IP-to-X.25 Protocol Interface Issues

Comparing two communications protocols, with discussion of interface issues and current standards approaches for interface implementation.

by Ramon Caceres

Ramon Caceres is a software development engineer for Pyramid Technology Corporation, 1295 Charleston Road, Mountain View, CA 94039; (415) 965-7200.

The Defense Advanced Research Projects Agency (DARPA) family of standard protocols includes a virtual terminal protocol (TELNET), the File Transfer Protocol (FTP), the Simple Mail Transfer Protocol (SMTP), the Transmission Control Protocol (TCP) and the Internet Protocol (IP). Together they provide a comprehensive set of network services over both local area and wide area networks. As widely accepted and machine-independent standards, they can be used for communication among a variety of computer equipment from many different vendors.

IP, in particular, adds important functionality to the DARPA protocol family. As its name implies, Internet Protocol handles addressing and routing between separate logical and/or physical networks (i.e., an internetwork or internet).

Crucial to this internet role is the IP addressing scheme. IP addresses consist of 32 bits typically divided into two parts: the network address and the host address. Together, the two provide the IP layer of the hosts in an internet with all the information necessary for routing IP packets from their origin to their destination.

Also included in the IP specification is provision for networks whose maximum packet size is smaller than the maximum IP packet size. Through the notion of a maximum transmission unit, IP packages data from higher-level protocols into IP packets of a size known to be acceptable to lower-level protocols and physical networks. This is commonly referred to as IP fragmentation.

IP is a datagram protocol, which means that each unit of data handled by the protocol is treated individually. This is in contrast to stream or virtual circuit protocols, which typically maintain a serial relationship between subsequent units of data for the same physical or logical destination, thereby ensuring both reliable and properly ordered delivery. Implied in the concept of datagrams is the fact that each datagram must by necessity contain enough information to traverse a path from its origin to its final destination. In addition, datagram protocols do not usually ensure reliable or properly ordered data delivery. Naturally, data is not purposefully lost without discretion, but rather, when conditions necessitate it (such as upon heavy network congestion). However, it is important to note that IP will allow data to be dropped and delivered out of order.

The X.25 Protocol

The Consultative Committee for International Telegraph and Telephone (CCITT) X.25 and its related protocols provide a similar set of network services to the DARPA protocols. They too can be used for communication among a wide variety of computer equipment from many different vendors. These protocols include X.25, X.28, X.29 and X.3, among others.

X.25 provides functionality that includes aspects of both TCP and IP. It is a reliable virtual circuit protocol covering the physical, frame and packet level of the International Standards Organization (ISO) Open System Interconnect (OSI) model. In X.25 terminology, these levels are referred to as X.25 Level 1, X.25 Level 2 and X.25 Level 3, respectively.

Similar to IP, X.25 is a host-to-host protocol. As such, X.25 maintains a host addressing scheme obeying the CCITT X.121 standard. X.25 addresses are variable length, encoded two digits per byte as Binary Coded Decimals (BCD). Each address is

IP-to-X.25
CONTINUED

limited to a maximum of 14 BCD digits. This limit is enforced by a 4-bit-address-length field in packets that explicitly contain X.25 addresses. In turn, X.25 addresses are broken up into a 4-BCD-digit Data Network Identification Code (DNIC) and a variable-length host address within that data network, sometimes referred to as the National Terminal Number (NTN).

In contrast to IP, X.25 is a virtual circuit protocol. An important fact to keep in mind regarding virtual circuit protocols is that there is a very definite and unavoidable overhead incurred before data can be transferred to a destination for which a virtual circuit does not already exist. In short, a virtual circuit must be explicitly established between the source and destination entities before any higher-level data can flow. For establishment to take place, virtual circuit protocols usually go through an exchange of control information that constitutes what is usually termed a handshake.

In X.25, the handshake includes the exchange of a call-request packet, and either a call-accept packet — if the called party is willing to go ahead with the transaction — or a clear-request packet, if the called party refuses to go ahead with the transaction. Until a call-request/call-accept handshake is successfully performed, no higher-level data transfer can take place over an X.25 virtual circuit. In X.25 terminology, this process is termed establishing a call.

General Design Issues in Interfacing IP to X.25

Many general design issues need to be resolved when the IP and X.25 protocols are interfaced. While a number of these can be considered purely implementation decisions, several stand out as more general design problems — for instance, the ability to perform packet fragmentation and reassembly, and internetwork routing functionality.

In addition, there are two major design issues to be resolved: IP-to-X.25 address translation and X.25 virtual circuit management. Each is described in more detail below.

IP-to-X.25 Address Translation

Both IP and X.25 maintain their own host-to-host addressing schemes, as described earlier. On outgoing IP packets, the interface must translate IP addresses into X.25 addresses. Similarly, on incoming X.25 call requests, the interface must translate X.25 addresses into IP addresses. There are several ways of performing this two-way translation.

First, the most straightforward means is for the interface to keep a table of IP-to-X.25 address mappings. This is very easy to build and maintain in a general sense, but presents some practical drawbacks. Specifically, in current UNIX systems, the IP protocol is almost always found only in the kernel, while the X.25 protocol is found either in the kernel *or* in the hardware controller. Thus, the most logical implementation of the IP-to-X.25 interface is a piece of kernel software between the IP protocol and the X.25 protocol. In order to maintain reasonable throughput, the IP-to-X.25 address mapping table must also be kept in the kernel where it is easily accessible to the interface. The drawback lies in the fact that the table must be kept to a reasonable size in order to prevent it from using too much of non-pageable kernel memory. This effectively limits the number of hosts that can be reached through the interface to the number of address mappings the system can hold in the kernel address mapping table.

An alternative to the address mapping table approach is to use one of a number of possible two-way mathematical mappings between the IP and X.25 address space. The advantage of the formula approach is that IP-to-X.25 address translation can be performed dynamically in both directions with negligible space consumption. If the formula and the resulting algorithm are simple enough, the computational cost could also be made to approach that of the table lookup operation needed with the address mapping approach, and the gain would be substantial.

X.25 Virtual Circuit Management

Another important issue to resolve when interfacing IP to X.25 is managing the available X.25 virtual circuits to transmit and receive IP datagrams effectively. Central to this issue is the fact that IP is a datagram protocol, while X.25 is a virtual circuit protocol.

Following the nature of datagram protocols, IP will hand the interface a series of outgoing packets that will have no explicit or implied locality of reference with respect to destination hosts. In practice, such locality of reference may very well take place and in fact be a common occurrence, but nevertheless, the interface must be prepared to handle outgoing datagrams for any destination at any time. The interface cannot assume that any sequence of outgoing IP datagrams is destined for any one host or, in fact, for any particular subset of all the hosts in the network.

At the same time, X.25 provides only a limited number of virtual circuits over which to send such datagrams. The number of X.25 virtual circuits available to a host is further limited by the X.25 protocol implementations on both the host and

network sides of the X.25 connection. This number of real simultaneous X.25 virtual circuits typically ranges from 32 to 128 — certainly smaller than the number of hosts in a large internet and much smaller than the number of hosts addressable by IP and X.25.

Given the fact that X.25 virtual circuits must be explicitly established to a particular host before any data can be sent to that host (a non-trivial operation in time and in some cases money), X.25 virtual circuits must be considered a very scarce resource in the context of an IP-to-X.25 interface. It follows that effectively managing this resource plays a crucial role in the behavior of the interface.

The problem of managing a scarce resource to satisfy a much greater or theoretically infinite demand is common to many aspects of operating system design. It is then reasonable to attempt to draw from past experience in other areas to solve the specific case of X.25 virtual circuits and IP datagrams. In particular, this problem is in many ways similar to that of managing a limited amount of real memory to satisfy a much larger demand for virtual memory. It can be expected that some of the more successful schemes for virtual memory management would also work well here. Describing the many virtual memory schemes that have been developed is outside the scope of this article, but suffice it to say that schemes such as preemptive Least Recently Used (LRU) algorithms and the working set concept apply very well to X.25 virtual circuit management in an IP-to-X.25 interface.

Aside from the consideration of which algorithm to use to schedule the number of available X.25 virtual circuits among the number of destination hosts referenced to IP, a related decision is how many X.25 virtual circuits to open per destination host. Because an unavoidable delay is incurred between the time one particular X.25 virtual circuit accepts a packet for output and the time it can accept another one, it is logical to assume that opening more than one virtual circuit per destination host will help improve throughput. In particular, the diminutive window sizes used by X.25 networks to ensure reliable transmission tend to prevent a single virtual circuit from approaching data rates anywhere close to the X.25 line speed.

Specifying Networking Standards

There are a number of other, more specific choices that must be addressed when interfacing the IP and X.25 protocols. The freedom in many of these areas is such that different IP-to-X.25 interfaces must agree exactly on many issues if they are to communicate.

For this reason, as is true with the protocols they serve to join, standards are necessary for different implementations of IP-to-X.25 interfaces to interact in a reasonable fashion. Two such standards have developed for two different types of X.25 networks — one for the X.25 portion of the Defense Data Network (DDN) and the other for X.25 Public Data Networks (PDNs). They agree in many areas, but differ in their approach to some of the general design issues described earlier. They also differ in their support of several features specific to either the DDN or particular PDNs.

This article is excerpted from "The Pyramid IP to X.25 Protocol Interface: Merging DDN and PDN Approaches," the author's technical paper presented at UniForum 1987.

The DDN Approach

The Defense Communications Agency (DCA) has decreed that all new host connections to the DDN be made with X.25, and not with the more traditional 1822 ARPANET protocol. Indeed, X.25 is slated to completely substitute 1822 ARPANET in the DDN. Furthermore, if a host wants to interact with other DDN hosts using the full functionality available in the DDN, then it must include an IP-to-X.25 interface.

To aid in this purpose, the DCA has published a definitive standards document, the DDN X.25 Host Interface Specification. This specification dramatically narrows the number of implementation decisions faced by developers of IP-to-X.25 interfaces for the DDN.

In the case of address translation, as with other issues, the standard allows different implementations to interact successfully by specifying exactly the actions to be taken by all IP-to-X.25 interfaces in the DDN. For IP-to-X.25 address translation, hosts in the DDN must use two simple translation formulas that directly map certain fields in the IP address to other fields in the X.25 address, and vice versa.

On the issue of how many virtual circuits are opened between each host pair, only one X.25 virtual circuit is opened between any two hosts on the DDN. This certainly simplifies things, although it

IP-to-X.25
CONTINUED

eliminates the possibility of added throughput between any pair of hosts through the use of multiple virtual circuits. However, as DDN X.25 links can operate near the upper limits of packet sizes (1,024 bytes per packet), window sizes (7 unacknowledged packets) and transmission speeds (56Kb per second) available in X.25 networks, the cost incurred is minimized.

Regarding internetwork routing, the DDN makes full use of the internetworking features of the IP protocol. This relieves the interface from having to resolve addresses for different physical networks in order to route packets among these networks.

Aside from the generic design issues explained earlier, there are several network-specific features that an IP-to-X.25 interface to the DDN must include. One of the most visible examples is the specification of DDN standard service. X.25 hosts on the DDN can subscribe to either basic or standard service. DDN basic service can be thought of as raw X.25 service provided by the network for host-to-host communication on one physical X.25 network. Basic service does not imply the use of an IP-to-X.25 interface to access the network. More relevant to this discussion, DDN standard service is oriented toward DDN X.25 hosts using the standard TCP/IP higher-level protocols. IP-to-X.25 interfaces on the DDN must clearly specify DDN standard service by means of a private or non-CCITT facility included in all outgoing X.25 call request packets. Similarly, these interfaces should expect this facility on incoming X.25 call requests. Other network-specific features are similarly made available through the use of non-CCITT facilities (for example, DDN call precedence).

The PDN Approach

In the case of Public Data Networks, a detailed standards document in the fashion of the DDN X.25 Host Interface Specification is not available. Rather, there is a Request for Comments (RFC) from the DDN Network Information Center (NIC) — RFC 877, "A Standard for the Transmission of IP Datagrams Over Public Data Networks." In addition, an excellent complement to this document is the source code release of Purdue University's X.25 Network Interface (XNI) software for the X25net portion of the Computer Science Research Network (CSNET); the software is described in the March 1983 issue of the ACM SIGCOMM's "Proceedings of the Symposium on Data Communications." This software, which is available to CSNET members directly from the CSNET Coordination and Information Center, can be considered a de facto standard for interfacing IP to X.25 for operation over a PDN.

The address translation formulas used in the DDN are convenient and efficient, but their use assumes that the network administration entity has complete control of both IP and X.25 address spaces. This is true in the DDN where a central organization, the NIC, is responsible for assigning both IP and X.25 addresses to all DDN X.25 sites. They can simply assign any available IP address and then use the IP-to-X.25 formula to assign a corresponding X.25 address, or vice versa.

Unfortunately, in the PDN case, the network administration is not so centralized. Typically, the X.25 address space is controlled by the PDN administrating agency, but the IP address space is still controlled by the NIC. In the case of a logical network like CSNET, the CSNET CIC does not control either address space. They are forced to rely on the PDN carrier for assignment of X.25 addresses, and on the NIC for assignment of IP addresses. Because of these practical limitations, IP-to-X.25 interfaces for PDNs must use an alternate method to the formula approach.

As described earlier (under the "General Design Issues in Interfacing IP to X.25" heading), using an address mapping table is a simple and effective way of performing IP-to-X.25 address translation. Such a table is used by the CSNET XNI software. This table contains one entry for each host that the interface needs to communicate with, including the local host. Each entry in the table contains a host name, its IP address, its X.25 address and other PDN-specific information for the host. IP-to-X.25 address translation is then possible by performing a table lookup operation.

On the issue of how many virtual circuits are opened between each host pair, RFC 877 and the CSNET XNI software allow for several X.25 virtual circuits to exist between a pair of hosts. This makes possible added throughput between any pair of hosts by demultiplexing data over multiple virtual circuits. This is a great advantage in the PDN context, where the packet and window sizes (128 bytes per packet and 2 unacknowledged packets, respectively), and transmission speeds (9.6Kb per second) used are typically small enough to cause significant

delays in IP datagram transmission. The number of virtual circuits to open to a remote host is specified in the address mapping table entry for that host.

Because of practical network administration limitations similar to those encountered with IP-to-X.25 address mapping, the CSNET XNI software is forced to take on many of the internetwork routing responsibilities usually associated with IP in order to operate successfully in an internetwork environment. The interface must resolve addresses for different physical networks in order to route packets among these networks and maintain enough information within itself to make proper use of any available gateways. The IP-to-X.25 interface in the gateway hosts must also take on additional routing responsibilities because IP is not being used for this purpose.

Aside from the generic design issues explained above, there are several network-specific features that an IP-to-X.25 interface to PDNs must include. An important example is the specification of reverse charging — collect calls. Each X.25 call request can be made using either direct charging or reverse charging. If desired, IP-to-X.25 interfaces on PDNs can specify reverse charging by means of a standard CCITT facility included in outgoing X.25 call request packets. Similarly, these interfaces should be prepared to handle this facility on incoming X.25 call requests from other hosts. Other network-specific features are made available through the use of facilities, both standard and non-standard. Examples of standard facilities are packet size, window size and throughput class negotiations. Examples of non-standard facilities are non-CCITT packet size and window size negotiations still used in some networks as historical remains of implementations that follow pre-1980 CCITT recommendations.

In Conclusion

The number of different networks now accessible with the DARPA family of protocols is considerably larger than the traditional set of Ethernet and ARPANET. Of these network types, one that promises to become increasingly important is X.25 networks. More specifically, connecting the DARPA family of protocols to X.25 networks involves creating an interface between the IP and X.25 protocols.

Designing such an interface involves solving many generic design problems, many of which have already been resolved for the implementor by one or both of the DDN and PDN standards described above. Pyramid Technology Corp. recently implemented this interface for its dualPort OSx operating system, extending its DARPA-style networking capabilities to the X.25 portion of the DDN, as well as to a number of X.25 PDNs. ■

The technical paper from which this article is excerpted covers some of the more interesting design and implementation issues encountered while developing an IP-to-X.25 interface that complies with both DDN and PDN requirements. A reprint of that paper is available from /usr/group, 4655 Old Ironsides Dr., #200, Santa Clara, CA 95054. In addition, the complete Conference Proceedings document covering UniForum 1987, where the technical paper was presented, can be purchased from /usr/group.

For a reprint of the complete technical paper from which this article is excerpted, contact /usr/group, 4655 Old Ironsides Dr., #200, Santa Clara, CA 95054.