Xerox Network Systems (XNS)

Background

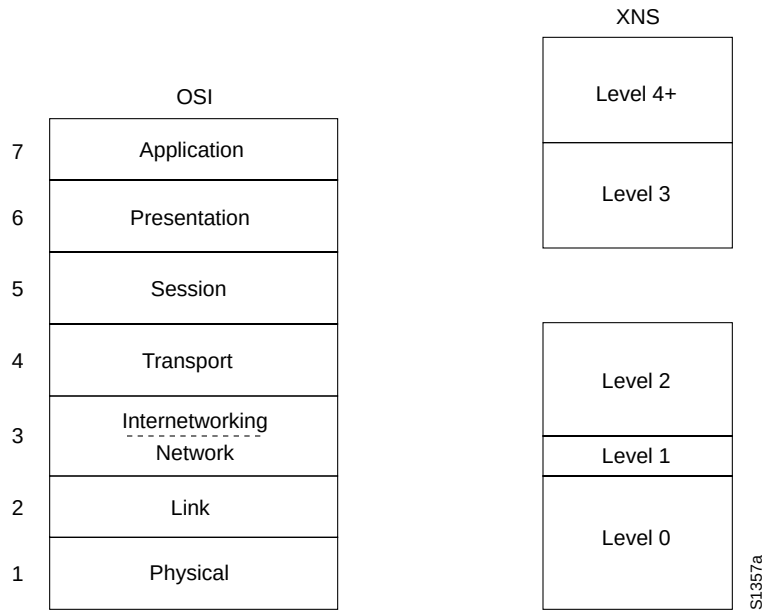
The Xerox Network Systems (XNS) protocols were created by Xerox Corporation in the late 1970s and early 1980s. They were designed to be used across a variety of communication media, processors, and office applications. Several XNS protocols resemble the *Internet Protocol* (IP*)* and Transmission- *Control Protocol* (TCP) entities developed by the Defense Advanced Research Projects Agency (DARPA) for the U.S. Department of Defense (DoD).

Because of its availability and early entry into the market, XNS was adopted by most of the early LAN companies, including Novell, Inc., Ungermann-Bass, Inc. (now a part of Tandem Computers), and 3Com Corporation. Each of these companies has since made various changes to the XNS protocols. Novell added the *Service Advertisement Protocol* (SAP) to permit resource advertisement and modified the OSI Layer 3 protocols (which Novell renamed *IPX*, for *Internetwork Packet Exchange*) to run on IEEE 802.3 rather than Ethernet networks. Ungermann-Bass modified RIP to support delay as well as hop count and made other small changes. Over time, the XNS implementations for PC networking have become more popular than XNS as it was designed by Xerox. This chapter presents a summary of the XNS protocol stack in the context of the OSI reference model.

XNS Hierarchy Overview

Although the XNS design objectives are the same as those for the OSI reference model, the XNS concept of a protocol hierarchy is somewhat different from that provided by the OSI reference model, as Figure 34-1 illustrates.

Figure 34-1 Xerox adopted a five-layer model of packet communication.



As illustrated in Figure 34-1, Xerox provided a five-level model of packet communications. Level 0 corresponds roughly to OSI Layers 1 and 2, handling link access and bit-stream manipulation. Level 1 corresponds roughly to the portion of OSI Layer 3 that pertains to network traffic. Level 2 corresponds to the portion of OSI Layer 3 that pertains to internetwork routing, and to OSI Layer 4, which handles interprocess communication. Levels 3 and 4 correspond roughly to the upper two layers of the OSI model, handling data structuring, process-to-process interaction and applications. XNS has no protocol corresponding to OSI Layer 5 (the session layer).

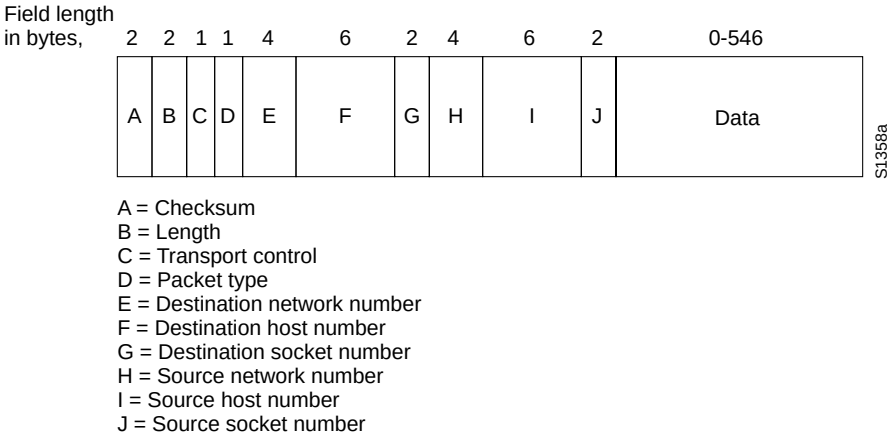
Media Access

Although XNS documentation mentions X.25, Ethernet, and High-Level Data Link Control (HDLC), XNS does not expressly define what it refers to as a Level 0 protocol. As with many other protocol suites, XNS leaves media access an open issue, implicitly allowing any such protocol to host the transport of XNS packets over a physical medium.

Network Layer

The XNS network-layer protocol is called the *Internet Datagram Protocol* (IDP). IDP performs standard Layer 3 functions, including logical addressing and end-to-end datagram delivery across an internetwork. Figure 34-2 illustrates the format of an IDP packet.

Figure 34-2 Eleven fields comprise an IDP packet.



The following descriptions summarize the IDP packet fields illustrated in Figure 34-2:

- *Checksum*—A 16-bit field that helps gauge the integrity of the packet after it traverses the internetwork.
- *Length*—A 16-bit field that carries the complete length (including the checksum) of the current datagram.
- *Transport Control*—An 8-bit field that contains the *hop count* and *maximum packet lifetime* (MPL) subfields. The hop- count subfield is initialized to zero by the source and incremented by one as the datagram passes through a router. When the hop- count field reaches 16, the datagram is discarded on the assumption that a routing loop is occurring. The MPL subfield provides the maximum amount of time, in seconds, that a packet can remain on the internetwork.
- *Packet Type*—An 8-bit field that specifies the format of the data field.
- *Destination Network Number*—A 32-bit field that uniquely identifies the destination network in an internetwork.
- *Destination Host Number*—A 48-bit field that uniquely identifies the destination host.
- *Destination Socket Number*—A 16-bit field that uniquely identifies a socket (process) within the destination host.
- *Source Network Number*—A 32-bit field that uniquely identifies the source network in an internetwork.
- *Source Host Number*—A 48-bit field that uniquely identifies the source host.
- *Source Socket Number*—A16-bit field that uniquely identifies a socket (process) within the source host.

IEEE 802 addresses are equivalent to host numbers, so that hosts that are connected to more than one IEEE 802 network have the same address on each segment. This makes network numbers redundant but nevertheless useful for routing. Certain socket numbers are *well known*, which means that the service performed by the software using them is statically defined. All other socket numbers are reusable.

XNS supports Ethernet Version 2.0 encapsulation for Ethernet, and three types of encapsulation for Token Ring: 3Com, SubNet Access Protocol (SNAP), and Ungermann-Bass.

XNS supports *unicast* (point-to-point), *multicast*, and *broadcast* packets. Multicast and broadcast addresses are further divided into *directed* and *global* types. Directed multicasts deliver packets to members of the multicast group on the network specified in the destination multicast network address. Directed broadcasts deliver packets to all members of a specified network. Global multicasts deliver packets to all members of the group within the entire internetwork, whereas global broadcasts deliver packets to all internetwork addresses. One bit in the host number indicates a single versus a multicast address. Conversely, all ones in the host field indicate a broadcast address.

To route packets in an internetwork, XNS uses the RIP dynamic routing scheme. Today, RIP is the most commonly used *Interior Gateway Protocol* (IGP) in the Internet community. For more information about RIP, see Chapter 44, "Routing Information Protocol (RIP)."

Transport Layer

OSI transport-layer functions are implemented by several protocols. Each of the following protocols is described in the XNS specification as a Level 2 protocol.

The *Sequenced Packet Protocol* (SPP) provides reliable, connection-based, flow-controlled packet transmission on behalf of client processes. It is similar in function to the Internet Protocol suite's *Transmission-Control Protocol* (TCP) and the OSI protocol suite's *Transport Protocol 4* (TP4). For more information about TCP, see Chapter 30, "Internet Protocols." For more information about TP4, see Chapter 32, "Open System Interconnection (OSI) Protocols."

Each SPP packet includes a *sequence number*, which is used to order packets and to determine whether any have been duplicated or missed. SPP packets also contain two 16-bit *connection identifiers*. One connection identifier is specified by each end of the connection, and together, the two connection identifiers uniquely identify a logical connection between client processes.

SPP packets cannot be longer than 576 bytes. Client processes can negotiate use of a different packet size during connection establishment, but SPP does not define the nature of this negotiation.

The *Packet Exchange Protocol* (PEP) is a request-response protocol designed to have greater reliability than simple datagram service (as provided by IDP, for example) but less reliability than SPP. PEP is functionally similar to the Internet Protocol suite's *User Datagram Protocol* (UDP). For more information about UDP, see Chapter 30, "Internet Protocols." PEP is single-packet based, providing retransmissions but no duplicate-packet detection. As such, it is useful in applications where request-response transactions can be repeated without damaging data, or where reliable transfer is executed at another layer.

The *Error Protocol* (EP) can be used by any client process to notify another client process that a network error has occurred. This protocol is used, for example, in situations where an SPP implementation has identified a duplicate packet.

Upper-Layer Protocols

XNS offers several upper-layer protocols. The *Printing Protocol* provides print services, the *.Filing Protocol* provides file-access services, and the. *Clearinghouse Protocol* provides name services. Each of these three protocols runs on top of the *Courier Protocol*, which provides conventions for data structuring and process interaction.

XNS also defines Level 4 protocols, which are application protocols, but because they have little to do with actual communication functions, the XNS specification does not include any pertinent definitions.

The Level 2 *Echo Protocol* is used to test the reachability of XNS network nodes and to support functions such as that provided by the **ping** command found in UNIX and other environments.