we multiply by a factor of 4 to account for peak versus average demand and requests as well as responses, Google needs little of that bandwidth.

Crawling the web and updating the sites needs much more bandwidth than serving the queries. Let’s estimate some parameters to put things into perspective. Assume that it takes 7 days to crawl a billion pages:

\[
\frac{1,000,000,000 \text{ pages} \times 4000 \text{ B/page} \times 8 \text{ bits/B}}{24 \times 60 \times 60 \text{ seconds/day} \times 7 \text{ days}} = \frac{32,000,000 \text{ Mbits}}{604,800 \text{ seconds}} = 59 \text{ Mbit/s}
\]

This data is collected at a single site, but the final multi-terabyte index and repository must then be replicated at the other two sites. If we assume we have 7 days to replicate the data and that we are shipping, say, 5 terabytes from one site to two sites, then the average bandwidth demand is

\[
\frac{2 \times 5,000,000 \text{ MB} \times 8 \text{ bits/B}}{24 \times 60 \times 60 \text{ seconds/day} \times 7 \text{ days}} = \frac{80,000,000 \text{ Mbits}}{604,800 \text{ seconds}} = 132 \text{ Mbit/s}
\]

Hence, the machine to person bandwidth is relatively trivial, with the real bandwidth demand being machine to machine. Moreover, Google’s search rate is growing 20% per month, and the number of pages indexed has more than doubled every year since 1997, so bandwidth must be available for growth.

Time of flight for messages across the United States takes about 0.1 seconds, so it’s important for Europe to be served from the Virginia site and for California to be served by Silicon Valley sites. To try to achieve the goal of 1/2 second latency, Google software normally guesses where the search is from in order to reduce time of flight delays.
Cost

Given that the basic building block of the Google cluster is a PC, the capital cost of a site is typically a function of the cost of a PC. Rather than buy the latest microprocessor, Google looks for the best cost-performance. Thus, in March 2000 an 800 MHz Pentium III cost about $800, while a 533 MHz Celeron cost under $200, and the difference in performance couldn’t justify the extra $600 per machine. (When you purchase PCs by the thousands, every $100 per PC is important.) By November the price of the 800 MHz Pentium III dropped to $200, so it was a better investment. When accounting for this careful buying plus the enclosures and power supplies, your authors estimate the PC cost was $1300 to $1700.

The switches cost about $1500 for the HP Ethernet switch and about $100,000 each for the Foundry switches. If the racks themselves cost about $1000 to $2000 each, the total capital cost of a 40-rack site is about $4.5M to $6.0M. Including 3200 microprocessors and 0.8 terabytes of DRAM, the disk storage costs about $10,000 to $15,000 per terabyte. To put this into perspective, the leading performer for the TPC-C database benchmark in August 2001 is a scaled up version of the cluster from the last example. The hardware alone costs about $10.8M, which includes 280 microprocessors, 0.5 terabytes of DRAM, and 116 terabytes SCSI disks organized as RAID I. Ignoring the RAID I overhead, disk storage costs about $93,000 per terabyte, about a factor of 8 higher than Google despite having 1/8 the number of processors and about 5/8 the DRAM.

The Google rack with 80 PCs, with each PC operating at about 55 Watts, uses 4500 Watts in 10 square feet. It is considerably higher than the 1000 Watts per rack expected by the collocation sites. Each Google rack also uses 60 amps. As mentioned above, reducing power per PC is a major opportunity for the future of such clusters, especially as the cost per kilowatt hour is increasing and the cost per Mbits/second is decreasing.

Reliability

Not surprisingly, the biggest failure in the Google PC is software. On an average day, about 20 machines will be rebooted, and that normally solves the problem. To reduce the number of cables per PC as well as cost, Google has no ability to remotely reboot a machine. The software stops giving work to a machine when it observes unusual behavior, and the operator calls the collocation site and tells them to location of the machine that needs to be rebooted, and a person at the site finds the label and pushes the switch on the front panel. Occasionally the person hits the wrong switch either by mistake or due to mislabeling on the outside of the box.

The next reliability problem is the hardware, which has about 1/10th the failures of software. Typically, about 2% to 3% of the PCs have need to be replaced per year, with failures due to disks and DRAM accounting for 95% of these failures. The remaining 5% are due to problems with the motherboard, power supply, and connectors, and so on. The microprocessors themselves never seem to fail.
The DRAM failures are perhaps a third of the failures. Google sees errors both to bits changing inside DRAM and when bits transfer over the 100 to 133 MHz bus. There was no ECC protection available on PC desktop motherboard chip sets in 2000, so it was not used. The DRAM is determined to be the problem when Linux cannot be installed with a proper checksum until the DRAM is replaced. As PC motherboard chip sets become available, Google plans to start using ECC both to correct some failures but, more importantly, to make it easier to see when DRAMs fail. The extra cost of the ECC is trivial given the wide fluctuation in DRAM prices: careful purchasing procedures are more important than whether or not the DIMM has ECC.

Disks are the remaining PC failures. In addition to the standard failures that result in message to error log in the console, in almost equal numbers these disks will occasionally result in a *performance failure*, with no error message to the log. Instead of delivering normal read bandwidths at 28 Mbytes/second, disks will suddenly drop to 4 MB/second or even 0.8 MB/second. As the disks are under warranty for 5 years, Google sends the disks back to the manufacture for either operational or performance failures to get replacements. Thus, there has been no exploration of the reason for the disk anomalies.

When a PC has problems, it is reconfigured out of the system, and about once a week a person removes the broken PCs. They are usually repaired and then re-inserted into the rack.

In regards to the switches, over a 2-year period perhaps 200 of the HP Ethernet switches were deployed, and 2 to 3 have failed. None of the six Foundry switches has failed in the field, although some have had problems on delivery. These switches have a blade-based design with 16 blades per switch, and 2 to 3 of the blades have failed.

The final issue is collocation reliability. The experience of many Internet service providers is that once a year there will be a power outage that affects either the whole site or a major fraction of a site. On average, there is also a network outage so that the whole site is disconnected from the Internet. These outages can last for hours.

There also that collocation site reliability follows a “bathtub” curve: high failures in the beginning, which quickly fall to low rates in the middle, and then rises to high rates at the end. When they are new, the sites are empty and so continuously filled with new equipment. With more people and new equipment being installed, there is a higher outage rate. Once the site is full of equipment, there are fewer people around and less change, so the site has a low failure rate. Once the equipment becomes outdated and it starts being replaced, the activity in the site increases and so does the failure rate. Thus, the failure rate of site depends in part on its age, just as the classic bathtub reliability curves would predict. It is also a function of the people, and if there is a turnover in people, the fault rate can change.

Google accommodates collocation unreliability by having multiple sites with different network providers, plus leased lines between pairs of site for emergen-
In 1999, there were 76 million cellular subscribers in the United States, a 25% growth from the year before. That growth rate is almost 35% per year worldwide, as developing countries find it much cheaper to install cellular towers than copper-wire-based infrastructure. Thus, in many countries, the number of cell phones in use exceeds the number of wired phones in use.

Not surprisingly, the cellular handset market is growing at 35% per year, with about 280 million cellular phone handsets sold in 1999. To put that in perspective, in the same year sales of personal computers were 120 million. These numbers mean that tremendous engineering resources are available to improve cell phones, and cell phones are probably leaders in engineering innovation per cubic inch. [Grice, 2000].

Before unveiling the anatomy of a cell phone, let’s try a short introduction to wireless technology.

**Background on Wireless Networks**

Networks can be created out of thin air as well as out of copper and glass, creating wireless networks. Much of this section is based on a report from the National Research Council [1997].

A radio wave is an electromagnetic wave propagated by an antenna. Radio waves are modulated, which means that the sound signal is superimposed on the stronger radio wave that carries the sound signal, and hence is called the carrier signal. Radio waves have a particular wavelength or frequency: they are measured either the length of the complete wave or as the number of waves per second. Long waves have low frequencies and short waves have high frequencies. FM radio stations transmit on the band of 88 MHz to 108 MHz using frequency modulations (FM) to record the sound signal.

By tuning into different frequencies, a radio receiver can pick up a specific signal. In addition to AM and FM radio, other frequencies are reserved for citizen band radio, television, pagers, air traffic control radar, Global Positioning System, and so on. In the United States, the Federal Communications Commission decides who gets to use frequencies and for what purpose.

The bit error rate (BER) of a wireless link is determined by the received signal power, noise due to interference caused by the receiver hardware, interference from other sources, and characteristics of the channel. Noise is typically proportional to the radio frequency bandwidth, and a key measure is the signal-to-noise ratio (SNR) required to achieve a given BER. Figure 8.46 lists more challenges for wireless communication.
Typically, wireless communication is selected because the communicating devices are mobile or because wiring is inconvenient, which means the wireless network must rearrange itself dynamically. Such rearrangement makes routing more challenging. A second challenge is that wireless signals are not protected and hence are subject to mutual interference, especially as devices move. Power is the another challenge for wireless communication, both because the devices tend to be battery powered and because antennas radiate power to communicate and little of it reaches the receiver. As a result, raw bit error rates are typically a thousand to a million times higher than copper wire.

There are two primary architectures for wireless networks: base-station architectures and peer-to-peer architectures. Base stations are connected by land lines for longer distance communication, and the mobile units communicate only with a single local base station. Peer-to-peer architectures allow mobile units to communicate with each other, and messages hop from one unit to the next until delivered to the desired unit. Although peer-to-peer is more reconfigurable, base stations tend to be more reliable since there is only one hop between the device and the station. Cellular telephony, the most popular example of wireless networks, relies on radio with base stations.

Cellular systems exploit exponential path loss to reuse the same frequency at spatially separated locations, thereby greatly increasing the number of customers served. Cellular systems will divide a city into nonoverlapping hexagonal cells which use different frequencies if nearby, reusing a frequency only when cells are far enough apart so that mutual interference is acceptable.

At the intersection of three hexagonal cells is a base station with transmitters and antennas that is connected to a switching office which coordinates handoffs when a mobile device leaves one cell and goes into another, as well as to accept and place calls over land lines. Depending on topography, population and so on, the radius of a typical cell is two to ten miles.
The Cell Phone

Figure 8.47 shows the components of a radio, which is the heart of a cell phone. Radio signals are first received by the antenna, then amplified, passed through a mixer, then filtered, demodulated, and finally decoded. The antenna acts as the interface between the medium through which radio waves travel and electronics of the transmitter or receiver. Antennas can be designed to work best in particular directions, giving both transmission and reception directional properties. Modulation encodes information in the amplitude, phase, or frequency of the signal to increase its robustness under impaired conditions. Radio transmitters go through the same steps, just in the opposite order.

![Diagram of cell phone components](image)

**FIGURE 8.47** A radio receiver consists of an antenna, radio frequency amplifier, mixer, filters, demodulator, and decoder. A mixer accepts two signal input and forms an output signal at the sum and difference frequencies. Filters select a narrower band of frequencies to pass on to the next stage. Modulation encodes information to make it more robust. Decoding turns signals into information. Depending on the application, all electrical components can be either analog or digital. For example, a car radio is all analog components, but PC modem is all digital except for the amplifier. Today analog silicon chips are used for the RF amplifier and first mixer in cellular phones.

Originally, all components were analog, but over time most were replaced by digital components, requiring the radio signal to be converted from analog to digital. The desire for flexibility in the number of radio bands led to software routines replacing some of these functions in programmable chips, such as digital signal processors. Because such processors are typically found in mobile devices, emphasis is placed on performance per joule to extend battery life, performance per square millimeter of silicon to reduce size and cost, and bytes per task to reduce memory size.

Figure 8.48 shows the generic block diagram of the electronics of a cell phone handset, with the DSP performing the signal processing and the microcontroller handling the rest of the tasks. Cell phone handsets are basically mobile computers acting as a radio. The include standard I/O devices—keyboard and LCD display—plus a microphone, speaker, and antenna for wireless networking. Battery efficiency affects sales, both in standby power when waiting for a call and in minutes of speaking.
When a cell phone is turned on, the first task is to find a cell. It scans the full bandwidth to find the strongest signal, which it keeps doing every seven seconds or if the signals strength drops, as its designed to work from moving vehicles. It then picks an unused radio channel. The local switching office registers the cell phone and records its phone number and electronic serial number, and assigns it a voice channel for the phone conversation. To be sure the cell phone got the right channel, the base station sends a special tone on it, which the cell phone sends back to acknowledge it. The cell phone times out after five seconds of it doesn’t hear supervisory tone, and starts the process all over again. The original base station makes a handoff request to the incoming base station as the signal strength drops off.

To achieve a two way conversation over radio, frequency bands are set aside for each direction, forming a frequency pair or channel. The original cellular base stations transmitted at 869.04 to 893.97 (called the forward path) and cell phones transmitted at 824.04 MHz to 848.97 MHz (called the reverse path), with the frequency gap to keep them from interfering with each other. Cells might have had between 4 and 80 channels. Channels were divided into setup channels for call setup, and voice channels that handle the data or voice traffic.

The communication is done digitally, just like a modem, at 9,600 bits/second. Since wireless is a lossy medium, especially from a moving vehicle, the handset send each message is five times. To preserve battery life, the original cell phones typically transmit at two signal strengths—0.6 watts and 3.0 watts—depending on the distance to cell. This relatively low power not only allows smaller batteries and thus smaller cell phones, it aids frequency reuse, which is key to cellular telephony.

Figure 8.49 shows a circuit board from an Ericsson digital phone, with the components identified. Note that the board contains two processors. A Z-80 mi-
crocontroller is responsible for controlling the functions of the board, I/O with the keyboard and display, and coordinating with the base station. The DSP handles all signal compression and decompression. In addition there are dedicated chips for Analog-to-Digital and Digital-to-Analog conversion, amplifiers, power management, and RF interfaces.

In 2001, a cell phone has about 10 integrated circuits, including parts made in exotic technologies like gallium arsenide and silicon germanium as well as to standard CMOS. The economics and desire for flexibility will likely shrink this to a few chips, but it appears that a separate microcontroller and DSP will be found inside those chips, with code implementing many of the functions.

Cell Phone Standards and Evolution

Improved communication speeds for cellular phone were developed, with multiple standards. Code division multiple access (CDMA), as one popular example, uses a wider radio frequency band for a path than the original cellular phones, called AMPS for Advanced Mobile Phone Service, a mostly analog system. The wider frequency makes it more difficult to block, and is called spread spectrum. Other standards are time division multiple access (TDMA) and global system for mobile communication (GSM). These second generation standards—CDMA, GSM, and TDMA—are mostly digital.
The big difference for CDMA is that all callers share the same channel, which operates at a much higher rate, and then distinguishes the different calls by encoding each one uniquely. Each CDMA phone call starts at 9600 bits/second, it is then encoded and transmitted as equal sized messages at 1.25 megabits/second. Rather than send each signal five times as in AMPS, each bit is stretched so that it takes eleven times the minimum frequency, thereby accommodating interference and yet successful transmission. The base station receives the messages its separates them into the separate 9600 bits/second streams for each call.

To enhance privacy, CDMA uses pseudo-random sequences from a set of 64 predefined codes. To synchronize the handset and base station so as to pick a common pseudo-random seed, CDMA relies on a clock from the Global Positioning System, which continuously transmits an accurate time signal. By carefully selecting the codes, the shared traffic sounds like random noise to the listener. Hence, as more users share a channel there is more noise, and the signal to noise ratio gradually degrades. Thus, the capacity of the CDMA system is a matter of taste, depending upon sensitivity of the listener to background noise.

In addition, CDMA uses speech compression and varies the rate of data transferred depending how much activity is going on in the call. Both these techniques preserve bandwidth, which allows for more calls per cell. CDMA must regulate power carefully so that signals near the cell tower do not overwhelm those from far away, with the goal of all signals reach the tower at about the same level. The side benefit is that CDMA handsets emit less power, which both helps battery life and increases capacity when users are close to the tower.

Thus, compared to AMPS, CDMA improves capacity of a system by up to an order of magnitude, has better call quality, has better battery life, and enhances users’ privacy. After considerable commercial turmoil, there is a new third generation standard called International Mobile Telephony 2000 (IMT-2000) which is based primarily on two competing versions of CDMA and one TDMA. This standard may lead to cell phones which work anywhere in the world.

8.14 Fallacies and Pitfalls

Myths and hazards are widespread with interconnection networks. This section has just a few warnings, so proceed carefully.

*Pitfall: Using bandwidth as the only measure of network performance.*

Many network companies apparently believe that given sophisticated protocols like TCP/IP that maximize delivered bandwidth, there is only one figure of merit for networks. This may be true for some applications, such as video, where there is little interaction between the sender and the receiver. Many applications, however, are of a request-response nature, and so for every large message there must be one or more small messages. One example is NFS.
Figure 8.50 compares a shared 10 Mbits/second Ethernet LAN to a switched 155 Mbits/second ATM LAN for NFS traffic. Ethernet drivers were better tuned than the ATM drivers, such that 10 Mbits/s Ethernet was faster than 155 Mbits/s ATM for payloads of 512 bytes or less. Figure 8.50 shows the overhead time, transmission time, and total time to send all the NFS messages over Ethernet and ATM. The peak link speed of ATM is 15 times faster and the measured link speed for 8-KB messages is almost 9 times faster. Yet the higher overheads offset the benefits so that ATM would transmit NFS traffic only 1.2 times faster.
Pitfall: Ignoring software overhead when determining performance.

Low software overhead requires cooperation with the operating system as well as with the communication libraries. Figure 8.50 gives one example.

Another example comes from supercomputers. The CM-5 supercomputer had a software overhead of 20 µsecs to send a message and a hardware overhead of 0.5 microseconds. The Intel Paragon reduced the hardware overhead to just 0.2 microseconds, but the initial release of software has a software overhead of 250 microseconds. Later releases reduced this overhead to 25 microseconds, which still dominates the hardware overhead.

This pitfall is simply Amdahl’s Law applied to networks: Faster network hardware is superfluous if there is not a corresponding decrease in software overhead.

Pitfall: Trying to provide features only within the network vs. end-to-end.

The concern is providing features at a lower level that only partially satisfy the communication demand that can only be accomplished at the highest level. Saltzer, Reed, and Clark [1984] give the end-to-end argument as

The function in question can completely and correctly be specified only with the knowledge and help of the application standing at the endpoints of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. [page 278]

Their example of the pitfall was a network at MIT that used several gateways, each of which added a checksum from one gateway to the next. The programmers of the application assumed the checksum guaranteed accuracy, incorrectly believing that the message was protected while stored in the memory of each gateway. One gateway developed a transient failure that swapped one pair of bytes per million bytes transferred. Over time the source code of one operating system was repeatedly passed through the gateway, thereby corrupting the code. The only solution was to correct the infected source files by comparing to paper listings and repairing the code by hand! Had the checksums been calculated and checked by the application running on the end systems, safety would have been assured.

There is a useful role for intermediate checks, however, provided that end-to-end checking is available. End-to-end checking may show that something is broken between two nodes, but it doesn’t point to where the problem is. Intermediate checks can discover the broken component.

A second issue regards performance using intermediate checks. Although it is sufficient to retransmit the whole in case of failures from the end point, it can be much faster to retransmit a portion of the message at an intermediate point rather than wait for time-out and a full message retransmit at the end point. Balakrishnan et al [1997] found that, for wireless networks, such an intermediate retransmission for TCP/IP communication results in 10-30% higher throughput.
8.15 Concluding Remarks

Pitfall: Relying on TCP/IP for all networks, regardless of latency, bandwidth, or software requirements.

The network designers on the first workstations decided it would be elegant to use a single protocol stack no matter where the destination of the message: across a room or across an ocean, the TCP/IP overhead must be paid. This might have been a wise decision especially given the unreliability of early Ethernet hardware, but it sets a high software overhead barrier for commercial systems. Such an obstacle lowers the enthusiasm for low-latency network interface hardware and low-latency interconnection networks if the software is just going to waste hundreds of microseconds when the message must travel only dozens of meters. It also can use significant processor resources. One rough rule of thumb is that each Mbits/second of TCP/IP bandwidth needs about 1 MHz of processor speed, and so a 1000 Mbits/second link could saturate a processor with a 800 to 1000 MHz clock.

The flip side is that from a software perspective, TCP/IP is the most desirable target since it is the most connected and hence largest number of opportunities. The downside of using software optimized to a particular LAN or SAN is that it is limited. For example, communication from a Java program depends on TCP/IP, so optimization for another protocol would require creation of glue software to interface Java to it.

TCP/IP advocates point out that the protocol itself is theoretically not as burdensome as the current implementations, but progress has been modest in commercial systems. There are also TCP/IP off-loading engines entering the market, with the hope of preserving the universal software model while reducing processor utilization and message latency. If processors to continue to improve much faster than network speeds, or if multiple processors become ubiquitous, software TCP/IP may become less significant on processor utilization and message latency.

Networking is one of the most exciting fields in computer science and engineering today. The purpose of this chapter to lower the cost of entry into this field by providing definitions and the basic issues so that readers can more easily go into more depth.

The Internet and World Wide Web pervade our society and will likely revolutionize how we access information. Although we couldn’t have the Internet without the telecommunication media, it is protocol suites such as TCP/IP that make electronic communication practical. More than most areas of computer science and engineering, these protocols embrace failures as the norm; the network must operate reliably in the presence of failures. Interconnection network hardware and software blend telecommunications with data communications, calling into
question whether they should remain as separate academic disciplines or be combined into a single field.

The silicon revolution has made its way to the switch: just as the “killer micro” changed computing, whatever turns out to be the “killer network” will transform communication. We are seeing the same dramatic change in cost/performance in switches as the mainframe-minicomputer-microprocessor change did to processors. In 2001, companies that make switches are acquiring companies that make embedded microprocessors, just to have better microprocessors for their switches.

Inexpensive switches mean that network bandwidth can scale with the number of nodes, even to the level of the traditional I/O bus. Both I/O designers and memory system designers must consider how to best select and deploy switches. Thus, networking issues apply to all levels of computers systems today: communication within chips, between chips on a board, between boards, and between computers in a machine room, on a campus, or in a country.

The availability and scalability of networks are transforming the machine room. Disks are being connected over SAN to servers versus being directly attached, and clusters of smaller computers connected by a LAN are replacing large servers. The cost-performance, scalability, and fault isolation of clusters have made them attractive to diverse communities: database, scientific computing, and Internet service providers. It’s hard to think what else these communities have in common. The challenges for clusters today are the cost of administration.

After decades of low network performance on shared media, networking is in “catch up” mode, and should improve faster than microprocessors. We are not near any performance plateaus, so we expect rapid advance SANs, LANs, and WANs.

This greater network performance is key to the information and communication centric vision of the future of our field. The dramatic improvement in cost/performance of communications has enabled millions of people around the world to find others with common interests. As the quotes at the beginning of this chapter suggest, the authors believe this revolution in two-way communication will change the form of human associations and actions.

### 8.16 Historical Perspective and References

This chapter has taken the unusual perspective that computers inside the machine room on a LAN or SAN and computers on an intercontinental WAN share many of the same concerns. Although this observation may be true, their histories are very different. We highlight readings on each topic, but good general texts on networking have been written by Davie, Peterson, and Clark [1999] and by Kurose and Ross [2001].
Wide Area Networks

The earliest of the data interconnection networks are WANs. The forerunner of the Internet is the ARPANET, which in 1969 connected computer science departments across the U.S. that had research grants funded by the Advanced Research Project Agency (ARPA), a U.S. government agency. It was originally envisioned as using reliable communications at lower levels. It was the practical experience with failures of underlying technology that led to the failure-tolerant TCP/IP, which is the basis for the Internet today. Vint Cerf and Robert Kahn are credited with developing the TCP/IP protocols in the mid 1970s, winning the ACM Software Award in recognition of that achievement. Kahn [1972] is an early reference on the ideas of ARPANET. For those interested in learning more about TCP/IP, Stevens [1994] has written classic books on the topic.

In 1975, there were roughly 100 networks in the ARPANET and only 200 in 1983; in 1995 the Internet encompasses 50,000 networks worldwide, about half of which are in the United States. In 2000, that number is hard to calculate, but the number of IP hosts grew by a factor of 20 in five years. The key networks that made the Internet possible, such as ARPANET and NSFNET, have been replaced by fully commercial systems, and yet the Internet still thrives.

The exciting application of the Internet is the World Wide Web, developed by Tim Berners-Lee, a programmer at the European Center for Particle Research (CERN) in 1989 for information access. In 1992, a young programmer at the University of Illinois, Marc Andreessen, developed a graphical interface for Web called Mosaic. It became immensely popular. He later became a founder of Netscape, which popularized commercial browsers. In May 1995, at the time of the second edition of this book, there were over 30,000 web pages, and the number was doubling every two months. In November 2000, during the writing of the third edition of this book, there were almost 100 million Internet hosts and more than 1.3 billion WWW pages.

Alles [1995] offers a good survey on ATM. ATM is just the latest of the ongoing standards set by the telecommunications industry, and it is undoubtedly the future for this community. Communication forces standardization by competitive companies, sometimes leading to anomalies. For example, the telecommunication companies in North America wanted to use 64-byte packets to match their existing equipment, while the Europeans wanted 32-byte packets to match their existing equipment. The new standard compromise was 48 bytes to ensure that neither group had an advantage in the marketplace!

Finally, WANs today rely on fiber. Fiber has made such advances that it’s original assumption of packet switching is no longer true: WAN bandwidth is not precious. Today WAN fibers are often underutilized. Goralski [1997] discusses advances in fiber optics.
Local Area Networks

ARPA’s success with wide area networks led directly to the most popular local area networks. Many researchers at Xerox Palo Alto Research Center had been funded by ARPA while working at universities, and so they all knew the value of networking. In 1974, this group invented the Alto, the forerunner of today’s desk-top computers [Thacker et al. 1982], and the Ethernet [Metcalfe and Boggs 1976], today’s LAN. This group--David Boggs, Butler Lampson, Ed McCreight, Bob Sprowl, and Chuck Thacker--became luminaries in computer science and engineering, collecting a treasure chest of awards between them.

This first Ethernet provided a 3 Mbits/sec interconnection, which seemed like an unlimited amount of communication bandwidth with computers of that era. It relied on the interconnect technology developed for the cable television industry. Special microcode support gave a round-trip time of 50 microseconds for the Alto over Ethernet, which is still a respectable latency. It was Boggs’ experience as a ham radio operator that led to a design that did not need a central arbiter, but instead listened before use and then varied back-off times in case of conflicts.

The announcement by Digital Equipment Corporation, Intel, and Xerox of a standard for 10 Mbits/sec Ethernet was critical to the commercial success of Ethernet. This announcement short-circuited a lengthy IEEE standards effort, which eventually did publish IEEE 802.3 as an standard for Ethernet.

There have been several unsuccessful candidates in trying to replace the Ethernet. The FDDI committee, unfortunately, took a very long time to agree on the standard and the resulting interfaces were expensive. It was also a shared me-dium when switches are becoming affordable. ATM also missed the opportunity due in part to the long time to standardize the LAN version of ATM.

Due to failures of the past, LAN modernization efforts have been centered on extending Ethernet to lower cost media, to switched interconnect, to higher link speeds, and to new domains such as wireless communication. Spurgeon [2001] has a nice on-line summary of Ethernet technology, including some of its history.

Massively Parallel Processors

One of the places of innovation in interconnect networks was in massively parallel processors (MPPs). An early MPP was the Cosmic Cube [Seitz 1985], which used Ethernet interface chips to connect 8086 computers in a hypercube. SAN interconnections have improved considerably since then, with messages routed automatically through intermediate switches to their final destinations at high bandwidths and with low latency. Considerable research has gone into the benefits over different topologies in both construction and program behavior. Whether due to faddishness or changes in technology is hard to say, but topologies certainly become very popular and then disappear. The hypercube, widely popular in the 1980s, almost disappeared from MPPs of the 1990s. Cut-through routing, howev-er, has been preserved and is covered by Dally and Seitz [1986].
Chapter 6 records the poor current state of such machines. Government programs such as the Accelerated Strategic Computing Initiative (ASCI) still result in a handful of one-of-a-kind MPPs costing $50 to $100 million, yet these are basically clusters of SMPs.

Clusters

Clusters were probably “invented” in the 1960s by customers who could not fit all their work in one computer, or who needed a backup machine in case of failure of the primary machine [Pfister, 1998]. Tandem introduced a 16-node cluster in 1975. Digital followed with VAXclusters, introduced in 1984. They were originally independent computers that shared I/O devices, requiring a distributed operating system to coordinate activity. Soon they had communication links between computers, in part so that the computers could be geographically distributed to increase availability in case of a disaster at a single site. Users log onto the cluster and are unaware of which machine they are running on. DEC (now Compaq) sold more than 25,000 clusters by 1993. Other early companies were Tandem (now Compaq) and IBM (still IBM), and today virtually every company has cluster products. Most of these products are aimed at availability, with performance scaling as a secondary benefit. Yet in 2000 clusters generally dominate the list of top performers of the TPC-C database benchmark.

Scientific computing on clusters emerged as a competitor to MPPs. In 1993, the Beowulf Project started with the goal of fulfilling NASA’s desire for a 1 GFLOPS computer for under $50,000. In 1994, a 16-node cluster build from off the shelf PCs using 80486s achieved that goal [Bell 2001]. This emphasis led to a variety of software interfaces to make it easier to submit, coordinate, and debug large programs or a large number of independent programs. In 2001, the fastest (and largest) supercomputers are typically clusters, at least by some popular measures.

Efforts were made to reduce latency of communication in clusters as well as to increase bandwidth, and several research projects worked on that problem. (One commercial result of the low latency research was the VI interface standard, which has been embraced by Infiniband, discussed below.) Low latency then proved useful in other applications. For example, in 1997 a cluster of 100 UltraS-PARC desktop computers at UC Berkeley, connected by 160-MB/sec per link Myrinet switches, was used to set world records in database sort—sorting 8.6 GB of data originally on disk in one minute—and in cracking an encrypted message—taking just 3.5 hours to decipher a 40-bit DES key. This research project, called Network of Workstations [Anderson et al, 1995], also developed the Inktomi search engine, which led to a startup company with the same name.

System or Storage Area Networks (SANs)

At the second edition of this book, a new class of networks was emerging: system area networks. These networks are designed for a single room or single floor and thus the length is ten to hundreds of meters, and were for use in clusters. Close distance means the wires can be wider and faster at lower cost, network hardware can ensure in order delivery, and cascading switches consume less handshaking time. There is also less reason to go to the cost of optical fiber, since the distance advantage of fiber is less important for SANs. The limited size of the networks also makes source-based routing plausible, further simplifying the network. Both Tandem Computers and Myricom sold SANs.

In the intervening years the acronym SAN has been co-opted to also mean storage area networks, whereby networking technology is used to connect storage devices to compute servers. Today most people mean storage when they say SAN. The most widely used example in 2001 is Fibre Channel Arbitrated Loop (FC-AL). Not only are disk arrays attached to servers via FC-AL links, there are even some disks with FC-AL links. There are also companies selling FC-AL switches so that storage area networks can enjoy the benefits of greater bandwidth and interconnectivity of switching.

In October 2000 the Infiniband Trade Association announced version 1.0 specification of Infiniband. Led by Intel, HP, IBM, Sun, and other companies, it was proposed as a successor to the PCI bus that brings point-to-point links and switches with its own set of protocols. Its characteristics are desirable potentially both for system area networks to connect clusters and for storage area networks to connect disk arrays to servers. To learn more, the Infiniband standard [2001] is available on the WWW.

The chief competition for Infiniband is the rapidly improving Ethernet technology. The Internet Engineering Task Force is proposing a standard called iSCSI to send SCSI command over IP networks (Satran[2001]). Given the likely cost advantages of the higher volume Ethernet switches and interface cards, in 2001, it is unclear who will win.

Will Infiniband take over the machine room, leaving the WAN as the only link that is not Infiniband? Or will Ethernet will dominate the machine room, even taking over some of the role of storage area networks, leaving Infiniband to simply be an I/O bus replacement? Or will there be a three-level solution: Infiniband in the machine room, Ethernet in the building and on the campus, and then WAN for country? Will TCP/IP off-loading engines become available that can reduce processor utilization and provide low latency yet still provide the software interfaces and generality of TCP/IP? Or will software TCP/IP and faster multiprocessors be sufficient?

In 2001, it is very hard to tell which will win. A wonderful characteristic of computer architecture is that such issues will not remain endless academic debates, unresolved as people rehash the same arguments repeatedly. Instead, the battle is fought in the marketplace, with well-funded and talented groups giving
their best efforts at shaping the future. Moreover, constant changes to technology reward those who are either astute or lucky. The best combination of technology and follow-through has often determined commercial success.

Let the games begin! Time will tell us who wins and who loses, and we will likely know the score by the next edition of this text.

References


**E X E R C I S E S**

- Using the examples from section 8.11, use the techniques from Chapter 7 to calculate the reliability of the cluster. The results of failures on Tertiary Disk give one set of failure information. What is the MTTF? Where are the single points of failure? How could the designs be changed to improve MTTF?

- Along similar lines, calculate the performance bottlenecks? How does it change if we use rules of thumb on utilization for Chapter 7 vs. assuming 100% utilization?

- The SAN versions just use FC-AL loops versus adding a FC-AL switch. What would have to change in the disk system to make a FC-AL switch valuable? (RAID is the bottleneck with only a single FC-AL loop between the box and the server.)

- Undoubtedly the top 10 of TPC-C has changed. Find a cluster from Dell or Compaq, and go to their web sites to determine the prices of the varying cluster strategies as we did in the examples. Note that the execute overview lists all the components and their prices at the time of the benchmark. They can serve as good placeholders until or unless you can find the current real prices online. They also supply maintenance costs.

- Add a discussion question on use of Ethernet vs. Infiniband in the machine room. What are the technical advantages of each? What are the economic ad-
vantages of each? Why would people maintaining the system prefer one to the other?

- In all exercises, should go to faster Ethernet (and ATM).
- We could use 5 to 10 more exercises.
- Some simple ones: go to the TPC web site and look at which architectures--clusters vs. some form of multiprocessors--dominate each benchmark in performance and in cost performance. Make a discussion question as to why this might vary between benchmarks. How has it changed since the data in the figure? Have the trends continued, or not?
- Do a similar study for the Linpack benchmarks (List of Top 500 supercomputers). See if there is older versions of the list so you can see how machine types and brand names change over time. How has it changed since the data in the figure? Have the trends continued, or not?
- If you have access to an SMP and a cluster, write a program to measure latency of communication and bandwidth of communication between processors.

8.1 [15] Assume the overhead to send a zero-length data packet on an Ethernet is 500 microseconds and that an unloaded network can transmit at 90% of the peak 10 Mbits/sec rating. Plot the delivered bandwidth as the data transfer size varies from 32 bytes to 1500.
- Change Ethernet speed in the next one. Figure still there?

8.2 [15] One reason that ATM has a fixed transfer size is that when a short message is behind a long message, a node may need to wait for an entire transfer to complete. For applications that are time-sensitive, such as when transmitting voice or video, the large transfer size may result in transmission delays that are too long for the application. On an unloaded interconnection, what is the worst-case delay if a node must wait for one full-size Ethernet packet versus an ATM transfer? See Figure 8.20 (page 605) to find the packet sizes. For this question assume you can transmit at 100% of the 155 Mbits/sec of the ATM network and 100% of the 10 Mbits/sec Ethernet.
- Update next one to larger tapes, speeds. Match assumptions in revised example?

8.3 [20/10] Is electronic communication always fastest for longer distances than the Example on page 583? Calculate the time to send 100 GB using 10 8-mm tapes and an overnight delivery service versus sending 100 GB by FTP over the Internet. Make the following four assumptions:
- The tapes are picked up at 4 P.M. Pacific time and delivered 4200 km away at 10 A.M. Eastern time (7 A.M. Pacific time).
- On one route the slowest link is a T1 line, which transfers at 1.5 Mbits/sec.
- On another route the slowest link is a 10 Mbits/sec Ethernet.
You can use 50% of the slowest link between the two sites.

a. [20] <8.3> Will all the bytes sent by either Internet route arrive before the overnight delivery person arrives?

b. [10] <8.3> What is the bandwidth of overnight delivery? Calculate the average bandwidth of overnight delivery service for a 100-GB package.

Perhaps a next exercise can add bandwidth of networking links on campus and over the internet. Mary Baker at Stanford has created a new set of software that is much more efficient at finding bandwidth. Latency can be figured out from ping and traceroute (I recall). I can imagine several exercises along these lines.

8.4 [20/20/20/20] <8.8> If you have access to a UNIX system, use ping to explore the Internet. First read the manual page. Then use ping without option flags to be sure you can reach the following sites. It should say that X is alive. Depending on your system, you may be able to see the path by setting the flags to verbose mode (-v) and trace route mode (-R) to see the path between your machine and the example machine. Alternatively, you may need to use the program traceroute to see the path. If so, try its manual page. You may want to use the UNIX command script to make a record of your session.

a. [20] <8.8> Trace the route to another machine on the same local area network.

b. [20] <8.8> Trace the route to another machine on your campus that is not on the same local area network.

c. [20] <8.8> Trace the route to another machine off campus. For example, if you have a friend you send email to, try tracing that route. See if you can discover what types of networks are used along that route.

d. [20] <8.8> One of the more interesting sites is the McMurdo NASA government station in Antarctica. Trace the route to mcmvax.mcmurdo.gov.

8.5 [12/15/15] <8.4> Assume 64 nodes and 16 × 16 ATM switches in the following. (This exercise was suggested by Mark Hill.)

a. [12] <8.4> Design a switch topology that has the minimum number of switches.

b. [15] <8.4> Design a switch topology that has the minimum latency through the switches. Assume unit delay in the switches and zero delay for wires.

c. [15] <8.4> Design a switch topology that balances the bandwidth required for all links. Assume a uniform traffic pattern.

I think this example was dropped?

8.6 [20] <8.4> Redo the cut-through routing calculation for CM-5 on page 590 of different sizes: 64, 256, and 1024 nodes.

I dropped the all-to-all example. Perhaps put in as exercise? Reword and see if this exercise still makes sense

8.7 [15] <8.4> Calculate the time to perform a broadcast (from-one-to-all) on each of the
Exercises

8.8 [20] <8.4> The two Examples on pages 584–588 assumed unlimited bandwidth between the node and the network interface. Redo the calculations in Figure 8.17 on page 598, this time assuming a node can only issue one message in a time unit.

8.9 [15] <8.4> Compare the interconnection latency of a crossbar, Omega network, and fat tree with eight nodes. Use Figure 8.13 on page 593 and add a fat tree similar to Figure 8.14 on page 595 as a third option. Assume that each switch costs a unit time delay. Assume the fat tree randomly picks a path, so give the best case and worst case for each example. How long will it take to send a message from node P0 to P6? How long will it take P1 and P7 to also communicate?

8.10 [15] <8.4> One interesting measure of the latency and bandwidth of an interconnection is to calculate the size of a message needed to achieve one-half of the peak bandwidth. This halfway point is sometimes referred to as \( n_{1/2} \), taken from the vector processing. Using Figure 7.36 on page 621, estimate \( n_{1/2} \) for TCP/IP message using ATM and the Ethernet.

8.11 [15] <8.8> Use FTP to transfer a file from a remote site and then between local sites on the same LAN. What is the difference in bandwidth for each transfer? Try the transfer at different times of day or days of the week. Is the WAN or LAN the bottleneck?

8.12 [15] <8.4> Draw the topology of a 6-cube similar to the drawing of the 4-cube in Figure 8.16 on page 597.

8.13 [12/12/12/15/15/18] <8.7> Use M/M/1 queuing model to answer this exercise. Measurements of a network bridge show that packets arrive at 200 packets per second and that the gateway forwards them in about 2 ms.

a. [12] <8.7> What is the utilization of the gateway?

b. [12] <8.7> What is the mean number of packets in the gateway?

c. [12] <8.7> What is the mean time spent in the gateway?

d. [15] <8.7> Plot the response time versus utilization as you vary the arrival rate.

e. [15] <8.7> For an M/M/1 queue, the probability of finding \( n \) or more tasks in the system is Utilization\(^n\). What is the chance of an overflow of the FIFO if it can hold 10 messages?

f. [18] <8.7> How big must the gateway be to have packet loss due to FIFO overflow to

topologies in Figure 8.17 on page 598, making the same assumptions as the two Examples on pages 584–588.

- I dropped the all-to-all example. Reword and see if this exercise still makes sense.
be less than one packet per million?

8.14 [20] <8.7> The imbalance between the time of sending and receiving can cause problems in network performance. Sending too fast can cause the network to back up and increase the latency of messages, since the receivers will not be able to pull out the message fast enough. A technique called bandwidth matching proposes a simple solution: Slow down the sender so that it matches the performance of the receiver [Brewer 1994]. If two machines exchange an equal number of messages using a protocol like UDP, one will get ahead of the other, causing it to send all its messages first. After the receiver puts all these messages away, it will then send its messages. Estimate the performance for this case versus a bandwidth-matched case. Assume the send overhead is 200 microseconds, the receive overhead is 300 microseconds, time of flight is 5 microseconds, and latency is 10 microseconds, and that the two machines want to exchange 100 messages.

8.15 [40] <8.7> Compare the performance of UDP with and without bandwidth matching by slowing down the UDP send code to match the receive code as advised by bandwidth matching [Brewer 1994]. Devise an experiment to see how much performance changes as a result. How should you change the send rate when two nodes send to the same destination? What if one sender sends to two destinations?