CHAPTER 1

Overview of API

This chapter provides an introduction and overview to the Cisco IOS for S/390 Application Program Interface (API). It includes these sections:

- Introduction

  Provides a brief overview of the API and transport service modes.

- Background

  Describes the history of multi-vendor computer networks, including a description of ARPANET, Internet, corporate networking technology, and international standards.

- OSI Reference Model

  Describes open systems interconnection concepts and the Internet protocol suite.

- OSI Terminology

  Defines common OSI terms and describes their use, including service access points, connections, primitives, and service data units.

- Transport Layer Services

  Describes the types of services available for transporting data between networks, as well as modes of service and transport layer addressing.

Introduction

The Cisco IOS for S/390 API is a programming interface between application programs and communication subsystems based on open network protocols. The API lets any application program operating in its own MVS address space to access and use communication services provided by an MVS subsystem that implements this interface. Cisco IOS for S/390, which provides communication services using TCP/IP protocols, is an example of such a subsystem.

This programmer's reference describes an interface to the transport layer of the Basic Reference Model of Open Systems Interconnection (OSI). Although the API is capable of interfacing to proprietary protocols, the Internet open network protocols are the intended providers of the transport service. This document uses the term "open" to emphasize that any system conforming to one of these standards can communicate with any other system conforming to the same standard, regardless of vendor. These protocols are contrasted with proprietary protocols that generally support a closed community of systems supplied by a single vendor.

References

For information about the Basic C Library and Socket API, refer to the *Cisco IOS for S/390 C/Socket Programmer's Reference*. The RPC interface to the API is described in the *Cisco IOS for S/390 RPC/XDR Programmer's Reference*.

Refer to the these documents for additional information:

- *TCP/IP – Architecture, Protocols, and Implementation*, Sidnie Feit, McGraw-Hill, 1993.
- *DDN Protocol Handbook, Volume 2: DARPA Internet Protocols*, DDN Network Information Center, SRI International, Menlo Park, California, 1985.
- *Handbook of Computer Communications Standards, Volume 1: The Open Systems Interconnection (OSI) Model and OSI-Related Standards*, William Stallings, Macmillan Publishing Company, New York, 1987.
- *Handbook of Computer Communications Standards, Volume 3: Department of Defense (DOD) Protocol Standards*, William Stallings, Macmillan Publishing Company, New York, 1987.
- *Information Processing Systems, Open Systems Interconnection, Basic Reference Model*, ISO-7498 and ISO-7498/Add.1, International Organization for Standardization, Switzerland, 1984.
- *Information Processing Systems, Open Systems Interconnection, Transport Service Definition*, ISO-8072 and ISO-8072/Add.1, International Organization for Standardization, Switzerland, 1986.
- *Internetworking with TCP/IP: Principles, Protocols, and Architecture*, Douglas Comer, Prentice Hall, New Jersey, 1988.
- *The Open Book: A Practical Perspective on OSI*, Marshall T. Rose, Prentice Hall, 1989.
- *UNIX System V Release 3 Network Programmers Guide*, P/N 307-230, AT&T, 1986.
- *UNIX System V Release 3 Programmer's Reference Manual*, Section 3N, P/N 307-226, AT&T, 1986.

Transport Service Modes

This document defines connection-mode and connectionless-mode protocols that support two basic modes of transport service:

- Connection-mode Protocols

  With connection-mode protocols, a virtual connection is established, over which an ordered, reliable stream of data bytes or messages can be transferred until the connection is released. Each connection is associated with a specific transport user.

- Connectionless-mode Protocols

  Connectionless-mode protocols provide a stateless form of data transfer where each message or datagram is unrelated to previous or successive datagrams, and may be received from or sent to any transport user.

Cisco IOS for S/390 API is compliant with ISO International Standard 8072 (CCITT X.214) and ISO 8072/Add.1, which specify the transport service definition for connection-mode and connectionless-mode service. The semantics of the interface were derived from the Transport Layer Interface (TLI) that is used by AT&T UNIX System V.3 for communicating with networks based on the OSI Reference Model. The mechanics for synchronizing with other MVS tasks and handling exceptional conditions are derived from IBM's Virtual Telecommunications Access Method (VTAM) to simplify learning the rudiments of the interface. Therefore, knowledge of VTAM application programming is helpful.

The basic interface is implemented for application programs written in assembler language, and a working knowledge of IBM Basic Assembler Language (BAL) is assumed. A library of basic interface functions is also provided for applications written in C language. This library implements the C-language equivalent of the basic functions defined for assembler language. For application programs that are derived or ported from BSD UNIX, a library of functions based on BSD sockets is also provided. Subroutine libraries for other high-level languages will be developed as the need arises.

Background

This section provides a brief description of how multi-vendor computer networks began and how they evolved to what they are today.

ARPANET

Research into multi-vendor computer networks began in the mid 1960s, and the first experimental network was deployed in 1969. This effort was sponsored by the Defense Advanced Research Projects Agency (DARPA) and led to the development of the largest network of computer systems in the world today. The original network, known as ARPANET, is still in operation.

The original communication protocols used on ARPANET have since been replaced by more advanced protocols that not only permit larger and more sophisticated networks, but also permit separate networks with their own administration to be connected together into a super-network. This network of networks has come to be known as the Internet and currently consists of tens-of-thousands of computer systems connected to thousands of interconnected networks. Any computer system on any one of these individual networks has the capability to communicate with any other computer system on some other network halfway around the world.

TCP/IP

The set of communication protocols that evolved out of this experimental effort is known as the Internet (TCP/IP) protocol suite. In 1983, the Department of Defense mandated that all DoD commands and agencies use these protocols for interconnecting computer systems over long haul facilities, and the Defense Data Network (DDN) was developed to provide wide-area service. At the same time, the Internet protocol suite was integrated into the University of California Berkeley Software Distribution (BSD*)* of the UNIX operating system. Since BSD UNIX is the basis for many workstation and mini-computer operating systems, the Internet protocol suite was rapidly propagated throughout the educational, scientific, and engineering community.

Today, the Internet protocol suite is the most widely used set of protocols for interconnecting heterogeneous systems. Hundreds of vendors offer products based on these protocols, including all of the major computer system manufacturers. Also, as rapidly as new transmission media and network technologies emerge, Internet protocol implementations are developed to support them. Virtually every form of local and wide-area transmission is supported, ranging from simple serial links to local area networks based on Ethernet, Token Ring, and token-bus designs, to national and international packet-switched networks using terrestrial and satellite links. Even nodes operating over amateur radio links can be connected to the Internet.

Corporate Networking Technology

This computer networking technology is also spreading beyond governmental and educational institutions into the commercial business sector as well. As corporations shift the emphasis from centralized control to decentralized management of information resources, computer networking is becoming a critical component of a business' operation. With the availability of inexpensive workstations and personal computers, applications are becoming widely distributed and computing resources are becoming specialized and diverse. Coupled with the need to integrate all departments into the corporate network and to automate the business process, heterogeneous networking and interoperability have never been so important.

While proprietary network protocols can provide sophisticated and comprehensive services to a homogeneous community of systems, they are inappropriate for large networks of multi-vendor systems due to their lack of universal implementation. Therefore, the trend is away from proprietary networks and towards open networks that can interconnect a variety of computer systems. Use of open networks eliminates the reliance on a single vendor for services and equipment and makes it possible to rapidly adapt to changing technology.

OSI Reference Model

To properly understand the role of the API and the services it provides, a brief overview of the OSI Reference Model may be helpful. The Reference Model is defined by ISO International Standard 7498 (1984). Addendum 1 (1987) expands the definition to include connectionless-mode transmission.

Open Systems Interconnection

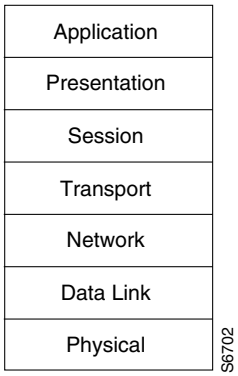
Proprietary networks such as SNA tend to be closed, even though they may have been conceived as open. Each manufacturer develops its own set of conventions for interconnecting its own equipment and refers to these conventions as its "network architecture". Recognizing the need to interconnect systems from many different manufacturers, ISO created a subcommittee with the objective of developing international standards required for Open Systems Interconnection (OSI). Their initial

effort resulted in the development of a standard model of architecture that would constitute the framework for the development of standard protocols. This architectural model is known as the Basic Reference Model of Open Systems Interconnection.

The Seven Layer Model

The OSI Reference Model generally is depicted as a vertical stack representing the seven layers of the model. This diagram illustrates this model:

Figure 1-1 The OSI Reference Model



The entire stack represents the OSI environment on the local system, and each layer represents a subsystem within the local system that implements the functionality of the given layer. The application layer is usually on the top, and the layer that represents the physical media is on the bottom. Systems that conform to the Reference Model are called open systems. Open systems are internally independent but provide common external services via standardized protocols publicly defined for each layer.

Layer Interaction

An entity operating within a layer (or subsystem) communicates with a peer entity operating within the same layer on another system, and using services provided by the lower layer, they cooperate to provide enhanced services to entities operating in the layer above. The basic idea of layering is that each layer adds value to services provided by the set of lower layers in such a way that the highest layer is offered services to run distributed applications. Layering thus divides the total communication process into smaller processes.

Another basic principle of layering is to ensure independence of each layer by defining services provided by the layer to the next higher layer, independent of how these services are performed. This lets changes be made in the way a layer or set of layers operates, provided they still offer the same service to the next higher layer.

Except for the highest layer, which operates for its own purpose, entities distributed among the interconnected open systems work collectively to provide specific services to entities in the layer above. Cooperation between entities within one layer is governed by protocols that precisely define how the entities work together using the services of the lower layer.

This list briefly describes the function of each layer and the basic services it provides to the layer above:

Application

The highest layer, layer 7, is the Application layer. It provides a means for the application program to access the OSI environment, and provides the distributed information service appropriate to the application. These are the basic application services:

- Identify and authenticate communication partners
- Determine adequacy of resources and quality of service
- Establish agreement on error recovery procedures, control of data integrity, and privacy mechanisms
- Select dialogue discipline and identify constraints on data syntax
- Provide synchronization and data transfer

Presentation

The next layer, layer 6, is the Presentation layer. It provides for the representation of information that application entities either communicate or refer to in their communication. The presentation layer is concerned only with the representation of data (in other words, syntax) and not its semantics. Provides session services plus these facilities:

- Negotiation and recognition of syntax
- Transformation of syntax, including data transformation, formatting and compression

Session

The Session layer, layer 5, provides the means necessary for cooperating presentation entities to organize and synchronize their dialog and to manage their data exchange. These are the basic session services:

- Establish, synchronize, and release session connections
- Exchange normal and expedited data
- Manage dialogue interaction and report exceptions

Transport

The Transport layer, layer 4, provides transparent transfer of data between session entities and relieves them from any concern with the detailed way in which reliable and cost effective transfer of data is achieved. These are the basic transport services:

- Establish, maintain, and release transport connections (connection- mode)
- Transfer normal or expedited data (connection-mode)
- Map a request for transmission of a *Transport Service Data Unit (TSDU)* onto a request for the connectionless-mode network service (connectionless- mode)

Network

The Network layer, layer3, provides the means to establish, maintain, and terminate network connections between open systems, and provides transport entities independence from routing and relay considerations. These are the basic network services:

- Establish, maintain, and terminate network connections (connection- mode only)
- Provide sequencing, flow control, and expedited data transfer (connection mode only)
- Select routes and quality of service
- Segment and/or block data, and transfer network service data units
- Provide error detection, recovery, and notification

Data Link

The Data Link layer, layer 2, provides functional and procedural means to establish, maintain, and release data link connections, and/or transfer data link service data units between network entities. These are the basic data link services:

- Establish, maintain, and release data link connection (connection- mode only)
- Provide sequence and flow control (connection-mode only)
- Delimit and transfer service data units over data link
- Provide error detection, recovery, and notification

Physical

The lowest layer, layer 1, is the Physical layer. It provides the mechanical, electrical, functional, and procedural means to activate, maintain, and deactivate physical connections for bit transmission between data link entities. These are the basic services:

- Provide physical connection between data link entities
- Transfer physical service data units
- Provide fault detection and notification

International standards either have been or are being defined at each layer of the OSI Reference Model. At least two standards are specified for each layer:

- One that defines the services provided by the layer.
- One that defines the protocol by which those services are provided.

A service interface standard at any layer frees users of the service from the details of how that layer's protocol is implemented, or even which protocol is used to provide the service, if more than one is defined. ISO International Standard 8072 defines the service interface for the transport layer.

The transport layer is important because it is the lowest layer in the OSI Reference Model that provides the basic service of reliable, end-to-end data transfer needed by application programs and higher-layer protocols. In doing so, this layer hides the topology and characteristics of the underlying network from its users. More important, however, the transport layer defines a set of services common to layers of many contemporary protocol suites, including Internet, ISO, XNS, and SNA. The protocol suite of primary interest to the API (Internet) is summarized in the following subsection.

Internet Protocol Suite

The maturity of the Internet protocol suite favors it as the dominant set of protocols for interconnecting multi-vendor systems. The protocol suite has been standardized by the U.S. Department of Defense, and certification procedures have been established to test protocol conformance. This suite of protocols is often referred to as the TCP/IP family of protocols, derived from the names of the transport and network layer protocols.

Major Application Layer Protocols

The Internet suite defines these major application layer protocols:

- Telnet virtual terminal service supports remote login from ASCII terminals. Telnet options are defined that permit login from full-screen, 3270-type terminals.
- File Transfer Protocol (FTP*)* supports the transfer of arbitrarily large files or programs between dissimilar file systems. File maintenance utilities such as renaming, deleting, and listing of file names are also supported.
- Simple Mail Transfer Protocol (SMTP) supports the exchange of electronic mail. SMTP can also be used to transfer files, but its use is generally limited to moderate size files containing ASCII text. Some systems may let SMTP be used to submit batch jobs and retrieve batch output.

Presentation and session layer services are not defined as separate Internet protocols, and the corresponding functions are embedded within the application layer protocols.

Major Transport Layer Protocols

Internet defines these major transport layer protocols:

- Transmission Control Protocol (TCP) provides ordered, reliable transfer of a stream of data bytes with no explicit message boundaries. TCP is a connection-mode transport service that establishes virtual connections to associate two communicating processes.
- User Datagram Protocol (UDP) provides unreliable transfer of independent units of data called datagrams. UDP is a connectionless-mode transport service that maintains no permanent association between network processes. Since UDP maps directly into the underlying connectionless network layer, data transfer is very efficient.

Internet Protocol

Internetwork Protocol (IP) is a sublayer protocol of the network layer that is common to all connected systems. IP provides a connectionless datagram delivery service to any IP node on the Internet. If the destination node is not on the local network, IP routes the datagram to a gateway that forwards the datagram to the next network in the sequence of connected networks. Fragmentation and reassembly of datagrams is also provided by IP to support datagrams larger than the underlying layers can accommodate on the physical media.

Generally, IP is implemented over a Connectionless Network Service (CLNP), but connection-mode services such a X.25 are also used. In addition to wide-area packet-switched networks, local area networks based on Ethernet, token-bus, and Token Ring technologies are generally used. These local area networks correspond to the IEEE 802.X series of protocols, usually without 802.2 (LLC) encapsulation.

OSI Terminology

The Cisco IOS for S/390 API is oriented towards the Internet protocol, and may also be described in terms of OSI-based, ISO protocols. Since the OSI is a general model, it is appropriate that OSI terminology be used to describe the operation and use of the API. This terminology applies to the Internet protocol suite as well. Sometimes the names are different, such as TSAP versus port for identifying upper-level entities, but, generally, the concepts are similar. This chapter will describe concepts in both TCP and OSI terminology. After the initial reference to both terms, the book will refer to concepts in TCP terminology.

The OSI architecture defines a set of rigorous conventions and terminology for specifying OSI-compliant protocols and service definitions. Some of this terminology has already been used in this document.

This document has already discussed that open systems are comprised of distinct, functional layers. Within each layer, entities communicate with entities above and below across an interface. For each layer of an open system, two standards must be defined: one describing the services provided by the layer and one describing the protocols used to provide the services.

Transport Services and Protocols

Since this document is concerned with the transport layer, it discusses transport services and transport protocols.

Transport Service Provider

The hardware and software that implement the protocols and services is called the provider. Thus, the provider of transport services is called the transport service provider.

Transport Service User

The hardware and software that uses the services of an adjacent layer is a user of those services and in this document is referred to as a transport service user.

Note For brevity, this document sometimes refers to these terms as the transport provider and transport user, and simply abbreviates them as TP and TU.

In the most basic sense, the API implements the interface between a TU and a TP.

Service Access Points

Each user of a given layer's services accesses the provider via a unique Service Access Point, generally abbreviated as SAP. A Transport Service Access Point is referred to as a TSAP. An identifier that specifies the location of an SAP is a Service Access Point Address, or simply "address," and if the identifier is that of an SAP in the transport layer, it is a "transport address."

Connections

An association established by a layer between two peer users for the transfer of data is called a connection, and each terminal end of a connection within a service access point is called a connection endpoint. Thus, a connection between two transport users is called a transport connection, and its terminal end within a TSAP is called a Transport Connection Endpoint (TCEP), or transport endpoint for short. This document uses the term endpoint to mean a transport endpoint.

Primitives

The provider supplies services to the user via the invocation of primitives. A primitive specifies the function to be performed and is used to pass parameters consisting of data and control information. The primitives and their parameters indicate what information flows between the user and the provider of the service. The interface determines the method by which this information is conveyed.

These types of primitives are used to define the interaction between adjacent layers:

Request

A request is the primitive initiated by a user to request a service from the provider.

Indication

An indication is initiated by the provider to inform the user of some condition, such as the presence of incoming data.

Response

Some indications require an action on the part of the user; this is the response primitive.

Confirm

A confirm primitive is initiated by the provider to convey to the user the information that was given by the peer user to the peer provider in a response primitive. The confirm primitive implicitly means that the original request has been completed.

Service Data Units

Finally, the unit of data transferred across the interface between a user and provider, and subsequently between two peer users, is called a Service Data Unit (SDU). The provider is responsible for mapping service data units onto Protocol Data Units (PDUs*)* by outgoing SDUs into PDUs and reassembling incoming PDUs into SDUs. A small service data unit whose transfer is expedited is called an expedited service data unit. Within the transport layer, these data units are referred to as TSDUs, TPDUs, and expedited TSDUs, respectively.

Transport Layer Services

The transport layer is responsible for the delivery of data between transport users across a communications facility. In providing this service, the transport layer may need to deal with a variety of networks with differing characteristics and capabilities. Depending on the sophistication of the lower layers, the transport layer may require complex protocols to carry out its designated functions.

By defining a set of common services and mapping these onto the capabilities and services of the network layer, the transport layer can shield upper layers from many of the details of data transfer. As such, the upper layers are able to focus on user requirements and applications.

While the services of any given layer are standardized, the method of acquiring them is not. This is because the manner in which information is conveyed between layers depends heavily on the environment in which they are implemented. Therefore, ISO 8072 defines the services of the transport layer and the API determines how these services are acquired.

The mechanisms used by the API to initiate, complete, and synchronize requests for service should be familiar to you as an MVS application programmer. However, the semantics of those requests, which derive from the underlying transport services, may be foreign to you if you are unfamiliar with OSI-based networking. Therefore, this subsection provides a brief overview of transport services and is provided as an introduction to the concepts and facilities of the API. As before, this discussion is oriented towards OSI-compliant systems. However, any major differences between ISO and Internet transport services are noted.

Modes of Service

These types of service are supported by Internet and ISO protocols:

- Connection mode

  With this service, a transport connection is established between two peer transport users prior to the exchange of data.

- Connectionless mode

  With this service, no transport connection is established and each data unit transferred is independent of previous or subsequent data units.

Connection-Oriented Transport Service (COTS)

Connection-mode service within the transport layer is referred to in this programmer's reference as Connection-Oriented Transport Service (COTS). COTS provides the means to establish, maintain, and release transport connections providing duplex transmission between two transport users. It guarantees that all data units transferred arrive at their proper destination intact, uncorrupted, in the order in which they were sent; if this cannot be achieved, the user of the transport service is notified. COTS is the most widely used of the two modes of service.

COTS provides these basic services:

- Transport connection establishment
- Data transfer
- Transport connection release

These services describe the three phases of operation within the transport layer:

- The initial phase is the connection establishment phase, during which the transport connection is established.
- The middle phase is the data transfer phase, during which all data is transferred.
- The final phase is connection release phase, during which the connection is released and the association between transport users is terminated.

COTS service primitives are many and varied and are discussed in terms of the basic services to which they apply. This table lists the COTS primitives associated with each basic service along with the parameters provided with each primitive:

Table 1-1 COTS Service Primitives

Service	Primitive	Parameters
Connection Establishment	T-CONNECT.request	Called Address Calling Address Service Options Quality of Service User Data
	T-CONNECT.indication	Called Address Calling Address Service Options Quality of Service User Data
	T-CONNECT.response	Responding Address Service Options Quality of Service User Data
	T-CONNECT.confirm	Responding Address Service Options Quality of Service User Data
Data Transfer	T-DATA.request T-DATA.indication T-EXPEDITED-DATA.request T-EXPEDITED-DATA.indication	User Data User Data User Data User Data
Connection Release	T-DISCONNECT.request T-DISCONNECT.indication	User Data Disconnect Reason User Data

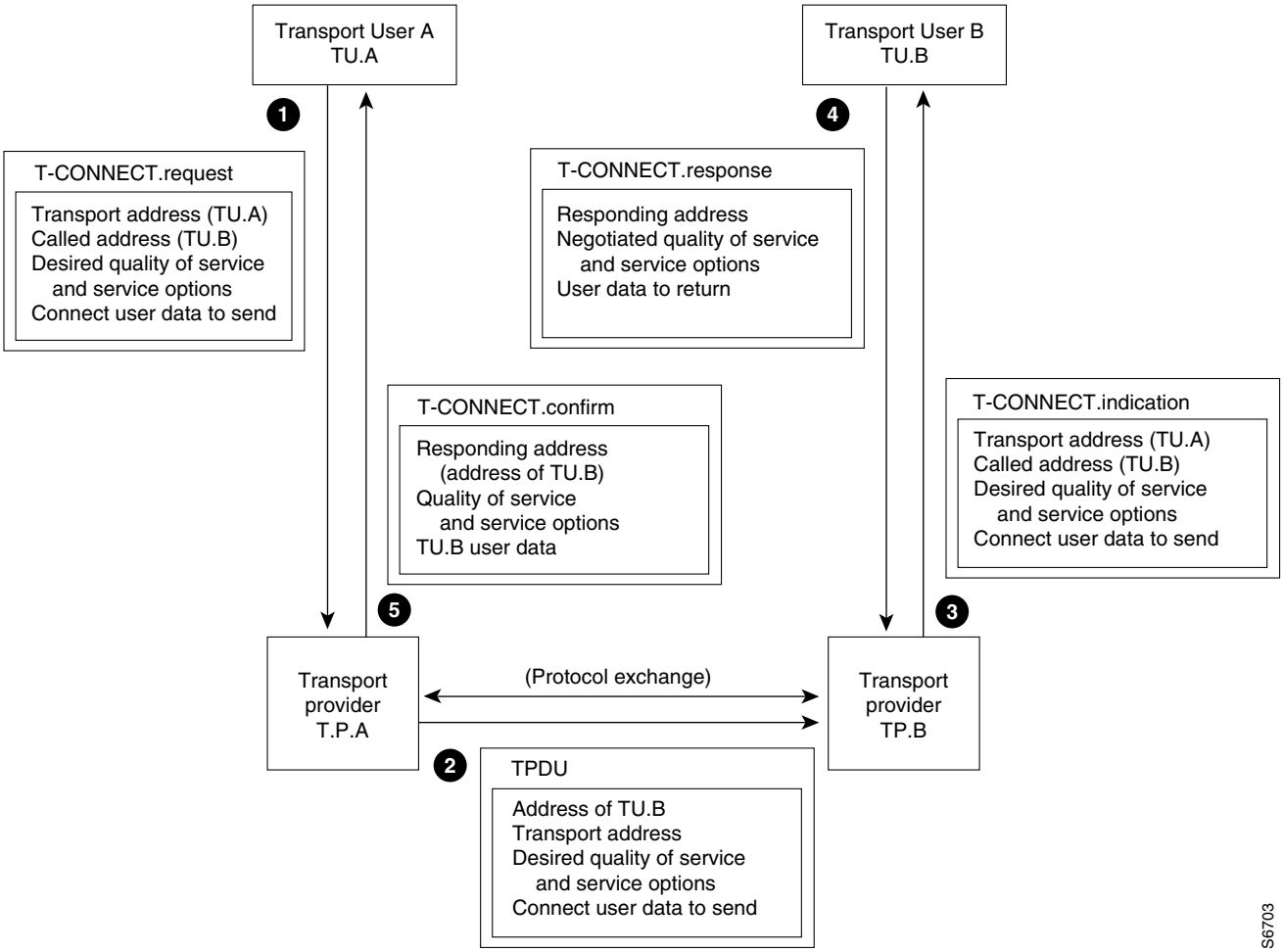
Transport Connection Establishment

Transport providers establish connections to transport users, which the provider identifies by transport address pairs. The transport service and the users negotiate the quality of service of the transport connection. When connections are established, the class of transport service to be provided can be selected from a defined set of available classes of service. These service classes are characterized by combinations of selected values of parameters such as throughput, transit delay, and connection set-up delay and by guaranteed values of parameters such as residual error rate and service availability.

Connection Establishment Example

The connection establishment phase is best described by example. This diagram illustrates this process between two transport users, where Transport User A (TU.A) wants to communicate with Transport User B (TU.B):

Figure 1-2 Connection Establishment Example



The steps in this process are:

Step 1 TU.A issues a T-CONNECT.request to its transport provider, TP.A, that includes this information:

- Its own transport address (calling address)
- The protocol address of TU.B (called address)
- The desired quality of service and service options
- Any connect user data to send to TU.B (connect user data is limited to 32 bytes)

At this time, TU.A may indicate whether the requested values for the quality of service and service option parameters are absolute or whether degraded values are acceptable.

Step 2 The provider sends a transport connection request (TPDU) containing these items to its peer, TP.B.

Step 3 TP.B issues a T-CONNECT.indication to TU.B.

If the T-CONNECT.request from TU.A indicated that degraded values for quality of service and service options was acceptable, either transport provider (TP.A or TP.B) or the peer transport user (TU.B) can negotiate particular parameters to a lesser value.

Step 4 If TU.B wants to communicate with TU.A, it accepts the connection request by issuing a T-CONNECT.response giving this information:

- The responding protocol address
- Negotiated quality service and service options
- Any user data to return to TU.A

Currently, the responding address given in the T-CONNECT.response primitive must be identical to the called address provided in the T-CONNECT.indication primitive. Future definitions of the transport service may let the responding address be different (for example, the responding address may be the specific address resulting from calling a generic address).

Step 5 After a suitable protocol exchange between TP.B and TP.A, TP.A issues a T-CONNECT.confirm primitive informing TU.A of the successful connection establishment. The confirmation message includes this information:

- The responding address
- Quality of service and service options
- TU.B user data

The parameter values returned with the T-CONNECT.confirm primitive are the final, negotiated values that must be equal to or inferior to the requested values.

The TCP Connection Model

The OSI connection model requires that the transport provider obtain the explicit permission of the transport user to establish the connection. The TCP connection model differs from this in a subtle but significant way. In the TCP model, the responding transport user is passive and does not intervene explicitly in the connection process. Rather, the transport user receiving connection requests advises its transport provider in advance which transport users it is willing to connect to. This is done by providing a partially or fully qualified protocol address of the calling transport user. Partially qualified protocol addresses are treated as wild-card specifications, and the null address indicates that a connection from any transport user is acceptable.

TCP also permits the establishment of a transport connection between two transport users that issue simultaneous connection requests. ISO does not. Simultaneous symmetric connection requests in the OSI model results in two transport connections. Generally, this is of academic importance, since the active and passive nature of clients and servers is such that simultaneous symmetric connection requests are avoided.

TCP connection service supports a type-of-service parameter that is defined at the IP layer and is used for the life of the connection. However, the definition of this parameter is not as rich and extensive as the ISO counterpart. Also, the results of negotiation are the opposite of ISO. If the two TCP transport users disagree on the choice of type-of-service, the superior value is used.

Data Transfer

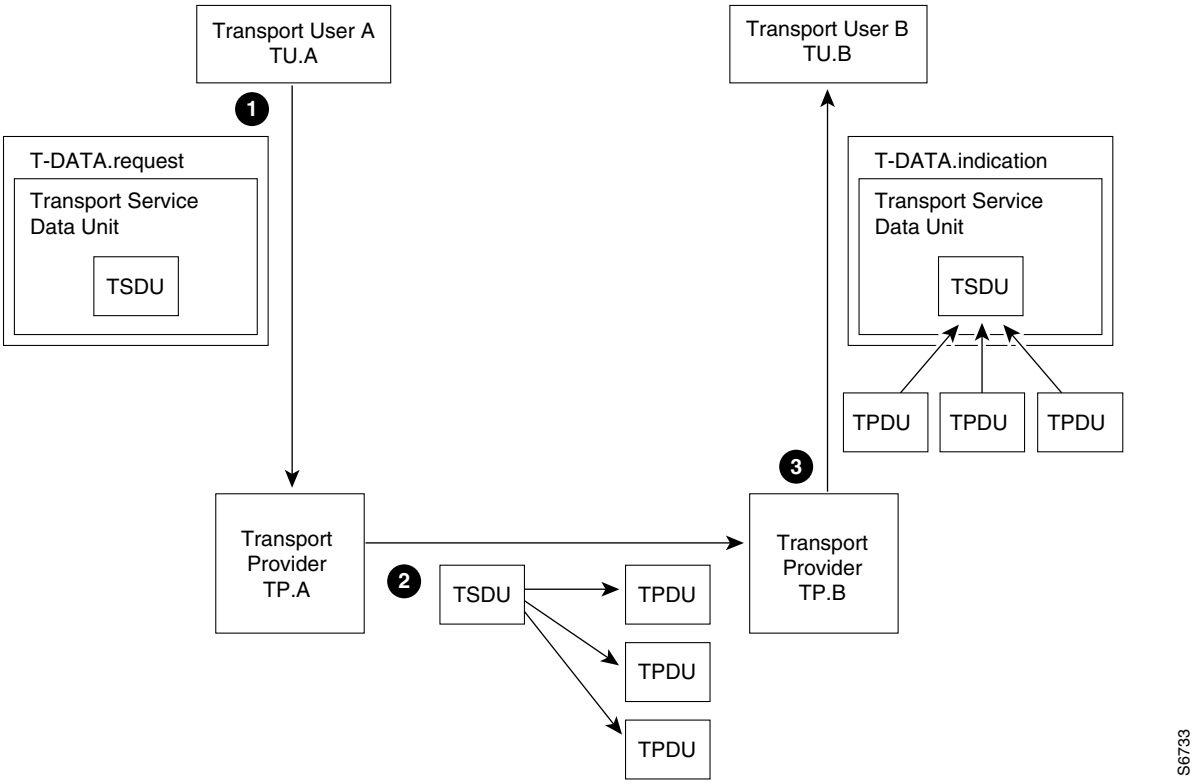
The OSI transport layer offers these services:

- The transfer of normal data
- The transfer of expedited data

The expedited data transfer service must be requested during connection establishment and is negotiated as a service option.

Two primitives are defined for normal data transfer.

Figure 1-3 Data Transfer Process



These are the steps in this process:

Step 1 TU.A issues a T-DATA.request to its transport provider (TP.A), providing the user data as a parameter. This user data parameter is the Transport Service Data Unit (TSDU) and may be of any arbitrary size.

Step 2 The transport provider TP.A takes one of these actions and transfers the data to the remote transport provider (TP.B)

- If the TSDU is longer than one TPDU, the transport provider segments the TSDU into multiple TPDUs.
- If the TSDU is shorter than one TPDU, the transport provider blocks it with previous TSDUs.

Note In the example diagram, transport provider TP.A segments the TSDU into multiple TPDUs.

Step 3 The transport provider TP.B must reassemble or unblock the TSDU and deliver it to the TU.B transport user using a T-DATA.indication primitive.

The TCP Data Transfer Model

TP.A and TP.B provide for the transfer of a stream of TSDUs in both directions simultaneously while preserving the integrity, sequence, and boundary of each TSDU. TCP provides a similar service except that there is no notion of a TSDU; that is, TCP transfers a continuous stream of bytes with no record boundaries. If record boundaries are required by the transport user, they must be embedded within the data stream itself.

The ISO transport user has no direct control over when the transport provider forwards the data. However, the TCP transport provider can be forced into forwarding any buffered data by using a push mechanism. When invoked by the sending transport user, any data buffered by the local transport provider must be transmitted to the destination provider and delivered to the transport user. However, since there is no method for the destination transport user to determine where within the data stream a push occurred, this mechanism cannot be used to mark data boundaries. It serves only to force the transport provider to forward locally buffered data, using partially filled packets, if necessary. Pushed data is subject to the normal flow control restrictions of the protocol.

OSI Expedited Data Transfer Service

OSI also offers a data transfer service for small, expedited TSDUs containing up to 16 bytes of user data. This service is optional in the sense that it must be requested by the calling transport user, agreed to by the called transport user, and supported by the selected class of transport protocol. When this service has been enabled, the sending transport user issues a T-EXPEDITED-DATA.request providing the data as a parameter, and the destination provider delivers the expedited TSDU as a parameter of the T-EXPEDITED-DATA.indication primitive.

The transfer of expedited TSDUs is subject to different quality of service and separate flow control from that applying to normal data transfer.

These are some of the key points:

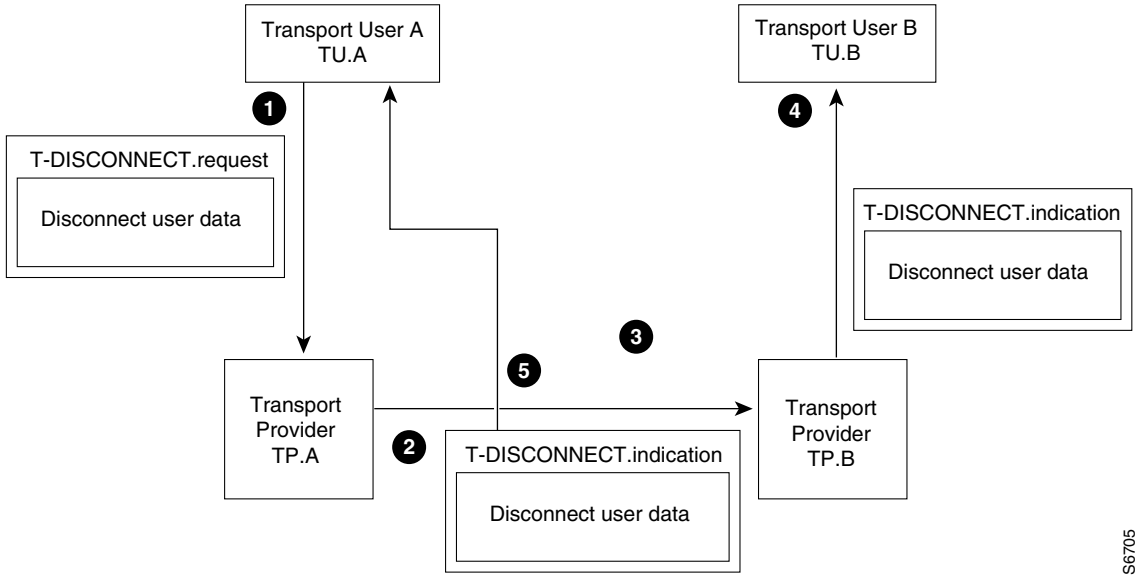
- The transport provider guarantees that an expedited TSDU is not delivered after any subsequent normal or expedited TSDUs transmitted on the same connection.
- Expedited TSDUs may bypass normal TSDUs but can be delivered to the destination transport user only when normal data is not being accepted.
- Since expedited TSDUs are small, and since only one may be in transit at a time, expedited data transfer is best used for exceptional conditions and is inappropriate for bulk data transfer.

The analogue in TCP transport service is called urgent data. It differs from ISO-expedited data transfer in that urgent data is transmitted in-band with normal data, and a pointer to where the urgent data ends may precede the data. The destination transport user is advised of the urgent condition and should dispose of unprocessed data quickly until the marked location in the input stream is reached. There is no length associated with TCP urgent data per se; the urgent pointer merely marks where the urgent data ends. Some providers attach special significance to the first byte of data following the urgent pointer and provide an out-of-band mechanism to read it. However, this is a nonstandard service.

Transport Connection Release

The connection release service is used to abandon connection establishment, release an established connection, or indicate a failure to establish or maintain a connection. Connection release is permitted at any time regardless of the current connection phase, and a request for release cannot be rejected. The transport provider does not guarantee delivery of any user data once the release phase is entered. Two primitives apply to connection release. This diagram illustrates this process:

Figure 1-4 Transport Connection Release



1 A T-DISCONNECT.request is issued to the applicable transport provider (for example, TP.A) by either transport user (for example, TU.A) to abandon connection establishment or to release an established connection.

2 The transport provider delivers a T-DISCONNECT.indication to the connected transport provider (TP.B).

3 The transport provider delivers the message to its transport user (TU.B). Up to 64 bytes of disconnect user data can be provided as a request parameter and is delivered to the destination transport user as long as the indication was the result of a user-initiated disconnect. A reason parameter is always provided with the indication.

4 The T-DISCONNECT.indication primitive also can be issued by a transport provider to indicate its inability to establish a connection or a failure to maintain an existing connection. In this case, no user disconnect data is provided with the indication, but a reason parameter indicates the cause of the indication (if known).

5 The ISO connection release service is similar to the TCP connection abort service (connection reset). However, unlike TCP, ISO does not support an orderly connection release. This form of release guarantees that any buffered data is delivered to the transport user before the connection is released. ISO merely guarantees that buffered data is delivered to the destination transport provider.

The TCP orderly release service is sometimes referred to as a “graceful close”. Each transport user must agree to release the connection before the connection is dissolved, and until then, the TCP transport provider maintains the connection. Once a connection has been released by a transport user, no more data can be sent, but the user can continue to receive data. Any data previously sent is delivered to the destination transport user.

Endpoint States and Service Sequence

Four states are defined for the connection endpoint when using Connection-Oriented Transport Service (COTS). This table lists these states:

Note Read Connection-Oriented Transport Service (COTS) for a list of valid sequences of COTS service primitives at a given connection endpoint.

State 1, Idle

The idle state reflects the absence of a transport connection. It is the initial and final state of any sequence and once it has been re-entered, the connection is released.

State 2, Outgoing-connection-pending

The outgoing-connection-pending state indicates that connection establishment is in process and is entered when the local transport user issues a valid connection request. The transport user must wait for a connect confirmation or abandon the connection attempt with a disconnect request.

State 3, Incoming-connection-pending

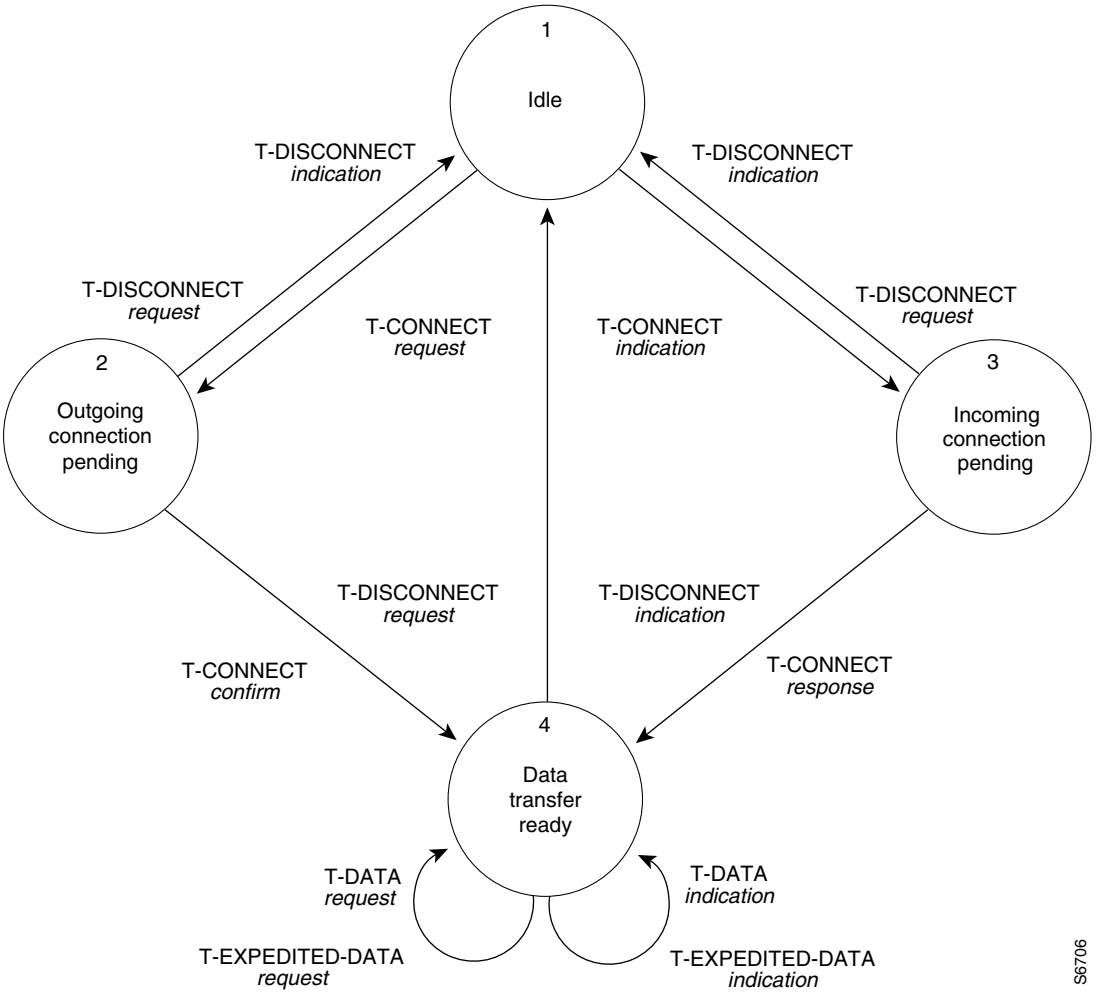
The incoming-connection-pending state indicates that connection establishment is in process and is entered when the local transport user has received a connect indication from its transport provider. The transport user must either accept or reject the connection request.

State 4, Data-transfer-ready

The data-transfer-ready state reflects the presence of a fully established transport connection. End-to-end data transfer can only occur in this state.

This sequence diagram shows the COTS service primitives. The states are represented by circles and transitions between states are represented by arrows. The labels next to an arrow represent the events that can cause the state transition. The current state is at the base of the arrow and the new state is at the head (in other words, point) of the arrow.

Figure 1-5 Data-Transfer-Ready State



Connectionless Transport Service

The OSI Reference Model as originally published did not provide for connectionless-mode service. However, recognizing that this unnecessarily limited the power and scope of the Reference Model and excluded important classes of applications and network technology that are fundamentally connectionless in nature, an addendum was published that provided for connectionless-mode service within all layers of the model. Connectionless-mode service within the transport layer is referred to in this document as Connectionless Transport Service (CLTS). Within the Internet suite of protocols, connectionless-mode usually is referred to as a datagram service.

CLTS provides for the transfer of a single unit of data (TSDU) from a source TSAP to a destination TSAP without establishing a connection. The transport user may initiate such a transfer by the performance of a single service access. Since the transport provider is not required to relate this service access with any previous or subsequent access, all information required to deliver the TSDU must be provided with each access. This is in sharp contrast to COTS, where such information need be supplied only during establishment of the connection.

CLTS requires a prearranged association between peer transport users that determines the characteristics of the data to be transmitted, and no dynamic negotiation of parameters and options may be done. Since this prearrangement is unknown to the transport provider, such information must be provided with each unit of data to be transmitted.

From the perspective of the transport provider, data units are unrelated to their predecessors and successors, and therefore, ordered delivery without duplication or loss cannot be provided. No indication is returned to the sender when data units have been delivered to the destination, and no flow control mechanism exists to prevent the source from sending data units faster than the destination can receive them. However, the transport provider does offer the optional guarantee that each data unit that arrives at the destination arrives intact – it is not truncated or corrupted. In the absence of congestion and network failures, CLTS can provide generally reliable service.

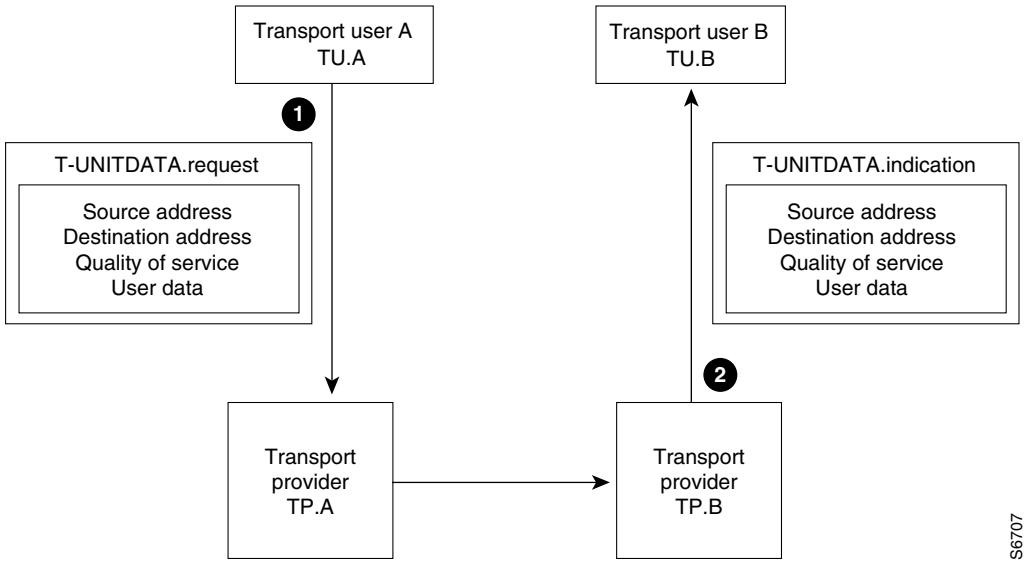
CLTS services are acquired via the invocation of two service primitives:

- T-UNITDATA.request
- T-UNITDATA.indication

Each primitive requires four parameters:

- The source transport address
- The destination transport address
- The quality of service
- The user data

Figure 1-6 Connectionless Transport Service

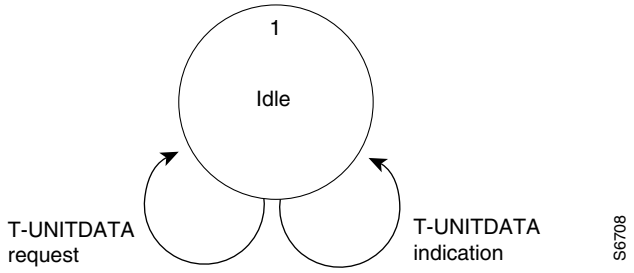


1 If a transport user (for example, TU.A) wants to send a unit of data to some destination, it provides the data and all required parameters to the transport provider (for example, TP.A) with a T-UNITDATA.request primitive.

2 When the transport provider (for example, TP.B) receives a unit of data destined for the transport user (for example, TU.B), it is presented to the user along with the accompanying parameters via a T-UNITDATA.indication primitive.

This sequence diagram illustrates the simplicity of connectionless service for primitives issued at a CLTS endpoint:

Figure 1-7 Connectionless Service for Primitives Issued at a CLTS Endpoint



Often the connectionless transport service is mapped directly onto a connectionless network service that has identical service primitives, resulting in a service that is efficient and easy to use.

UDP is the Internet equivalent of ISO connectionless-mode transport service. UDP is similar in capability, except that UDP does not support the quality of service parameter.

Transport Layer Addressing

Many of the services described in this chapter require addresses to be supplied as parameters of the service primitive.

Example

COTS connection establishment primitives require the calling and/or called (responding) protocol address of the transport user, and the CLTS data transfer primitives require the source and destination protocol address of the TSDU. The addresses supplied with these primitives must identify the transport user, the transport provider through which the transport user is accessed, and the end-system (in other words, host) in which the transport user and transport provider reside.

Internet Domain Addressing

Transport layer addresses for the Internet domain contain a transport-layer and network-layer component. The transport-layer component is called a port number. At the network layer, the address is an internet address. Both the port number and internet address are fixed in length: four bytes for the internet address and two bytes for the port number.

An internet address is a 32-bit (four-octet) binary value. This value defines the overall address space which is the set of address numbers. The dot notation is a common way of representing an internet address so that users can read and write it easily. Each octet of the address is converted into a decimal number and the numbers are separated by dots.

The port number identifies services and/or clients using those services at a specified Internet address. Server port numbers are generally used by multiple clients. Client port numbers are generally used by a single client and assigned dynamically by the transport provider.

Example

For MERMAID.HQ.MYCOMPANY.COM:

- The 32-bit binary address is 10001010 00101010 10000000 11010000
- The dot notation address is 138.42.128.208

An internet address is made up of two components:

- A network address
- A local (host) address

Figure 1-8 Internet Address Structure

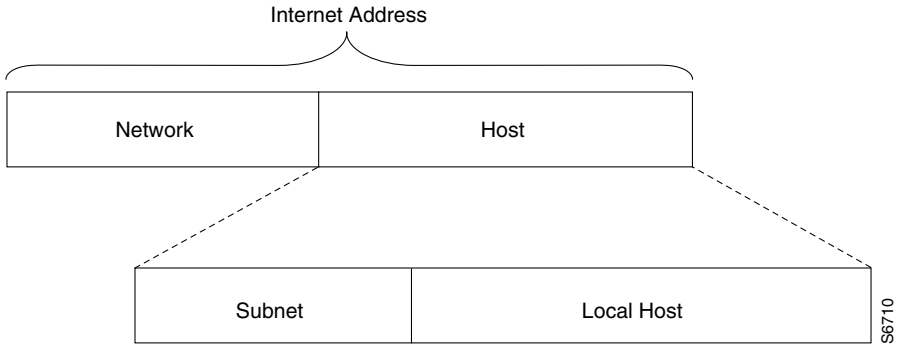


Table 1-2 Internet Address Components

Component	Part	Description
Internet Address	Network Address	Indicates the network to which the node is attached. Assigned by the Internet Network Information Center (NIC)
	Host Address	Identifies an individual node. Assigned by the authority that administers the particular network. The host number may be further subdivided into a subnet number and a local host address. This subdivision is administered by the network authority and is generally transparent to hosts outside of the network.
Host Address	Subnet	The network authority is responsible for assigning and administering subnet numbers. Unlike the network number, the subnet number does not contain any embedded information that would allow determination of its length. This information must be configured for each host connected to the subnet.
	Local Host	The network authority or subnet administrator is responsible for assigning local host numbers.

Network Classes

The internet address is always four bytes in length, but the network and host numbers (in other words, addresses) vary in length according to the class of network. Three network classes are currently in general use and are identified by the first two bits of the network number. This diagram illustrates these network classes:

Note The DECIMAL VALUE RANGE column indicates the decimal value the address starts with when the address is written in dot notation (for example, 138.42.128.208).

Table 1-3 Network Classes

Class	Initial Bits	Description	Decimal Value Range
A	0xxx	Used for large networks. The NIC assigns a fixed value to the first byte of the address. The last three bytes are managed by the organization. The navy has a Class A address of 26, so all navy addresses are in this format: 26.nnn.nnn.nnn	0 – 127
B	10xx	Used for medium sized networks The NIC assigns a fixed value to the first two bytes of the address. The last two bytes can be managed by the organization. **Example:** My Company has a Class B address of 138.42, so all Internet addresses are in this format: 138.42.nnn.nnn	128 – 191
C	110x	Used for small networks The NIC assigns one or more fixed values to use in the first three bytes of the organization's addresses. The organization has control of the last byte. **Example:** My Company has a Class C address of 198.137.88, so all Internet addresses are in this format: 198.137.88.nn	192 – 223

Connecting to a Internet Address

The transport user initiating a connection, or initiating a connectionless data transfer, must determine the address of the destination transport user in advance. Generally, this information is acquired in one of these ways:

- The destination transport user is a standard service whose address is fixed and known throughout the network, such as a file server supporting a group of workstations on a local area network. Such addresses are generally referred to as well-known addresses.
- The destination transport user is not a well-known service, but its address is configured as part of the application program or is configured in a system database, and the application program can read it.
- A directory service or name server is used to map the name of a service or host into its address. The directory service may be located on the same host, in which case the transport user can access it directly, or it may be located on some remote host, in which case the transport user needs to know the address of the directory service.
- The user connects to a logger service whose address it knows and requests access to some particular service. The logger service spawns a process to service the user and returns its address so it can be connected to directly.

Sometimes a combination of these techniques is used.

Example

The well-known transport address may be known in advance and a name server is used to obtain the network address of a well-known service at some particular host.

It is generally not advisable to configure static information into an application program unless it is unlikely that the information will change. Therefore, the preferred approach is to use a dynamic service such as a network-wide directory service. In the Internet environment, the Domain Name System (DNS) maps global names to addresses.

In either case, the user requests a local user agent to search the distributed directory for information associated with a globally-unique name. The local user agent then contacts a server agent and requests the desired information. If the information is available, it is returned to the requesting agent and forwarded to the user. Otherwise, the user agent might be redirected to another server agent, or the server agent might contact another agent on its behalf. This process continues until it is determined that the name is unknown, or a server agent is found that has the required information.