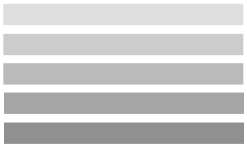

PART FIVE

THE FUTURE OF OPEN SYSTEMS NETWORKING

16 MULTIPROTOCOL OPEN SYSTEMS

The world of open systems networking does not look the way industry pundits predicted it would less than a decade ago. The great open-system sirens of the eighties—homogeneity, uniformity, transition—are now largely silent; the new age of networking must embrace and accommodate what appears to be a permanently heterogeneous mix of technologies and standards. Having examined OSI and TCP/IP side by side and in some detail, it is instructive for us to discuss their present status in the real world and to speculate on their future.

The Myth of "OSI Migration"

By the mid-1980s, OSI was widely heralded as the gospel of a "new age" networking religion that would use the power of international standards to achieve for data networks the same ubiquitous, multivendor connectivity that is enjoyed by users of the international telephone network. Even proprietary network architectures that explicitly eschew "openness" in favor of "optimal performance" began applying the language of the OSI reference model to describe the way in which their protocols and services are organized (a favorite claim across the industry at that time was, "Our architecture conforms to the OSI reference model"). Industry pundits of the eighties predicted that OSI would eventually dominate the world of open systems networking, and the widely accepted inevitability of OSI at that time forced the issue of network evolution into a narrow box: specifically, how to manage a "transition," or "migration," from whatever one currently had to OSI.

Just as the evolution of a species is influenced by external conditions—climate, in particular—network evolution toward a single open systems networking technology has been affected by the climate of the information-processing marketplace, which has too many networking choices. TCP/IP, DECnet, IPX, SNA, and AppleTalk are "durable, if incompatible, and they just won't die peacefully."[1] At the same time that OSI was being touted as the networking technology of the future, network administrators were dealing with a major change in culture, as society embraced electronic mail, distributed file service, and LAN operating systems as part of its daily routine: networks were growing and the demands of users were changing, and a wholesale shift to a new technology was not a high priority in the field. Vendors seized the opportunity to extend the life cycle of existing technology rather than invest in a new one, and the standards process provided them with a convenient excuse: "critical-path" standards—notably, routing and management—were not yet completed, so many vendors decided to wait for closure before tackling OSI. Thus, the real networks of the world are running the bulk of their traffic over something other than OSI—SNA, DECnet, TCP/IP, or one of a host of LAN protocols (often referred to as "LAN operating systems").

Many real networks are running something best described as "all of the above and then some." In terms of deployment, the market presence of TCP/IP dwarfs that of OSI and is likely to continue growing at a rapid pace for many years, no matter how many GOSIPs (government OSI procurement regulations) are adopted by national governments. TCP/IP is now widely available and understood; it works; and it has benefited enormously from decades of steady improvement in both the protocols and their implementations. And it has carefully attended to the protection of the installed base: more often than not, changes are introduced only after lengthy evaluation of how those changes will affect TCP/IP networks already in operation.

The assertion that TCP/IP is fine for universities but will never make it in the commercial world—a myopic OSI bigot once called the Internet an "academic toy"—has been proved wrong: in 1990, 56 percent of the sales of TCP/IP-based networking equipment and software were in the traditional commercial market, and this figure is projected to rise to 77 percent by the end of 1993. It is important to recognize, too, that the

1. A quote fron Dan Lynch, founder and chairman of Interop, Inc., and former member of the ARPANET team that assisted in the transition from NCP protocols to TCP/IP in the early 1980s, found in *Update: Interop '92 Spring*. If anyone would know about the difficulties of transitioning a network, it would be Dan.

strengths of OSI are for the most part also the strengths of TCP/IP; from a technical standpoint, the rationale for deployment of TCP/IP networks is no less compelling than the rationale for deployment of OSI networks. TCP/IP developers who recognize that there are useful lessons to be learned from the evolution of OSI (particularly in the upper layers) will be building and selling TCP/IP networks long after OSI has established itself in the marketplace.

It is also important to recognize that the TCP/IP marketplace itself is dwarfed by the cumulative share held by LAN operating systems such as Novell Netware and Banyan Vines and that proprietary networking solutions continue to hold a formidable share as well. These alternatives won't die peacefully either. They play an important role in mainstream distributed processing and will continue to do so despite bureaucratic efforts to the contrary.

OSI Is an Alternative, Not a Substitute

As a practical matter, perfect network homogeneity is no longer viewed as attainable or necessarily desirable. Marshall Rose's "D-day conjecture" was anything but farfetched: TCP/IP-based networks already offer OSI application services alongside TCP applications (Rose 1990, 551). The change in climatic conditions now forces the issue of network evolution onto an open field: the truly interesting issue is not how to provoke every conceivable network architecture through a "transition to OSI" but how to accommodate heterogeneity within a *multiprotocol* network architecture. OSI is no longer the only alternative; it is one among many, and it will have to survive by the virtue of its own strengths.

So what *are* OSI's strengths? Although it is often most harshly attacked, one strength of OSI is its status as an international standard. This is of relatively small importance to a considerable proportion of the Internet community which tends to exclude the politics and diplomacy of international networking from its list of valid concerns; however, for others, the imperatives driving the international acceptance of OSI were —and remain—impressive. OSI standards represent a powerful international alignment of computer vendor and telecommunication carrier interests in market stability and predictability, augmented by the emergence of GOSIPs, which may ultimately have the political effect of making OSI the norm, rather than the exception, in many large contract bids. Whether networking technology should be determined by political fiat is not at issue here; it is simply a fact that certain organizations will (per-

haps dogmatically) accept OSI because they feel they must. It should also be noted that a lucrative marketplace for OSI continues to exist among companies that believe that OSI is a strategic imperative—for example, telecommunications equipment and service vendors, which constitute a huge market for OSI to support public directory and message-handling services as well as internal "operations" (management services).

Another of OSI's strengths is its applications infrastructure. From a technical standpoint, the elaborate OSI upper-layer architecture—often maligned as unnecessarily complex and a performance-stifling burden—will be increasingly important as sophisticated distributed-processing applications enter the computing mainstream. Much of this is in evidence today, as the Internet community experiments with the OSI directory and message-handling applications over ISODE, RFC 1006, and TCP/IP. The OSI Directory has become an increasingly important tool to the Internet; efforts to improve the response time for directory queries, complemented by efforts to reduce the overhead of directory user agents so that they can run easily on desktop computers, will undoubtedly make the OSI Directory a finer tool.

Because this book discusses OSI and TCP/IP together, it is inevitable that the future of OSI will be highly colored with the perspective of "the future of OSI in the Internet." But advocacy for OSI message-handling and directory services exists outside the Internet community to an even greater extent than within. It is instructive to note that corporate internets that have little or no connectivity to the Internet (due, in some cases, to a conscious effort to ensure complete privacy and security) and that have, in some cases, little or no TCP/IP (especially environments in which PC and mainframe OSs are pervasive and UNIX is virtually nonexistent) see OSI message-handling and directory services as providing a welcome reprieve from the proliferation of PC, proprietary, and homebrew mail gateways and directory services. Not surprisingly, the absence of a single, dominant mail system like the Internet's SMTP and a universal name system like the DNS in these environments has played to the advantage of OSI.

OSI's lower layers are also often treated with disdain, primarily because of the exaggerated "connections versus connectionless" interworking problem, but also because of performance. Both of these criticisms will disappear over time. OSI's strength in the lower layers is that in architecture and implementation, it is very similar to TCP/IP. A very positive side effect of GOSIPs is that they have forced system designers, implementers, and users to build and use "first-generation" OSI products, if only to satisfy a checklist item on a "request for proposal" from a

government agency. As a consequence of being coerced into implementing or, worse still, *using* OSI, implementers have begun to realize that the widely advertised problems of performance and resource utilization are not due to some inherently threatening "OSI baggage"; instead, they discover that OSI's lower layers can benefit from the experience and engineering expertise that have come gradually with TCP/IP. It is easy to forget that TCP/IP as it stands in 1993 is the result of nearly two decades of improvement and refinement. And as with TCP/IP, experiments to trim fat from OSI will happen only after the fat has been distinguished from the muscle. The process of improving OSI performance will accelerate as the expertise acquired by TCP/IP engineers is applied to the implementation and refinement of OSI protocols—assuming, of course, that the TCP/IP engineers continue to overcome their initial disdain for OSI and that the OSI developers pay attention to what the TCP/IP engineers have to say.

It is a small matter of time and experience before improvements to OSI's underlying transport service and solutions to the "connectionless versus connection-oriented" interworking problem make these nonissues as well. Implementations of OSI's transport class 4 have already been improved by the use of slow-start algorithms, dynamic window-adjustment mechanisms, and other forms of congestion avoidance and control that are commonplace in TCP implementations. Implementers are also beginning to understand that in some cases—notably, the issue of efficiency in checksums—"apples are not being compared to apples"; the checksum used in TCP is faster, but the one used in TP4 is stronger.

The "connections versus connectionless" debate may also end soon. Although ISO has developed and adopted a technical report describing a recommended means of interworking between connection-oriented and connectionless networks (ISO/IEC TR 10172: 1991), and the very fine work of folks like Jeremy Onions has provided us with useful transport service bridges, the authors speculate that attention to the "Tinkertoy transports and X.25" will wane as the decade closes, even among PTTs and public network providers, and that the application of TP4 over ISO CLNP as *the recommended* underlying transport service will be widespread. This speculation is based partly on the authors' conviction that this is technically the right choice, also on the steadily increasing interest in IP and connectionless metropolitan-area network services throughout Europe and North America.

In the minds of some, OSI will never be acceptable, in any form; and although it is unlikely ever to be the universal panacea for networking, it most certainly will do useful work for a sizable population. In the

Internet, it coexists alongside TCP/IP today and will continue to do so for the forseeable future.

OSI and TCP/IP Coexistence: Networking Détente

If integration, rather than migration, has become the issue, the question remains "how to make it happen." To date, several alternatives have been deployed with varying degrees of success. Some of these were introduced and discussed in Marshall Rose's *Open Book* in the context of "transition to OSI"; here, the authors examine these alternatives with the luxury of hindsight, considering whether and how each has been applied or extended and also placing the discussion in the context of "coping with a multiprotocol environment."

Dual and "Multiple" Stacks

The simplest solution to the problem of multiple protocol architectures in the Internet is to run each protocol "stack" independently in each system. Figure 16.1 illustrates how OSI and TCP/IP can be deployed as "dual stacks," but the principle applies for multiple stacks as well. In the dual-stack approach, OSI and TCP/IP applications and protocols coexist but do not interoperate—although they may be engineered to share the avail-

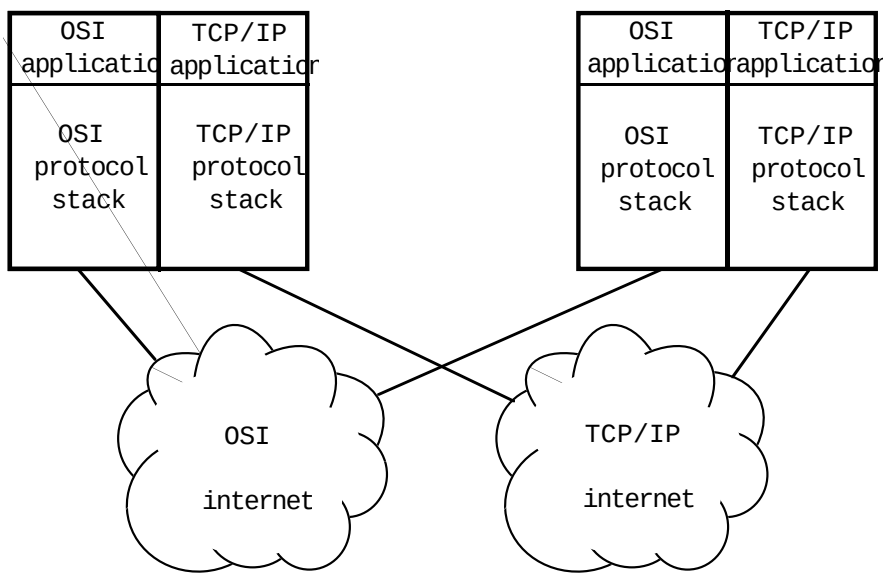

FIGURE 16.1 The Dual-Stack Approach

able underlying physical transmission resources (link drivers, repeaters, and cables) and switching resources (bridges and multiprotocol routers) when economies result from their doing so.

The dual-stack approach is sometimes called *ships in the night*, or S.I.N. The metaphor of ships "passing in the night"—independently and without interaction—captures the sense of multiple protocol suites communicating over the same network links, ostensibly without interference, in much the same way as independent telephone conversations can be carried over a common set of telephone wires and switches. It should be noted that multiple protocol stacks have operated over LAN technologies such as Ethernet for years: with the advent of multiprotocol routers, the multistack approach has become nearly ubiquitous in the Internet today. The NSFnet has been capable of switching both CLNP and IP datagrams since 1990. Many regional, national agency, international, and connected ("stub") subnetworks now offer CLNP as well as IP connectivity to both single- and dual-stack hosts. (Figure 16.2 depicts the approximate CLNP connectivity across the Internet as of March 1993.) Member sites in these networks are piloting, experimenting with, or have in production one or more OSI application services, including file transfer (FTAM), virtual terminal, electronic mail (X.400), and directories (X.500).

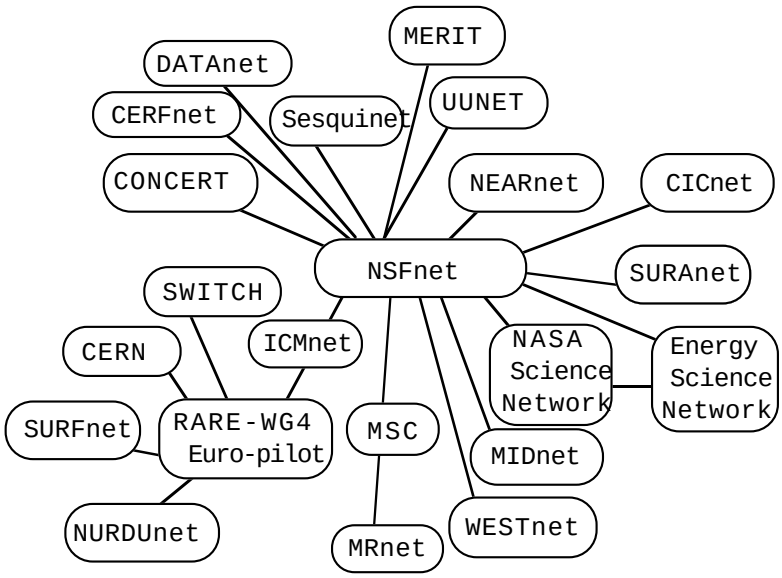

FIGURE 16.2 Internet CLNP Connectivity ("Stub" Networks Not Illustrated)

Dual (multiple) stacks are often the only way to accommodate more than one protocol suite within a single system. However, the drawbacks discussed in the *Open Book* remain in evidence today:

- The dual-stack (multiple-stack) approach is *expensive*, since it requires "two (many) of everything": network management, software, protocol overhead, on-call systems-engineering expertise, and administration of addresses, operation, software and equipment updates, and access control. This solution is expensive in the traditional interpretation as well, as it often requires more CPU, more memory, and more storage to operate.
- It *affects all systems*. Hosts, in particular, must have two (many) of everything, and where multiprotocol switching systems (routers, bridges) cannot be deployed, parts of the infrastructure must be duplicated (multiplied).
- It often affects *performance*. Absent a perfect "memory and CPU are cheap" utopia, running two or more protocol stacks costs.
- It is difficult to achieve transparency. The UNIX *socket()* interface provides access to both OSI and TCP/IP, but the services are distinct. The same is true for the X/Open Transport Interface (XTI) and various application program interfaces (APIs); transparency is an elusive target.
- Unless all systems have dual (multiple) stacks, there will not be complete interoperability.

Unless the protocol suites are implemented and deployed very carefully, it may not be possible to avoid interference. When both protocol suites are operated over the same transmission facilities, for example, and both do not subscribe to the same philosophy of congestion avoidance, the effectiveness of such feedback mechanisms as explicit "congestion-experienced bits" is compromised. Even when it is possible to prevent direct interference between two different protocol suites, the metaphorical ships cannot pass obliviously, as they must be subject to at least some form of common system-management and resource-utilization arbitration.

Application Gateways

The dual-stack (multiple-stack) approach typically accommodates diversity among protocol suites everywhere except at the very bottom of the stacks, where common physical interconnections among systems can often be shared (this includes common data-link layer framing as well). Although this permits different protocol suites to share transmission resources (with the constraints and compromises noted earlier), it does not provide for any interaction among them—although systems running

different stacks may be able to communicate at the level of their common links, their applications cannot interoperate. This adversely affects end users who are unfortunate (or wise) enough to have a single-stack host, since they cannot run the same distributed applications everywhere and to everyone; it also affects those fortunate enough to have a dual-stack host, because they must distinguish the end users reachable over one stack from those reachable over the other. Rose (1990) and others describe a sort of meta-application to deal with this sticky business—i.e., to hide the decision-making uglies from the end user—but this involves even more software (a directory service or a local cache-inquiry mechanism) and administration.

Adding an "application gateway" to a dual-stack configuration addresses the problem of application interoperability by establishing well-defined (and accessible) points at which the intrinsic semantics of an application—electronic mail, for example, or file transfer—can be extracted from the protocols of one stack and translated into the corresponding protocols of another stack (see Figure 16.3). This approach permits applications within each protocol suite to interoperate homogeneously with applications of their own kind (e.g., X.400 [1984] MHS with X.400 [1988] MHS), yet also bridges the gap to a foreign protocol suite by passing through a gateway that preserves the essential semantics of the application (e.g., electronic mail) while converting or translating the application protocols into alternatives that are compatible with the foreign protocol suite (e.g., Internet mail *à la* SMTP). This is a widely practiced

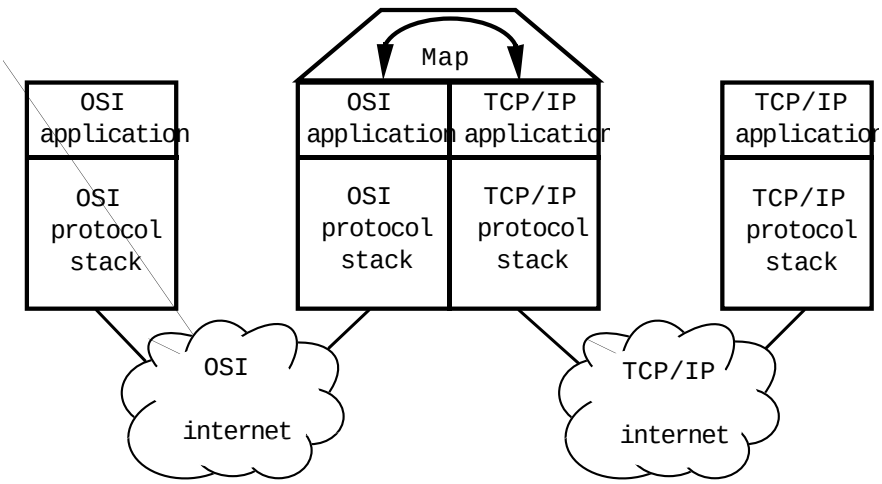

FIGURE 16.3 The Application Gateway Approach

method of interconnecting users in the Internet today.

Application gateways are natural for some applications, such as store-and-forward electronic mail, that are designed to pass through application-relay points in the normal course of events even in a homogeneous environment. The results are much less satisfying for other distributed applications—networked operating systems, file systems, distributed database systems—where the paradigm for such distributed systems may differ radically: an application gateway situated between clients and servers of AT&T's Remote File Service and Sun's Network File System is decidedly more complicated than one involving X.400–to–RFC 822 mail; the entire paradigm for a distributed filestore differs more than one would ever hope to reconcile in a gateway.

Application gateways also permit network managers to establish and monitor a strong administrative boundary between a local application environment (the campus mail network, for example) and the rest of the Internet; this is sometimes referred to as "outside-world contraception." And in some cases, of course, an application gateway is the only way to glue two otherwise oblivious environments together without changing an operating system software in which the lower-layer protocol implementations are embedded.

The unavoidable "least-common-denominator" effect of an application gateway is its worst drawback; users whose electronic mail traverses an X.400-1988–to–RFC 822 mail gateway, for example, may only be able to send transparent ASCII text in the interpersonal message (and similarly, MIME users to X.400-1984 MHS). Users will generally have to be keenly aware of the presence of the gateway (especially true when one is dealing with electronic mail addresses), and conduct their communications accordingly, in order to achieve success. Every type of application, of course, requires its own special type of application gateway, and the problem is exacerbated each time another stack is introduced—a separate gateway or extension is then required for every pair of *like* applications (an X.400 MHS–to–RFC 822 mail gateway is a separate ordeal from an SMTP to PC LAN-based mail/messaging gateway); this is exactly the situation in which many corporate internets having no "core" TCP/IP service find themselves.

Because the gateway must perform a fairly complicated transformation on the data flowing through it, it often represents a performance bottleneck (for some applications, such as electronic mail, the performance degradation is unobtrusive). Users must ensure that only one application gateway intervenes between the source and destination systems, since it is generally true that application gateways do not chain well. If an

FTAM-to-FTP gateway is used to transfer a file, for example, and the original FTAM virtual filestore was other than one of the file structure/data representations handled by FTP, a gateway will make a best effort at matching the virtual filestore, but without altering FTP, some information about the original filestore will be lost. If that file passes through a (subsequent) FTP-to-FTAM gateway, it won't be possible to reconstruct the file in the original filestore, and so despite the fact that the file originated from and was delivered to an FTAM application, the original file will differ from the one delivered.

Finally, the proliferation of application gateways tends to carve up the Internet into xenophobic nation-states with heavily guarded border crossings and long lines at customs and immigration—quite different from the "universal connectivity" goal of most network designers.

Transport Service Bridge

It is, of course, not strictly necessary to go all the way up the protocol stack to the application before establishing a point at which translation from the protocols of one suite into those of another can be made. In the "transport service bridge" approach, the end-to-end characteristics of the transport service of a given protocol suite (e.g., the OSI transport service) are provided using the transport protocol of another protocol suite (e.g., TCP [RFC 1006). (See Figure 16.4 as well as Chapter 12.) End users may then use the same applications despite having different underlying transports.

The appeal of the transport service bridge approach lies in the fact that it requires no changes to host (end-system) software—as long as all

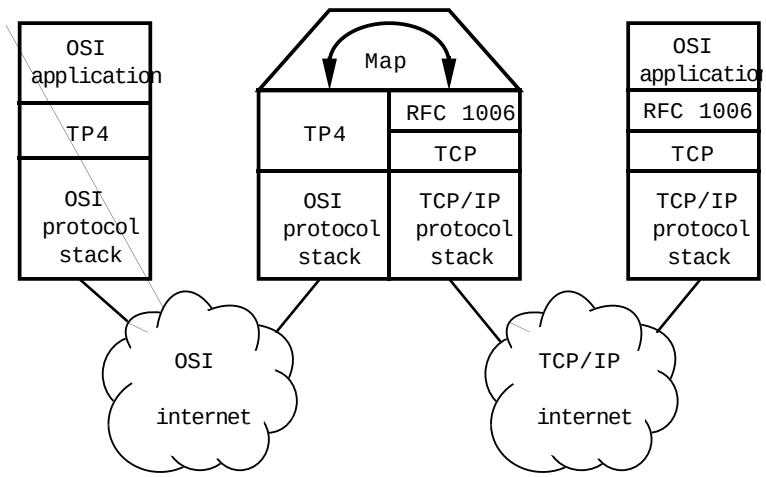

FIGURE 16.4 The Transport Service Bridge Approach

the systems that wish to communicate (1) mimic the same (OSI) transport service and (2) run the same (OSI) application. The transport service bridge is essentially an "interworking unit" in OSI terminology. Rather than relaying the OSI network service, the transport service bridge relays transport service primitives (see Figure 16.5); when, for example, an OSI application residing on the pure-stack OSI system "brady" attempts to establish an association with an OSI application on the hybrid-stack system "bunch" (in which TCP/IP provides the OSI network service), the following primitive sequence occurs:

1. The application on "brady" initiates an association; the upper-layers connection requests are composed, and a T-CONNECT.request is issued. Assuming for this example that TP4 and CLNP are used, the connect request packet is forwarded to a network address associated with the transport service bridge (rather than to the actual destination network address). A T-CONNECT.indication is passed up to the transport service bridge junction (denoted as event 1 in Figure 16.5).
2. The transport service bridge examines the TSAP address information (contained in the T-CONNECT.indication) and determines

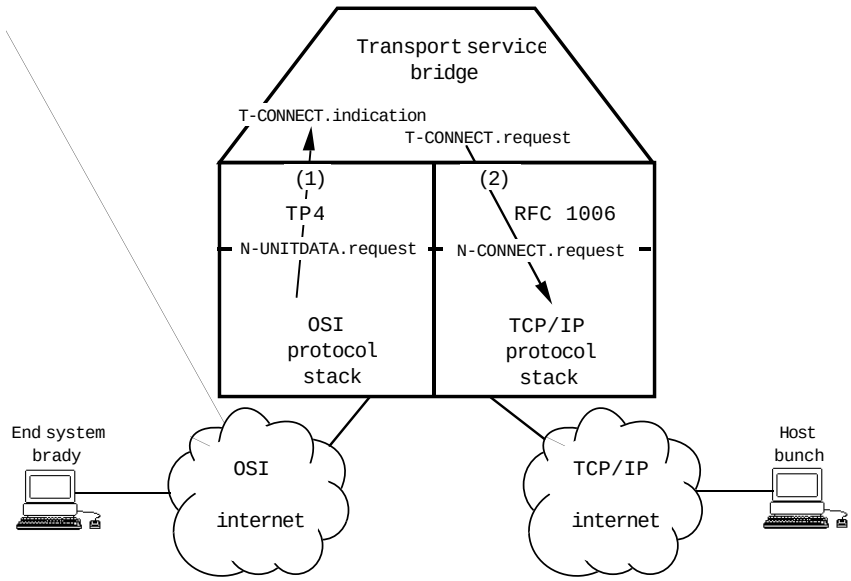

FIGURE 16.5 Transport Service Bridge

from a local "transport-bridging table" that an RFC 1006 host is the target for this connection. Extracting the actual destination TSAP address, port, and IP number from the table, the transport service bridge generates a T-CONNECT.request and attempts to establish a transport connection between itself and a transport server on "bunch" (denoted as event 2 in Figure 16.5).

3. Per RFC 1006, the T-CONNECT.request results in the establishment of a transport connection between the transport server at "bunch" and the transport service bridge using a protocol functionally identical to OSI transport protocol class 0 over TCP.
4. When the transport connection to "bunch" has been established, the transport service bridge confirms the establishment of the transport connection between itself and "brady."
5. The upper-layer connection-establishment protocol packet is transferred from "brady" to the transport service bridge in TP4 data packets; at the bridge, the upper-layer protocol packet is forwarded over the TCP connection to "bunch" using the packetization scheme described in RFC 1006. The upper-layer connection-acceptance protocol packet is returned over the reverse path. Following completion of the upper-layer connection-establishment process, data exchanges follow the same course.

The transport service boundary is often implemented as a programmatic interface between user programs and the host operating system. Chapter 12 describes how multiple transport stacks are accommodated in UNIX by extending the set of *socket()* types to include "sequenced packet" (seqpacket); other operating systems offering direct access to transport services can apply similar extensions. The text description of the transport server in RFC 1006 is a mere 8 pages; one might suspect that transport service bridges similar to the one that makes use of RFC 1006 can easily be defined for other non-OSI protocol architectures as well.

In fact, a transport service bridge offers a more attractive and simpler method of solving the problem of connection-oriented versus connectionless interworking in OSI than the transport relay described in ISO/IEC TR 10172: 1991, since it easily accommodates scenarios in which hosts using TP4 over CLNP wish to communicate with hosts using TP0 over X.25. The transport service bridge approach today is a deployed and expedient choice for users who would like to introduce OSI applications into an existing internet gradually and with little or no disruption of day-to-day operations as well as users who have to cope with multiple OSI network services.

The transport service bridge approach succeeds more often than the application gateway approach in preserving not merely the essential semantics but all semantics of end-user applications. As one might expect, however, the transport service bridge approach is not without its problems. The transport service bridge terminates the transport protocol on either side, interrupting the end-to-end transport features of flow control, error detection and correction, and congestion avoidance (the latter usually a collaboration with underlying network protocols). It also completely breaks any security mechanisms that rely on cryptographic protection of transport protocol header information (such as sequence numbers). A transport service bridge represents a single point of failure in an otherwise dynamically adaptive internetwork. Also, because it must "spoof" the network addresses (so that each party thinks it is talking to the other, when in fact it is talking to the transport service bridge), it makes it much more difficult to diagnose routing problems in the internet.

Contrary to popular belief, the operation of a packetization protocol over TCP described in RFC 1006 is not the only (nor the first) example of using a transport protocol to provide the OSI connection-oriented network service. In the mid-1980s, United Kingdom representatives to ISO introduced a variation of OSI TP4 called "network protocol 4 (NP4)" and proposed it as an alternative to further modifying X.25 PLP for use in providing the connection-oriented network service over connectionless local area networks. NP4 was a TP4 clone capable of conveying OSI NSAP addresses rather than TSAP addresses and could run over CLNP or IEEE 802 LANs (encapsulated in logical link control type 1 frames). NP4 would interwork at the network layer (between a private and public network) with the X.25 packet level protocol. It failed to catch on for the simple reason that it was not X.25, and not CLNP, and so had no natural constituency in the standards-development community.

Hybrid Stacks

The big win for transport service bridges (and transport relays, although with greater complexity) is that they accommodate a "mix and match" of applications across multiple transport fabrics. In principle, and with appropriate transport service bridges, one can run OSI applications over TCP/IP, AppleTalk, etc. The transport service bridge most widely used today is the OSI transport service bridge based on RFC 1006; however, with a transport service bridge designed to provide TCP's stream service, one could certainly run TCP/IP applications over OSI transports.

Encapsulation

In some cases, the problem of multiprotocol interoperability is not how to talk to a foreign host or application but simply how to transit a network that doesn't support a given protocol stack to reach a compatible host or application on the other side. A specific example of such a scenario is one in which the path between two OSI networks is an IP-based network. One way to solve this problem is for the IP-based network to play the role of a "subnetwork" for an OSI network (Figure 16.6). The IP-based network—perhaps the entire TCP/IP Internet—is treated as a point-to-point subnetwork (link), and CLNP data units are simply "encapsulated" in IP datagrams in order to "tunnel" to the friendly system on the other side.

This approach, of course, works equally well if the roles of CLNP and IP are reversed, or if other network protocols are encapsulated for the purpose of tunneling AppleTalk, XNS/IPX, etc. In such configurations, a CLNP network provides a point-to-point link for IP. The principal advantage of this approach is its near transparency—neither network needs to be aware of the protocols of the other, since one network is acting as a simple point-to-point link between two systems belonging to the other. Hosts are unaffected (actually, they have no idea what's going on), and the integrity of transport services is preserved.

Encapsulation does not solve the host interoperability problem; hosts communicating over a path that includes a tunnel lack awareness of the existence of hosts in the network providing the tunnel and do not have the wherewithal to talk to them. Routing is collectively more complicated, since it is taking place independently at two different levels

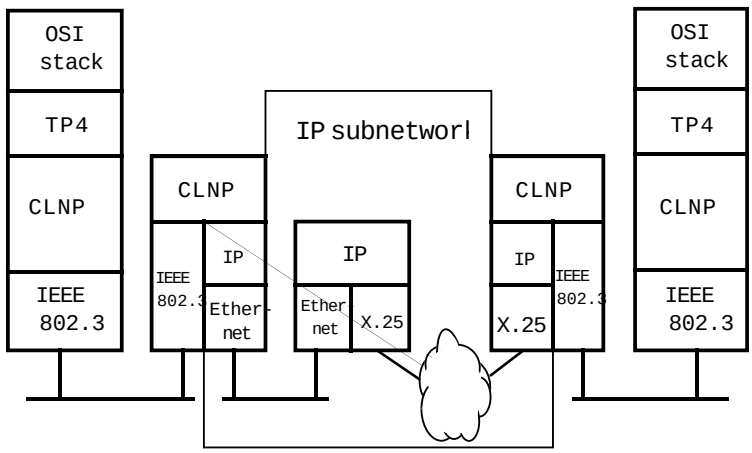

FIGURE 16.6 The Encapsulation Approach

(network and "subnetwork") for paths that include a tunnel (good news for router vendors). Over some paths—if one tunnels across the Internet using the standard 576-octet IP datagram maximum segment size, for example—link efficiency goes to hell in a handbasket.

At present, no standard exists for operating OSI's connectionless network protocol (CLNP) over TCP/IP's internetwork protocol. Although conceptually straightforward, details such as the identification of "encapsulated" protocols, efficient route selection, the use of IP addresses as "subnetwork addresses," segmentation efficiency, and packet size negotiation must be considered. Tunneling, however, is widely practiced across the Internet and is not restricted to the network layer; IBM SNA traffic is frequently encapsulated in TCP/IP packets, and AppleTalk traffic is conveyed using IP encapsulation (and vice versa in AppleTalk networks).

Bringing OSI into a Network

For some, the question remains, "Given these alternatives, how do I bring OSI into my network?" Here, then, is a practical scenario.

Step 1: Introduce OSI Applications in a "Hybrid-Stack" Environment Bring OSI applications to TCP/IP hosts by (1) using the RFC 1006 techniques to run the OSI upper layers over the TCP transport interface provided by the host operating system software and (2) using application gateways (particularly for electronic mail) to tie together OSI and TCP/IP application environments that cannot be changed. The benefit of this method of introduction is that end users get to play with OSI applications while the network infrastructure remains the same (see the network administrator smile). End users also don't have to completely cut over to new applications—the existing applications remain available. OSI applications still don't talk to TCP/IP applications, but needs in this area can be satisfied with gateways. Note that by applying the RFC 1006 principles, this strategy can be employed in corporate internets that have "core" transport services other than TCP/IP.

Step 2: Build TS Bridges Use TS bridges to bring new OSI-only hosts into the TCP/IP network as they become available. Dual-stack hosts may be configured to use either transport service; the choice will depend on the direction the "internet" is taking (e.g., if there is to be an eventual cutover to primarily OSI service, it might be desirable to bring dual-stack hosts onto the OSI backbone). Again, note that by applying the transport service bridge principles, this strategy can be employed in corporate

internets that have "core" transport services other than TCP/IP.

Step 3: Expand the Network Infrastructure to Include OSI As OSI-only or multiprotocol routers (which can route both OSI and TCP/IP traffic) replace or complement IP-only routers in the network, begin to use OSI transport services as host operating system software with embedded OSI transport comes on line. If absolutely necessary, carry TCP/IP-only hosts along with IP-over-CLNP encapsulation.

If homogeneity is an absolute must for a given enterprise network, and OSI is an imperative, routers initially placed to switch both CLNP and IP datagrams can be gradually configured to switch only CLNP data units, and traffic from TCP/IP hosts can be tunneled using CLNP. This is a highly unlikely scenario for the Internet; rather than solving interoperability problems, it merely swings the pendulum to the left rather than the right.

Step 4: Learn to Rely on Names, Not Numbers Wherever possible, rely on a centrally administered (if operationally distributed) directory service to keep track of the mappings between names and the attributes associated with them.

Why Is This Scenario Practical?

OSI has been—and continues to be—successfully introduced into parts of the existing Internet, alongside TCP/IP using steps 1 through 3 (step 4 is coming). OSI applications appeared first in the Internet over ISODE and RFC 1006, and gateway experiments followed shortly thereafter. From the gateway experiences, the Internet community learned to appreciate the transport service bridge approach and has popularized it.

The evolution toward a "dual-stack" Internet (step 3) has proceeded in the same fashion as ripples spreading out from a stone thrown into a pond. The NSFnet staff introduced CLNP alongside IP in the backbone while most end users were busily going about their business running applications over IP, and others were experimenting with ISODE and OSI applications over RFC 1006. Several regional and commercial IP subnetworks are now proceeding with their introduction of CLNP in the same manner, and the attached "stub" networks, encouraged by the presence of backbone CLNP connectivity, are likely to follow suit. Applications running on OSI and dual-stack hosts can reach IP-only hosts of the Internet via transport service bridges and application gateways.

Are the Instrumentation and Expertise Available to Operate OSI Networks?

Network administrators who remain skeptical or unconvinced will ask, "What problems can I expect to run into when I introduce OSI into my network?" OSI demonstrations at Interop and continued OSI testing across the Internet have revealed a great deal of what one can expect when integrating OSI alongside TCP/IP (Hares92, Connexions). Initially, there will be a limited pool of network operators and engineers familiar with OSI protocols. Tools for diagnosing problems with OSI protocols exist, but not for all products. Fortunately, the Network OSI Operations (NOOP) Working Group in the IETF, along with working groups in RIPE and RARE, are preparing the way for OSI operations staff in emerging OSI environments. There is, for example, an OSI equivalent to *ping* (RFC 1139), which has been incorporated into the second edition of CLNP. There is also an OSI *traceroute* that uses increasing values of the CLNP lifetime field of the echo packet in the same manner as IP trace-route uses time-to-live. Implementations will gradually offer utilities such as routing table dumps. Multiprotocol systems in particular may use TCP/IP's Simple Network Management Protocol over OSI (RFC) to collect routing information from the CLNS management information base (RFC 1238), while purist-stack OSI implementations may use the OSI common management information protocol (CMIP) to accomplish the same feats.

Many of these problems may all but disappear if the Network Layer OSI Operations working group of the IETF is successful in establishing a permanent, readily available OSI test bed within the Internet. The Internet community has benefited for years from the practice of using the resources of the community—the Internet protocols and applications—to understand how networks work, and the community has openly shared its experiences conducting daily IP operations. There is no reason to assume that the same recipe should not succeed for OSI.

Conclusion

The Internet becomes more commercialized each day. The cost of "Internet access" through electronic mail is in some places comparable to the cost of plain old telephone service. Even "host access" is within reach of small-business and residential consumers. On-line information available via the Internet is no longer confined to science, engineering, and

technology, and information retrieval tools make navigating through the "maze of data" child's play. You can even send electronic mail to the White House; more importantly, the White House can send electronic mail to Congress!

Whether any of the competing technologies now deployed throughout the Internet or in enterprise networks is "best" has become a largely irrelevant question. The Internet *is* multiprotocol, and is likely to remain so for the forseeable future, if not forever. The application of gateways, hybrid stacks, service bridges, and tunneling described in this chapter will inevitably grow as people who neither know nor care what protocol stack they use make the Internet as fundamental to their daily lives as the telephone.