Network Working Group
Request for Comments: RFC 1006
Obsoletes: RFC 983

Marshall T. Rose, Dwight E. Cass
Northrop Research and Technology Center

May 1987

ISO Transport Service on top of the TCP
Version: 3

Status of this Memo

This memo specifies a standard for the Internet community. Hosts on the Internet that choose to implement ISO transport services on top of the TCP are expected to adopt and implement this standard. TCP port 102 is reserved for hosts which implement this standard. Distribution of this memo is unlimited.

This memo specifies version 3 of the protocol and supersedes [RFC983]. Changes between the protocol as described in Request for Comments 983 and this memo are minor, but are unfortunately incompatible.

1. Introduction and Philosophy

The Internet community has a well-developed, mature set of transport and internetwork protocols (TCP/IP), which are quite successful in offering network and transport services to end-users. The CCITT and the ISO have defined various session, presentation, and application recommendations which have been adopted by the international community and numerous vendors. To the largest extent possible, it is desirable to offer these higher level directly in the ARPA Internet, without disrupting existing facilities. This permits users to develop expertise with ISO and CCITT applications which previously were not available in the ARPA Internet. It also permits a more graceful convergence and transition strategy from TCP/IP-based networks to ISO-based networks in the medium-and long-term.

There are two basic approaches which can be taken when "porting" an ISO or CCITT application to a TCP/IP environment. One approach is to port each individual application separately, developing local protocols on top of the TCP. Although this is useful in the short-term (since special-purpose interfaces to the TCP can be developed quickly), it lacks generality.

A second approach is based on the observation that both the ARPA Internet protocol suite and the ISO protocol suite are both layered systems (though the former uses layering from a more pragmatic perspective). A key aspect of the layering principle is that of layer-independence. Although this section is redundant for most readers, a slight bit of background material is necessary to introduce this concept.

Externally, a layer is defined by two definitions:

- a service-offered definition, which describes the services provided by the layer and the interfaces it provides to access those services; and,

- a service-required definitions, which describes the services used by the layer and the interfaces it uses to access those services.

Collectively, all of the entities in the network which co-operate to provide the service are known as the service-provider. Individually, each of these entities is known as a service-peer.

Internally, a layer is defined by one definition:

- a protocol definition, which describes the rules which each service-peer uses when communicating with other service-peers.

Putting all this together, the service-provider uses the protocol and services from the layer below to offer the its service to the layer above. Protocol verification, for instance, deals with proving that this in fact happens (and is also a fertile field for many Ph.D. dissertations in computer science).

The concept of layer-independence quite simply is:

> IF one preserves the services offered by the service-provider
>
> THEN the service-user is completely naive with respect to the protocol which the service-peers use

For the purposes of this memo, we will use the layer-independence to define a Transport Service Access Point (TSAP) which appears to be identical to the services and interfaces offered by the ISO/CCITT TSAP (as defined in [ISO8072]), but we will in fact implement the ISO TP0 protocol on top of TCP/IP (as defined in [RFC793,RFC791]), not on top of the the ISO/CCITT network protocol. Since the transport class 0 protocol is used over the TCP/IP connection, it achieves identical functionality as transport class 4. Hence, ISO/CCITT higher level layers (all session, presentation, and application entities) can operate fully without knowledge of the fact that they are running on a TCP/IP internetwork.

2. Motivation

In migrating from the use of TCP/IP to the ISO protocols, there are several strategies that one might undertake. This memo was written with one particular strategy in mind.

The particular migration strategy which this memo uses is based on the notion of gatewaying between the TCP/IP and ISO protocol suites at the transport layer. There are two strong arguments for this approach:

1. Experience teaches us that it takes just as long to get good implementations of the lower level protocols as it takes to get implementations of the higher level ones. In particular, it has been observed that there is still a lot of work being done at the ISO network and transport layers. As a result, implementations of protocols above these layers are not being aggressively pursued. Thus, something must be done "now" to provide a medium in which the higher level protocols can be developed. Since TCP/IP is mature, and essentially provides identical functionality, it is an ideal medium to support this development.

2. Implementation of gateways at the IP and ISO IP layers are probably not of general use in the long term. In effect, this would require each Internet host to support both TP4 and TCP. As such, a better strategy is to implement a graceful migration path from TCP/IP to ISO protocols for the ARPA Internet when the ISO protocols have matured sufficiently.

Both of these arguments indicate that gatewaying should occur at or above the transport layer service access point. Further, the first argument suggests that the best approach is to perform the gatewaying exactly AT the transport service access point to maximize the number of ISO layers which can be developed.

NOTE: This memo does not intend to act as a migration or intercept document. It is intended ONLY to meet the needs discussed above. However, it would not be unexpected that the protocol described in this memo might form part of an overall transition plan. The description of such a plan however is COMPLETELY beyond the scope of this memo.

Finally, in general, building gateways between other layers in the TCP/IP and ISO protocol suites is problematic, at best.

To summarize: the primary motivation for the standard described in this memo is to facilitate the process of gaining experience with higher-level ISO protocols (session, presentation, and application). The stability and maturity of TCP/IP are ideal for

providing solid transport services independent of actual implementation.

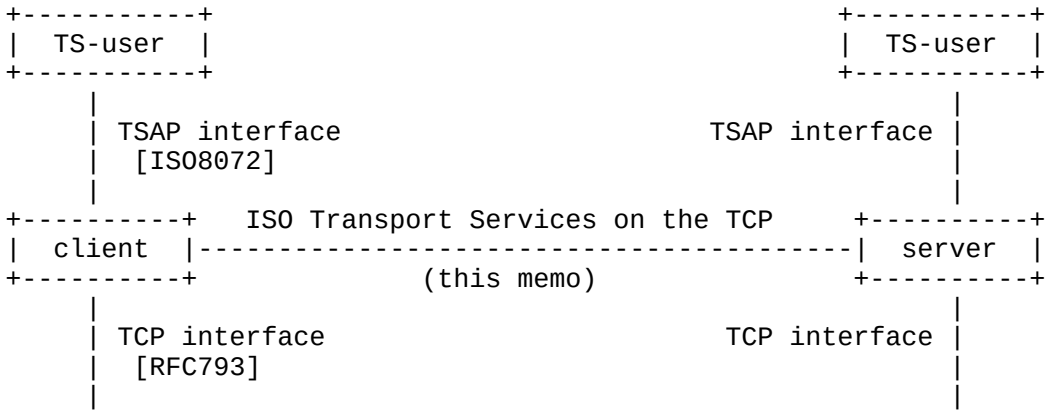
3. The Model

The [ISO8072] standard describes the ISO transport service definition, henceforth called TP.

ASIDE: This memo references the ISO specifications rather than the CCITT recommendations. The differences between these parallel standards are quite small, and can be ignored, with respect to this memo, without loss of generality. To provide the reader with the relationships:

Transport service	[ISO8072]	[X.214]
Transport protocol	[ISO8073]	[X.224]
Session protocol	[ISO8327]	[X.225]

The ISO transport service definition describes the services offered by the TS-provider (transport service) and the interfaces used to access those services. This memo focuses on how the ARPA Transmission Control Protocol (TCP) [RFC793] can be used to offer the services and provide the interfaces.



For expository purposes, the following abbreviations are used:

TS-peer a process which implements the protocol described by this memo

TS-user a process talking using the services of a TS-peer

TS-provider the black-box entity implementing the protocol described by this memo

For the purposes of this memo, which describes version 2 of the TSAP protocol, all aspects of [ISO8072] are supported with one exception:

Quality of Service parameters

In the spirit of CCITT, this is left "for further study". A future version of the protocol will most likely support the QOS parameters for TP by mapping these onto various TCP parameters.

The ISO standards do not specify the format of a session port (termed a TSAP ID). This memo mandates the use of the GOSIP specification [GOSIP86] for the interpretation of this field. (Please refer to Section 5.2, entitled "UPPER LAYERS ADDRESSING".)

Finally, the ISO TSAP is fundamentally symmetric in behavior. There is no underlying client/server model. Instead of a server listening on a well-known port, when a connection is established, the TS-provider generates an INDICATION event which, presumably the TS-user catches and acts upon. Although this might be implemented by having a server "listen" by hanging on the INDICATION event, from the perspective of the ISO TSAP, all TS-users just sit around in the IDLE state until they either generate a REQUEST or accept an INDICATION.

4. The Primitives

The protocol assumes that the TCP[RFC793] offers the following service primitives:

Events

connected	- open succeeded (either ACTIVE or PASSIVE)
connect fails	- ACTIVE open failed
data ready	- data can be read from the connection
errored	- the connection has errored and is now closed
closed	- an orderly disconnection has started

Actions

listen on port	- PASSIVE open on the given port
open port	- ACTIVE open to the given port
read data	- data is read from the connection
send data	- data is sent on the connection
close	- the connection is closed (pending data is sent)

This memo describes how to use these services to emulate the following service primitives, which are required by [ISO8073]:

Events

N-CONNECT.INDICATION
- An NS-user (responder) is notified that connection establishment is in progress

N-CONNECT.CONFIRMATION
- An NS-user (responder) is notified that the connection has been established

N-DATA.INDICATION
- An NS-user is notified that data can be read from the connection

N-DISCONNECT.INDICATION
- An NS-user is notified that the connection is closed

Actions

N-CONNECT.REQUEST
- An NS-user (initiator) indicates that it wants to establish a connection

N-CONNECT.RESPONSE
- An NS-user (responder) indicates that it will honor the request

N-DATA.REQUEST - An NS-user sends data

N-DISCONNECT.REQUEST
- An NS-user indicates that the connection is to be closed

The protocol offers the following service primitives, as defined in [ISO8072], to the TS-user:

Events

T-CONNECT.INDICATION
- a TS-user (responder) is notified that connection establishment is in progress

T-CONNECT.CONFIRMATION
- a TS-user (initiator) is notified that the connection has been established

T-DATA.INDICATION
- a TS-user is notified that data can be read from the connection

T-EXPEDITED DATA.INDICATION
- a TS-user is notified that "expedited" data can be read from the connection

T-DISCONNECT.INDICATION
- a TS-user is notified that the connection is closed

Actions

T-CONNECT.REQUEST
- a TS-user (initiator) indicates that it wants to establish a connection

T-CONNECT.RESPONSE
- a TS-user (responder) indicates that it will honor the request

T-DATA.REQUEST - a TS-user sends data

T-EXPEDITED DATA.REQUEST
- a TS-user sends "expedited" data

T-DISCONNECT.REQUEST
- a TS-user indicates that the connection is to be closed

5. The Protocol

The protocol specified by this memo is identical to the protocol for ISO transport class 0, with the following exceptions:

- for testing purposes, initial data may be exchanged during connection establishment
- for testing purposes, an expedited data service is supported
- for performance reasons, a much larger TSDU size is supported
- the network service used by the protocol is provided by the TCP

The ISO transport protocol exchanges information between peers in discrete units of information called transport protocol data units (TPDUs). The protocol defined in this memo encapsulates these TPDUs in discrete units called TPKTs. The structure of these TPKTs and their relationship to TPDUs are discussed in the next section.

PRIMITIVES

The mapping between the TCP service primitives and the service primitives expected by transport class 0 are quite straight-forward:

network service	TCP
CONNECTION ESTABLISHMENT	
N-CONNECT.REQUEST	open completes
N-CONNECT.INDICATION	listen (PASSIVE open) finishes
N-CONNECT.RESPONSE	listen completes
N-CONNECT.CONFIRMATION	open (ACTIVE open) finishes
DATA TRANSFER	
N-DATA.REQUEST	send data
N-DATA.INDICATION	data ready followed by

	read data
CONNECTION RELEASE	
N-DISCONNECT.REQUEST	close
N-DISCONNECT.INDICATION	connection closes or errors

Mapping parameters is also straight-forward:

network service	TCP
CONNECTION RELEASE	
Called address	server's IP address (4 octets)
Calling address	client's IP address (4 octets)
all others	ignored
DATA TRANSFER	
NS-user data (NSDU)	data
CONNECTION RELEASE	
all parameters	ignored

CONNECTION ESTABLISHMENT

The elements of procedure used during connection establishment are identical to those presented in [ISO8073], with three exceptions.

In order to facilitate testing, the connection request and connection confirmation TPDUs may exchange initial user data, using the user data fields of these TPDUs.

In order to experiment with expedited data services, the connection request and connection confirmation TPDUs may negotiate the use of expedited data transfer using the negotiation mechanism specified in [ISO8073] is used (e.g., setting the "use of transport expedited data transfer service" bit in the "Additional Option Selection" variable part). The default is not to use the transport expedited data transfer service.

In order to achieve good performance, the default TPDU size is 65531 octets, instead of 128 octets. In order to negotiate a smaller (standard) TPDU size, the negotiation mechanism specified in [ISO8073] is used (e.g., setting the desired bit in the "TPDU Size" variable part).

To perform an N-CONNECT.REQUEST action, the TS-peer performs an active open to the desired IP address using TCP port 102. When the TCP signals either success or failure, this results in an N-CONNECT.INDICATION action.

To await an N-CONNECT.INDICATION event, a server listens on TCP port 102. When a client successfully connects to this port, the event occurs, and an implicit N-CONNECT.RESPONSE action is performed.

NOTE: In most implementations, a single server will perpetually LISTEN on port 102, handing off connections as they are made

DATA TRANSFER

The elements of procedure used during data transfer are identical to those presented in [ISO8073], with one exception: expedited data may be supported (if so negotiated during connection establishment) by sending a modified ED TPDU (described below). The TPDU is sent on the same TCP connection as all of the other TPDUs. This method, while not faithful to the spirit of [ISO8072], is true to the letter of the specification.

To perform an N-DATA.REQUEST action, the TS-peer constructs the desired TPKT and uses the TCP send data primitive.

To trigger an N-DATA.INDICATION action, the TCP indicates that data is ready and a TPKT is read using the TCP read data primitive.

CONNECTION RELEASE

To perform an N-DISCONNECT.REQUEST action, the TS-peer simply closes the TCP connection.

If the TCP informs the TS-peer that the connection has been closed or has errored, this indicates an N-DISCONNECT.INDICATION event.

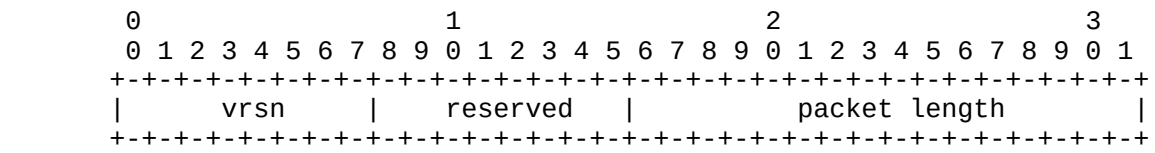
6. Packet Format

A fundamental difference between the TCP and the network service expected by TP0 is that the TCP manages a continuous stream of octets, with no explicit boundaries. The TP0 expects information to be sent and delivered in discrete objects termed network service data units (NSDUs). Although other classes of transport may combine more than one TPDU inside a single NSDU, transport class 0 does not use this facility. Hence, an NSDU is identical to a TPDU for the purposes of our discussion.

The protocol described by this memo uses a simple packetization scheme in order to delimit TPDUs. Each packet, termed a TPKT, is viewed as an object composed of an integral number of octets, of variable length.

NOTE: For the purposes of presentation, these objects are shown as being 4 octets (32 bits wide). This representation is an artifact of the style of this memo and should not be interpreted as requiring that a TPKT be a multiple of 4 octets in length.

A TPKT consists of two parts: a packet-header and a TPDU. The format of the header is constant regardless of the type of packet. The format of the packet-header is as follows:



where:

vrsn 8 bits

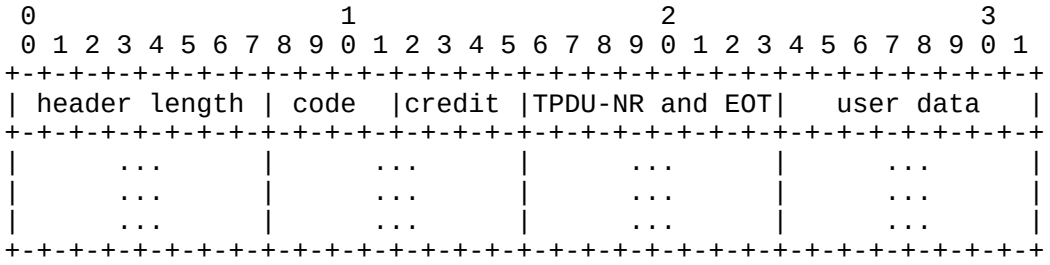
This field is always 3 for the version of the protocol described in this memo.

packet length 16 bits (min=7, max=65535)

This field contains the length of entire packet in octets, including packet-header. This permits a maximum TPDU size of 65531 octets. Based on the size of the data transfer (DT) TPDU, this permits a maximum TSDU size of 65524 octets.

The format of the TPDU is defined in [ISO8073]. Note that only TPDUs formatted for transport class 0 are exchanged (different transport classes may use slightly different formats).

To support expedited data, a non-standard TPDU, for expedited data is permitted. The format used for the ED TPDU is nearly identical to the format for the normal data, DT, TPDU. The only difference is that the value used for the TPDU's code is ED, not DT:



After the credit field (which is always ZERO on output and ignored on input), there is one additional field prior to the user data.

TPDU-NR and EOT 8 bits

Bit 7 (the high-order bit, bit mask 1000 0000) indicates the end of a TSDU. All other bits should be ZERO on output and ignored on input.

Note that the TP specification limits the size of an expedited transport service data unit (XSDU) to 16 octets.

7. Comments

Since the release of RFC983 in April of 1986, we have gained much experience in using ISO transport services on top of the TCP. In September of 1986, we introduced the use of version 2 of the protocol, based mostly on comments from the community.

In January of 1987, we observed that the differences between version 2 of the protocol and the actual transport class 0 definition were actually quite small. In retrospect, this realization took much longer than it should have: TP0 is is meant to run over a reliable network service, e.g., X.25. The TCP can be used to provide a service of this type, and, if no one complains too loudly, one could state that this memo really just describes a method for encapsulating TPO inside of TCP!

The changes in going from version 1 of the protocol to version 2 and then to version 3 are all relatively small. Initially, in describing version 1, we decided to use the TPDU formats from the ISO transport protocol. This naturally led to the evolution described above.

8. References

[GOSIP86] The U.S. Government OSI User's Committee.
"Government Open Systems Interconnection Procurement (GOSIP) Specification for Fiscal years 1987 and 1988." (December, 1986) [draft status]

[ISO8072] ISO.
"International Standard 8072. Information Processing Systems -- Open Systems Interconnection: Transport Service Definition."
(June, 1984)

[ISO8073] ISO.
"International Standard 8073. Information Processing Systems -- Open Systems Interconnection: Transport Protocol Specification."
(June, 1984)

[ISO8327] ISO.
"International Standard 8327. Information Processing Systems -- Open Systems Interconnection: Session Protocol Specification."
(June, 1984)

[RFC791] Internet Protocol.
Request for Comments 791 (MILSTD 1777)
(September, 1981)

[RFC793] Transmission Control Protocol.
Request for Comments 793 (MILSTD 1778)
(September, 1981)

[RFC983] ISO Transport Services on Top of the TCP.
Request for Comments 983
(April, 1986)

[X.214] CCITT.
"Recommendation X.214. Transport Service Definitions for Open Systems Interconnection (OSI) for CCITT Applications."
(October, 1984)

[X.224] CCITT.
"Recommendation X.224. Transport Protocol Specification for Open Systems Interconnection (OSI) for CCITT Applications." (October, 1984)

[X.225] CCITT.
"Recommendation X.225. Session Protocol Specification for Open Systems Interconnection (OSI) for CCITT Applications."
(October, 1984)