Communication

❖ Layered Protocols

Due to the absence of shared memory, all communication in distributed systems is based on sending and receiving (low level) messages. When process A wants to communicate with process B, it first builds a message in its own address space. Then.it executes a system call that causes the operating system to send the message over the network to B. Although this basic idea sounds simple enough, in order to prevent chaos, A and B have to agree on the meaning of the bits being sent. If A sends a brilliant new novel written in French and encoded in IBM's EBCDIC character code, and B expects the inventory of a supermarket written in English and encoded in ASCII, communication will be less than optimal. Many different agreements are needed. How many volts should be used to signal an O-bit, and how many volts for an I-bit? How does the receiver know which is the last bit of the message? How can it detect if a message has been damaged or lost, and what should it do if it finds out? How long are numbers, strings, and other data items, and how are they represented? In short, agreements are needed at a variety of levels, varying from the low-level details of bit transmission to the high-level details of how information is to be expressed.

To make it easier to deal with the numerous levels and issues involved in communication, the International Standards Organization (ISO) developed a reference model that clearly identifies the various levels involved, gives them standard names, and points out which level should do which job. This model is called the Open Systems Interconnection Reference Model (Day and Zimmerman, 1983), usually abbreviated as ISO OSI or sometimes just the OSI model.

The OSI model is designed to allow open systems to communicate. An open system is one that is prepared to communicate with any other open system by using standard rules

that govern the format, contents, and meaning of the messages sent and received. These rules are formalized in what are called protocols. To allow a group of computers to communicate over a network, they must all agree on the protocols to be used. A distinction is made between two general types of protocols. With connection oriented protocols, before exchanging data the sender and receiver first explicitly establish a connection, and possibly negotiate the protocol they will use. When they are done, they must release (terminate) the connection. The telephone is a connection-oriented communication system. With connectionless protocols, no setup in advance is needed. The sender just transmits the first message when it is ready. Dropping a letter in a mailbox is an example of connectionless communication. With computers, both connection-oriented and connectionless communication are common.

In the OSI model, communication is divided up into seven levels or layers, as shown in Fig. 4-1. Each layer deals with one specific aspect of the communication. In this way, the problem can be divided up into manageable pieces, each of which can be solved independent of the others. Each layer provides an interface to the one above it. The interface consists of a set of operations that together define the service the layer is prepared to offer its users.

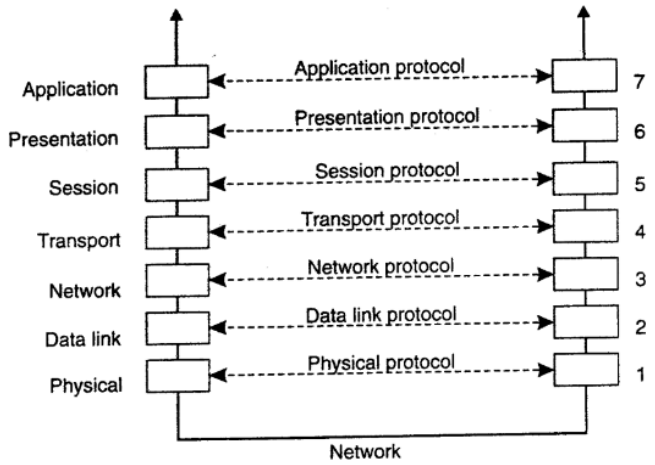

Figure 4-1. Layers, interfaces, and protocols in the OSI model.

When process A on machine 1 wants to communicate with process B on machine 2, it builds a message and passes the message to the application layer on its machine. This layer might be a library procedure, for example, but it could also be implemented in some other way (e.g., inside the operating system, on an external network processor, etc.). The application layer software then adds a header to the front of the message and passes the resulting message across the layer 6/7 interface to the presentation layer. The presentation layer in tum adds its own header and passes the result down to the session layer, and so on. Some layers add not only a header to the front, but also a trailer to the end. When it hits the bottom, the physical layer actually transmits the message (which by now might look as shown in Fig. 4-2) by putting it onto the physical transmission medium.

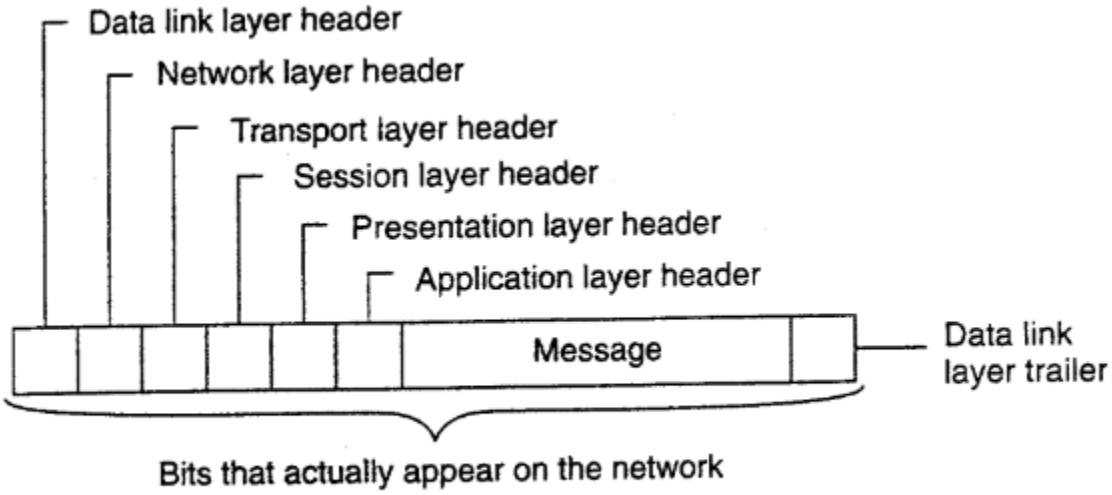

Figure 4-2. A typical message as it appears on the network.

When the message arrives at machine 2, it is passed upward, with each layer stripping off and examining its own header. Finally, the message arrives at the receiver, process B, which may reply to it using the reverse path. The information in the layer n header is used for the layer n protocol.

Lower-Level Protocols

It consist of three lowest layers of the OSI protocol suite.

1. The Physical layer

The physical layer is concerned with transmitting the Os and Is. How many volts to use for 0 and 1, how many bits per second can be sent, and whether transmission can take place in both directions simultaneously are key issues in the physical layer. In addition, the size and shape of the network connector (plug), as well as the number of pins and meaning of each are of concern here.

The physical layer protocol deals with standardizing the electrical, mechanical, and signaling interfaces so that when one machine sends a 0 bit it is actually received as a 0 bit and not a 1 bit. Many physical layer standards have been developed (for different media), for example, the RS-232-C standard for serial communication lines.

The physical layer just sends bits. As long as no errors occur, all is well. However, real communication networks are subject to errors, so some mechanism is needed to detect and correct them. This mechanism is the main task of the data link layer. What it does is to group the bits into units, sometimes called frames, and see that each frame is correctly received.

2. The Datakink layer

The data link layer does its work by putting a special bit pattern on the start and end of each frame to mark them, as well as computing a checksum by adding up all the bytes in the frame in a certain way. The data link layer appends the checksum to the frame. When the frame arrives, the receiver recomputes the checksum from the data and compares the result to the checksum following the frame. If the two agree, the frame is considered correct and is accepted. It they disagree. the receiver asks the sender to retransmit it. Frames are assigned se quence numbers (in the header), so everyone can tell which is which. On a LAN, there is usually no need for the sender to locate the receiver. It just puts the message out on the network and the receiver takes it off. A wide-area net work, however, consists of a large number of machines, each with some number of lines to other machines, rather like a large-scale map showing major cities and roads connecting them. For a message to get from the sender to the receiver it may have to make a number of hops, at each one choosing an outgoing line to use.

The question of how to choose the best path is called routing, and is essentially the primary task of the network layer. The problem is complicated by the fact that the

shortest route is not always the best route. What really matters is the amount of delay on a given route, which, in tum, is related to the amount of traffic and the number of messages queued up for transmission over the various lines. The delay can thus change over the course of time. Some routing algorithms try to adapt to changing loads, whereas others are content to make decisions based on long-term averages.

3. Network Protocol

At present, the most widely used network protocol is the connectionless IP (Internet Protocol), which is part of the Internet protocol suite. An IP packet (the technical term for a message in the network layer) can be sent without any setup. Each IP packet is routed to its destination independent of all others. No internal path is selected and remembered.

Transport Protocols

The transport layer forms the last part of what could be called a basic network protocol stack, in the sense that it implements all those services that are not provided at the interface of the network layer, but which are reasonably needed to build network applications. In other words, the transport layer turns the underlying network into something that an application developer can use.

Packets can be lost on the way from the sender to the receiver. Although some applications can handle their own error recovery, others prefer a reliable connection. The job of the transport layer is to provide this service. The idea is that the application layer should be able to deliver a message to the transport layer with the expectation that it will be delivered without loss.

Upon receiving a message from the application layer, the transport layer breaks it into pieces small enough for transmission, assigns each one a sequence number, and then sends them all. The discussion in the transport layer header concerns which packets have

been sent, which have been received, how many more the receiver has room to accept, which should be retransmitted, and similar topics.

Reliable transport connections (which by definition are connection oriented) can be built on top of connection-oriented or connectionless network services. In the former case all the packets will arrive in the correct sequence (if they arrive at all), but in the latter case it is possible for one packet to take a different route and arrive earlier than the packet sent before it. It is up to the transport layer software to put everything back in order to maintain the illusion that a transport connection is like a big tube-you put messages into it and they come out undamaged and in the same order in which they went in. Providing this end-to-end communication behavior is an important aspect of the transport layer.

The Internet transport protocol is called TCP (Transmission Control Protocol) and is described in detail in Comer (2006). The combination TCP IP is now used as a de facto standard for network communication. The Internet protocol suite also supports a connectionless transport protocol called UDP (Universal Datagram Protocol), which is essentially just IP with some minor additions. User programs that do not need a connection-oriented protocol normally use UDP.

High Level Protocols

Above the transport layer, OSI distinguished three additional layers. In practice, only the application layer is ever used. In fact, in the Internet protocol suite, everything above the transport layer is grouped together. In the face of middleware systems, we shall see in this section that neither the OSI nor the Internet approach is really appropriate.

The session layer is essentially an enhanced version of the transport layer. It provides dialog control, to keep track of which party is currently talking, and it provides synchronization facilities. The latter are useful to allow users to insert checkpoints into long transfers, so that in the event of a crash, it is necessary to go back only to the last

checkpoint, rather than all the way back to the beginning. In practice, few applications are interested in the session layer and it is rarely supported. It is not even present in the Internet protocol suite. However, in the context of developing middleware solutions, the concept of a session and its related protocols has turned out to be quite relevant, notably when defining higher-level communication protocols.

Unlike the lower layers, which are concerned with getting the bits from the sender to the receiver reliably and efficiently, the presentation layer is concerned with the meaning of the bits. Most messages do not consist of random bit strings, but more structured information such as people's names, addresses, amounts of money, and so on. In the presentation layer it is possible to define records containing fields like these and then have the sender notify the receiver that a message contains a particular record in a certain format. This makes it easier for machines with different internal representations to communicate with each other.

Middleware Protocols

Middleware is an application that logically lives (mostly) in the application layer, but which contains many general-purpose protocols that warrant their own layers, independent of other, more specific applications. A distinction can be made between high-level communication protocols and protocols for establishing various middleware services. Middleware communication protocols support high-level communication services. Some of the middleware communication protocols could equally well belong in the transport layer, but there may be specific reasons to keep them at a higher level.

Taking this approach to layering leads to a slightly adapted reference model for communication, as shown in Fig. 4-3. Compared to the OSI model, the session and presentation layer have been replaced by a single middleware layer that contains application-independent protocols. These protocols do not belong in the lower layers we

just discussed. The original transport services may also be offered as a middleware service, without being modified. This approach is somewhat analogous to offering UDP at the transport level. Likewise, middleware communication services may include message-passing services comparable to those offered by the transport layer.

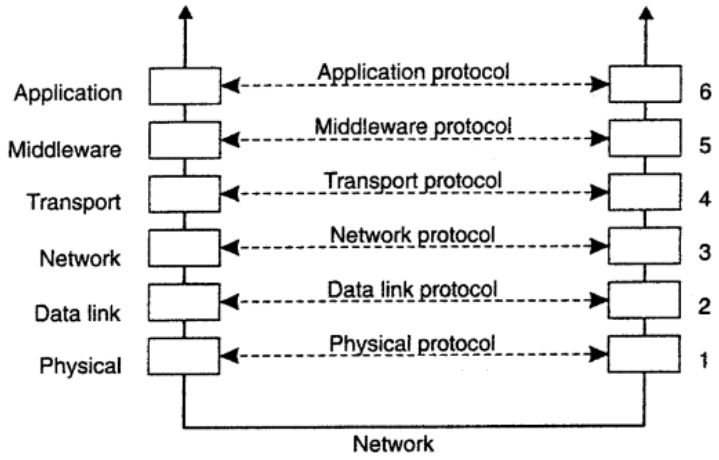

Figure 4-3. An adapted reference model for networked communication.