Network Working Group Brian Bidulock
INTERNET-DRAFT OpenSS7 Corporation

Expires in six months May 2001

Stream Control Transmission Protocol (SCTP)
Transport Provider Interface (TPI)
SCTP-TPI
<draft-bidulock-sctpstreams-sigtran-00.txt>

Status of this Memo

Abstract

This Internet Draft provides a specification mapping the Stream Control Transmission Protocol RFC 2960[1] into a SVR 4.2 STREAMS provider interface. The benefit of this specification includes compatibility to UNIX International TLI (Transport Layer Interface)[2], TPI (Transport Provider Interface)[2], and XOpen/XPG4 XTI (XOpen Transport Interface)[3] for maximum compatibility with OSI and IP transport provider applications based on SVR 4.2 STREAMS. In addition, this specification provides access to the features of SCTP which extend beyond those of existing transport level interfaces.

1. Introduction

In contrast with the RFC 2126 (ITOT)[4] which uses TCP to provide ISO Network Service in support of ISO Transport Service, this specification takes the approach of using the Open Group ISO Transport Provider Interface[2] as an interface to the RFC 2960 Stream Control Transmission Protocol (SCTP)[1]. This permits applications which are written to the XPG4/XTI[3] library which interfaces to the TPI[2] and socket compatibility libraries to be used without much modification for use with a SCTP Transport Provider which conforms to the SCTP-TPI.

This document provides an SCTP-specific implementation of the Transport Provider Interface and provides details concerning address formats, options and option formats, and specific IOCTLs and error handling around the framework provided for in the TPI[2].

This document describes the specification of an SCTP Transport Provider which uses the Open Group STREAMS Transport Provider Interface as an interface to Stream Control Transmission Protocol features and functions. Because of the need to select the SCTP stream upon which data is transmitted or received, it is necessary to use Version 2, Draft 2 of the Transport Provider Interface[2] as a basis for the specification, as this version incudes the **T_OPTDATA_REQ** and **T_OPTDATA_IND** primitives which are necessary for the selection of SCTP stream identifier, payload protocol identifier, and other SCTP-specific options. It is possible to use a subset of this specification which is compatible with the TLI[5] or older versions of the TPI[2], however, selection of SCTP stream for transmission becomes awkward and distinguishing streams upon reception may be impossible.

The TPI specification[2] leaves a number of implementation- and provider-specific details to be described by a particular implementation or provider. This document details the SCTP provider-specific details which are necessary for access to SCTP features and functions. In addition, it proposes some common implementation-specific managment features and functions necessary to support the global SCTP options as well as provide access to the SCTP-MIB.

Transport Primitives	Stream Msg Type	IS 8072 Transport Primitives
T_CONN_REQ	M_PROTO	T-CONNECT request
T_CONN_IND	M_PROTO	T-CONNECT indication
T_CONN_RES	M_PROTO	T-CONNECT response
T_CONN_CON	M_PROTO	T-CONNECT confirm
T_DATA_REQ	M_PROTO	T-DATA request
T_DATA_IND	M_PROTO	T-DATA indication
T_EXDATA_REQ	M_PROTO	T-EXPEDITED-DATA request
T_EXDATA_IND	M_PROTO	T-EXPEDITED-DATA indication
T_DISCON_REQ	M_PROTO	T-DISCONNECT request
T_DISCON_IND	M_PROTO	T-DISCONNECT indication
T_UNITDATA_REQ	M_PROTO	T-UNITDATA request
T_UNITDATA_IND	M_PROTO	T-UNITDATA indication
T_ORDREL_REQ	M_PROTO	not defined in ISO
T_ORDREL_IND	M_PROTO	not defined in ISO
T_BIND_REQ	M_PROTO	not defined in ISO
T_BIND_ACK	M_PCPROTO	not defined in ISO
T_UNBIND_REQ	M_PROTO	not defined in ISO
T_OK_ACK	M_PCPROTO	not defined in ISO
T_ERROR_ACK	M_PCPROTO	not defined in ISO
T_INFO_REQ	M_PCPROTO	not defined in ISO
T_INFO_ACK	M_PCPROTO	not defined in ISO
T_UDERROR_IND	M_PROTO	not defined in ISO
T_OPTMGMT_REQ	M_PROTO	not defined in ISO
T_OPTMGMT_ACK	M_PCPROTO	not defined in ISO
T_ADDR_REQ	M_PROTO	not defined in ISO
T_ADDR_ACK	M_PCPROTO	not defined in ISO
T_OPTDATA_REQ	M_PROTO	not defined in ISO
T_OPTDATA_IND	M_PROTO	not defined in ISO

Figure 1. Mapping of ISO IS8072 and RFC 2960 to Kernel-level Transport Service Primitives

2. SCTP Provider-Specific Details

2.1. SCTP Transport Addresses

The SCTP Transport Address is used in the following primitives:

Transport Primitives	Parameter Fields
T_INFO_ACK	ADDR_size
T_ADDR_ACK	LOCADDR_length, LOCADDR_offset
	REMADDR_length, REMADDR_offset
T_BIND_REQ	ADDR_length, ADDR_offset
T_BIND_ACK	ADDR_length, ADDR_offset
T_CONN_REQ	DEST_length, DEST_offset
T_CONN_CON	RES_length, RES_offset
T_CONN_IND	SRC_length, SRC_offset

Figure 2. Transport Primitives which use the SCTP Transport Address

The format of the SCTP address is as follows:

```
struct sctp_addr {
```

```
        uint16_t    port;
        uint32_t    addr[0];
    };
```

Where

port is the 16-bit local or peer SCTP port number.

addr[] is a list of 0 or more 32-bit IPv4 network addresses.

The SCTP Transport Address is interpreted by the SCTP Transport Provider in the designated primitives and according to the rules described as follows:

T_INFO_ACK

This primitive indicates the maximum size of a protocol address in the *ADDR_size* field of the primitive. SCTP providers should set this field to accomodate a number of network addresses in the *addr* list in support of multi-homed hosts. SCTP providers which do not support multihoming should set this size to 6 for IPv4 providers.

T_ADDR_ACK

This primitive indicates the locally bound protocol address and the remote connected protocol address in the *LOCADDR_length*, *LOCADDR_offset*, *REMADDR_length*, and *REMADDR_offset*, respectively. The following rules define the SCTP provider-specific interpretation of the transport address:

(1) The *LOCADDR* is the locally bound *port* and *addr* list and is the same as is indicated in the **T_BIND_ACK** primitive. For rules concerning the SCTP provider-specific interpretation of this address, see the rules for the **T_BIND_ACK** primitive below.

(2) The *REMADDR* is the remote connected protocol address and is the same as is returned in the **T_CONN_IND** or the **T_CONN_CON** primitive. For rules concerning the SCTP provider-specific interpretation of these addresses, see the rules for **T_CONN_IND** and **T_CONN_CON** below.

T_BIND_REQ

This primitive provides the protocol address to which the user is requesting that the STREAM be bound in the *ADDR_length* and *ADDR_offset* fields. The following rules define the SCTP provider-specific interpretation of the transport address:

T_BIND_ACK

This primitive indicates the protocol address to which the STREAM has been bound by the provider in the *ADDR_length* and *ADDR_offset* fields. The following rules define the SCTP provider-specific interpretation of the transport address:

(1) The **T_BIND_ACK** primitive always returns a non-zero *ADDR_length* in the primitive.

(2) If the *port* field of the address is zero, then the STREAM is bound as a "default listener" on the returned network addresses.

(3) If the *port* number is non-zero then the port number specified is that to which the stream was bound.

(4) If the *addr* list is empty, then the STREAM is bound to any and all network addresses which are available to the SCTP provider.

(5) If the *addr* list is not emply, then the STREAM is bound to only those network addresses which are present in the list.

The bound protocol address has provider-specific meaning to the SCTP Transport Provider. When the STREAM is bound as a "listener" (i.e. *CONIND_number* is non-zero), it will accept incoming **INIT** messages sent to the specified port number on any of the listed network addresses (or if there is no list, any network address). When the STREAM response to an SCTP **INIT** chunk with an SCTP **INIT-ACK** chunk, the list of bound network addresses will be placed in the address list in the SCTP **INIT-ACK** chunk.

When the STREAM is bound with *CONIND_number* set to zero, a subsequent **T_CONN_REQ** primitive will launch an SCTP **INIT** chunk with the list of bound network addresses in the address list in the SCTP **INIT** chunk.

T_CONN_REQ

This primitive provides the destination protocol address to which the user is requesting that the STREAM be connected in the *DEST_length* and *DEST_offset* fields. The following rules define the SCTP provider-specific interpretation of the transport address:

(1) The **T_CONN_REQ** primitive must always be issued by the user with a non-zero *ADDR_length* field, and the protocol address must contain at least one network address in the address list.

(2) If the *port* field in the transport address is zero, the SCTP provider will attempt to select a port.

(3) If the protocol address *addr* list has more than one network address in the list, only the first network address will be used as the destination of the SCTP **INIT** chunk which wil be issued to attempt the connection.

(4) Additional destination network addresses in the *addr* list may be used as eligible destination network addresses once the association is formed later and in conjunction with the network addresses which are returned by the peer in the SCTP **INIT-ACK** chunk.

T_CONN_CON

This primitive indicates the protocol address of the peer endpoint which is connected to the STREAM in the *RES_length* and *RES_offset* fields. The following rules define the SCTP provider-specific interpretation of the transport address:

(1) The **T_CONN_CON** primitive is always issued by the provider with a non-zero *RES_length* field, and the protocol address will contain at least one network address in the address list.

(2) The first network address in the *addr* list is the network address of the peer endpoint which responded with an SCTP **INIT-ACK** chunk and a subsequent **COOKIE-ACK** chunk confirming the connection.

(3) The remaining network addresses (if any) are addresses which were provided in the SCTP **INIT-ACK** chunk. These are the alternative network addresses to which the local host may send SCTP packets belonging to the association.

T_CONN_IND

This primitive indicates the protocol address of the peer endpoint which is attempting to connect to the STREAM in the *SRC_length* and *SRC_offset* fields. The following rules define the SCTP provider-specific interpretation of the transport address:

(1) The **T_CONN_IND** primitive is always issued by the provider with a non-zero *SRC_length* field, and the protocol address will contain at least one network address in the address list.

(2) The first network address in the *addr* list is the network address of the peer endpoint which initiated an SCTP **INIT** chunk and responded with an SCTP **COOKIE-ECHO** chunk.

(3) The remaining network addresses (if any) are addresses which were provided in the SCTP **INIT** chunk and echoed in the cookie in the **COOKIE-ECHO** chunk. These are the alternative network addresses to which the local host may send SCTP packets belonging to the association.

2.2. SCTP Options

The SCTP Provider Options are used in the following primitives:

Transport Primitives	Parameter Fields
T_INFO_ACK	OPT_size
T_OPTMGMT_REQ	OPT_length, OPT_offset
T_OPTMGMT_ACK	OPT_length, OPT_offset
T_CONN_REQ	OPT_length, OPT_offset
T_CONN_CON	OPT_length, OPT_offset
T_CONN_IND	OPT_length, OPT_offset
T_CONN_RES	OPT_length, OPT_offset
T_OPTDATA_REQ	OPT_length, OPT_offset
T_OPTDATA_IND	OPT_length, OPT_offset

Figure 3. Transport Primitives which use the SCTP Transport Options

The general format of the Transport Option is as follows:

```
struct t_opthdr {
        t_uscalar_t len;    /* Option length, incl. header  */
        t_uscalar_t level;  /* Option level                 */
```

```
        t_uscalar_t name;   /* Option name                  */
        t_uscalar_t status; /* Negotiation result           */
        /* followed by option data */
};
```

T_SCTP_NODELAY

This option is formatted as a single *t_uscalar_t* boolean flag.

Turns the Nagle algorithm off. This means that packets are always sent as soon as possible and no unnecessary delays are introduced, at the cost of more packets in the network. Expects a *t_uscalar_t* boolean flag.

This parallels the *TCP_NODELAY* option for compatibility with **TCP**. *T_SCTP_NODELAY* and *TCP_NODELAY* can be used interchangeably. This setting applies to the entire SCTP association.

This option has meaning in **T_OPTMGMT_REQ** and **T_OPTMGMT_ACK** primitives.

T_SCTP_MAXSEG

This option is formatted as a single *t_uscalar_t* which contains the maximum segment size in bytes.

This option has meaning in **T_OPTMGMT_REQ** and **T_OPTMGMT_ACK** primitives.

T_SCTP_RECVSID

This option is formatted as a single *t_uscalar_t* boolean flag.

This option has meaning in **T_OPTMGMT_REQ** and **T_OPTMGMT_ACK** primitives.

T_SCTP_RECVPPI

This option is formatted as a single *t_uscalar_t* boolean flag.

This option has meaning in **T_OPTMGMT_REQ** and **T_OPTMGMT_ACK** primitives.

T_SCTP_RECVSSN

This option is formatted as a single *t_uscalar_t* boolean flag.

This option has meaning in **T_OPTMGMT_REQ** and **T_OPTMGMT_ACK** primitives.

T_SCTP_HB

This option has meaning in **T_OPTMGMT_REQ** and **T_OPTMGMT_ACK** primitives.

T_SCTP_RTO

This option has meaning in **T_OPTMGMT_REQ** and **T_OPTMGMT_ACK** primitives.

T_SCTP_SID

This option is formatted as a single *t_uscalar_t* which contains the stream id significant to 16-bits.

This option has meaning in **T_OPTDATA_REQ**, **T_OPTDATA_IND**, **T_OPTMGMT_REQ**, and **T_OPTMGMT_ACK** primitives.

T_SCTP_PPI

This option is formatted as a single *t_uscalar_t* which contains the Payload Protocol Identifier significant to 32-bits.

This option has meaning in **T_OPTDATA_REQ**, **T_OPTDATA_IND**, **T_OPTMGMT_REQ**, and **T_OPTMGMT_ACK** primitives.

T_SCTP_SSN

This option is formatted as a single *t_uscalar_t* which contains the stream sequence number significant to 16-bits.

This option has meaning in **T_OPTDATA_REQ**, **T_OPTDATA_IND**, **T_OPTMGMT_REQ**, and **T_OPTMGMT_ACK** primitives.

T_SCTP_CORK

This option is formatted as a single *t_uscalar_t* boolean flag.

If enabled, the provider will not send out partial TSDUs. All queued partial TSDUs are sent when the option is cleared again. This is useful for prepending headers or for throughput optimization. This option cannot be combined with **T_SCTP_NODELAY**.

This parallels *TCP_CORK* for compatibility with **TCP**. *T_SCTP_CORK* and *TCP_CORK* can be used interchangeably. For **T_OPTDATA_REQ**, this setting applies to the default or specified SCTP stream as set by the *T_SCTP_SID* option.

This option has meaning in **T_OPTDATA_REQ** primitives.

T_SCTP_STREAMS

This option is formated as follows:

```
struct sctp_opt_streams {
        uint16_t    i_streams;
        uint16_t    o_streams;
};
```

Where

i_streams

indicates the maximum or actual number of inbound SCPT streams.

o_streams

indicates the requested or actual number of outbound SCTP streams.

This option has meaning in **T_CONN_REQ**, **T_CONN_CON**, **T_CONN_IND**, **T_OPTMGMT_REQ**, and **T_OPTMGMT_ACK** primitives.

2.3. SCTP Errors

3. SCTP Implementation-Specific Details

3.1. SCTP IOCTLs

A number of SCTP Transport Provider IO Controls (IOCTLs) are included in the OpenSS7 implementation of the SCTP Transport Provider to permit the portable control of system-wide parameters and statistics. These IOCTLs are as follows:

4. SCTP Transport Provider Interface

4.1. STREAM Management

4.1.1. Capabilities

4.1.1.1. T_INFO_REQ, T_INFO_ACK

The TPI specifies that the Transport Provider reply to a **T_INFO_REQ** primitive with a **T_INFO_ACK** primitive. Although there are a number of alternatives in the responding **T_INFO_ACK** primitive, the SCTP Transport Provider will respond with a smaller set of responses as defined here. The **T_INFO_ACK** primitive will have values populated as follows:

PRIM_type

This indicates the primitive type and is always set to **T_INFO_ACK**.

TSDU_size

The size of the TSDU (Transport Service Data Unit) should be set to -1 to indicate that the provider supports the concept of a TSDU and that there is no limit on the maximum size of a TSDU.

ETSDU_size

The size of the ETSDU (Expedited Transport Service Data Unit) should be set to -1 to indicate that the provider support the concept of a ETSDU and that there is no limit on the maximum size of a ETSDU.

CDATA_size

The size of CDATA (Connection Data) should be set to -1 to indicate that the provider supports the transfer of data with connect primitives and that there is no limit on the maximum amount of data sent with the connect primitives.

DDATA_size

The size of DDATA (Disconnect Data) should be set to -1 to indicate that the provider supports the transfer of data with disconnect primitives and that there is no limit on the maximum amount of data sent with the disconnect primitives. (Note: this only really has practical use with orderly release primitives **T_ORDREL_REQ** and **T_ORDREL_IND**.)

ADDR_size

If the provider has a restriction on the maximum number of IP addresses which can be provided in an SCTP endpoint address, this size should be set to reflect that value. Otherwise, this value should be set to -1 to indicate that there is no limit on the number of IP addresses which are provided with any primitive.

OPT_size

If the provider has a restriction on the maximum number of options which can be provided in an operation, this size should be set to reflect that value. Otherwise, this value should be set to -1 to indicate that there is no limit on the number of options which are provided with any primitive.

TIDU_size

This should be set to the system-tunable system limit on the maximum size of a STREAMS message. It might be possible to set this value to the current maximum size of the data in a DATA chunk which will currently fit within the path MTU, however, the provider cannot reject TIDUs which are larger than this dynamic value but which are smaller than the value first advertized when the STREAM was opened.

SERV_type

This field should be set to **T_COTS_ORD** to indicate that the SCTP provider is connection oriented with orderly release support.

CURRENT_state

This field should be set to the TPI provider state.

PROVIDER_flag

This field should NOT have **SENDZERO** set (because SCTP does not support the sending of zero-length TSDUs in general) and should have **XPG4_1** set to indicate support for XPG4 semantics (**T_ADDR_REQ** and **T_ADDR_ACK**).

4.1.2. Binding and Listening

Once a STREAM is opened the provider places the STREAM in the **TS_UNBND** state and the STREAM is available for binding with the **T_BIND_REQ** primitive and will be placed in the **TS_BIND** state. A STREAM which is in the **TS_BIND** state may be unbound again using the **T_UNBIND_REQ** primitive and returned to the **TS_UNBND** state. Unlike BSD sockets, this permits a STREAM to be reused for another connection without closing and reopening another STREAM.

4.1.2.1. T_BIND_REQ, T_BIND_ACK

STREAMS must be bound to a local protocol address before they can become involved in a connection and used for data transfer. The semantics for binding and unbinding STREAMS are identical to those provided in the Transport Provider Interface. The only specifics of the SCTP implementation lie in the format and interpretation of the transport address. The **T_BIND_REQ** primtive provides an opaque protocol address as indicated by *ADDR_length* and *ADDR_offset* which in the SCTP implementation has the following format:

```
typedef struct sctp_addr {
      uint16_t    port;
      uint32_t    addr[0];
} sctp_addr_t;
```

Where

port indicates the 16-bit (host order) port number to which to bind; and,

addr indicates an array of 32-bit IPv4 addresses (host order) to which to bind.

The transport address will be interpreted according to the following rules by the SCTP Transport Provider:

(1) If the transport address is not provided (i.e. *ADDR_length* is zero), the SCTP provider will assign an available port number from the range of port numbers available to SCTP and will bind the STREAM to any and all IPv4 addresses which are available to the SCTP Transport Provider.

(2) If the transport address is provided, but does not contain any IPv4 addresses, the SCTP Transport Provider will bind the STREAM to any and all IPv4 addresses which are available to the SCTP Transport Provider.

(3) If the transport address is provided with the *port* set to zero, and the *CONIND_number* is also set to zero, the SCTP TP will assign a port number.

(4) If the transport address is provided with the *port* set to zero, and the *CONIND_number* is non-zero, the SCTP TP will treat the STREAM as a "default listener." This *default listener* will receive indications for any port number on the bound IPv4 addresses which are not otherwise bound.

One and only one STREAM may be bound as a *default listener* to any given IPv4 address. If the STREAM is bound as a *default listener* and no IPv4 addresses are provided in the transport address, the STREAM will receive all SCTP messages to which no other STREAM is bound.

(5) All valid IPv4 addresses may be bound to a SCTP Transport Provider STREAM. This includes *broadcast* and *multicast* addresses; however, to bind to a *broadcast* or *multicast* address may require that the STREAM was opened with sufficient access permissions to allow binding of these addresses. *Broadcast* and *multicast* addresses will only be used by the SCTP Transport Provider for establishment of an SCTP association (INIT chunks) and will not be used for normal transmission or retransmission of chunks.

Other than the above rules, the TPI semantics of the **T_BIND_REQ** and **T_BIND_ACK** primitives are preserved, particularly with regards to the treatment of the *CONIND_number* field.

Considering the SCTP Transport Provider interpretation of the transport address, the following errors may be returned in a **T_ERROR_ACK** primitive in response to a problematic **T_BIND_REQ** primitive:

TACCES

This indicates that the user did not have proper permissions for the use of the requested address. This includes the following reasons:

(1) The user attempted to bind to a port number within the UNIX protected port number range (1-1024) and did not have sufficient priviledge.

(2) The user attempted to bind to a *broadcast* or *multicast* address and did not have sufficient priviledge.

(3) The user attempted to bind a *default listener* (i.e. to bind to port 0) and did not have sufficient priviledge.

TADDRBUSY

This indicates that the requested address was in use. This includes the following reasons:

(1) The SCTP TP only permits one STREAM to be a "listener" (i.e. with a non-zero *CONIND_number*) for a given port number IPv4 address combination.

The **T_BIND_REQ** will be rejected it the user attempts to bind a second STREAM with a non-zero *CONIND_number* to the same port number and IPv4 address combination to which another STREAM is already bound with a non-zero *CONIND_number*.

(2) The SCTP TP only permits one STREAM to be a "default listener" (i.e. listening on all ports) for a given IPv4 address.

The **T_BIND_REQ** will be rejected if the user attempts to bind a second "default listener" with a zero *port* number and a non-zero *CONIND_number* to the same IPv4 address to which another STREAM is already bound with a zero *port* number and a non-zero *CONIND_number*.

TBADADDR

This indicates that the protocol address was in an incorrect format or the address contained invalid information. It is not intended to indicate protocol errors. This may include the following reasons:

(1) The size of the address as given by *ADDR_length* is not the size of the **sctp_addr** structure plus a positive integer number of 4-byte IPv4 addresses.

(2) The format of the **sctp_addr** structure is correct, but one or more of the IPv4 addresses provided are not valid IPv4 addresses (e.g. 0.1.1.1 or 0.0.0.0).

(Note: the SCTP TP does not permit binding to address 0.0.0.0)

TNOADDR

This indicates that the transport provider could not allocate an address. This may include the following reasons:

(1) The user has requested that the SCTP Transport Provider provide a transport address with *CONIND_number* equal to zero, and the SCTP TP is unable to assign a port number because all the port numbers within the assignable range are in use.

TOUTSTATE

The primitive would place the transport interface out of state.

TSYSERR

A system error occurred and the UNIX system error is indicated in the primitive. This may include the following errors:

ENOMEM

There was insufficient memory resources to complete the operation.

EFAULT

A recoverable internal software error has occured.

4.1.2.2. T_UNBIND_REQ

STREAMS which are in the **TS_IDLE** state (i.e. those which are not currently involved in an SCTP association) can be unbound at any time using the **T_UNBIND_REQ** primitive. As provided for the TPI specification, this primitive permits the unbinding, or reversal of the bind operation.

There are no SCTP Transport Provider specifics concerning this primitive.

4.1.3. Options Management

4.1.3.1. T_OPTMGMT_REQ, T_OPTMGMT_ACK

4.2. STREAM Connection

4.2.1. Client

4.2.1.1. T_CONN_REQ, T_CONN_CON, T_DISCON_IND

4.2.2. Server

4.2.2.1. T_CONN_IND, T_CONN_REQ, T_DISCON_REQ

4.3. STREAM Data Transfer

4.3.1. Sending Data

4.3.1.1. T_DATA_REQ, T_EXDATA_REQ, T_OPTDATA_REQ

4.3.2. Receiving Data

4.3.2.1. T_DATA_IND, T_EXDATA_IND, T_OPTDATA_IND

4.4. STREAM Release

4.4.1. Disconnection

4.4.1.1. T_DISCON_REQ, T_DISCON_IND

4.4.2. Orderly Release

4.4.2.1. T_ORDREL_REQ, T_ORDREL_IND

5. Mapping of SCTP states to TPI states

TPI State	SCTP State
TS_UNBND	SCTP_CLOSED
TS_WACK_BREQ	SCTP_CLOSED
TS_WACK_UREQ	SCTP_CLOSED or SCTP_LISTEN
TS_IDLE	SCTP_CLOSED or SCTP_LISTEN
TS_WACK_OPTREQ	SCTP_CLOSED or SCTP_LISTEN
TS_WACK_CREQ	SCTP_COOKIE_WAIT
	SCTP_COOKIE_ECHOED
TS_WCON_CREQ	SCTP_COOKIE_WAIT
	SCTP_COOKIE_ECHOED
TS_WRES_CIND	SCTP_LISTEN
TS_WACK_CRES	SCTP_ESTABLISHED
TS_DATA_XFER	SCTP_ESTABLISHED
TS_WIND_ORDREL	SCTP_SHUTDOWN_PENDING
	SCTP_SHUTDOWN_SENT
TS_WREQ_ORDREL	SCTP_SHUTDOWN_RECEIVED
TS_WACK_DREQ6	SCTP_CLOSED or SCTP_LISTEN
TS_WACK_DREQ7	SCTP_LISTEN
TS_WACK_DREQ9	SCTP_ESTABLISHED
TS_WACK_DREQ10	SCTP_SHUTDOWN_PENDING
	SCTP_SHUTDOWN_SENT
TS_WACK_DREQ11	SCTP_SHUTDOWN_RECEIVED
TS_NOSTATES	SCTP_UNREACHABLE

Figure 4. Mapping of SCTP States to TPI States

6. Use of XTI with SCTP protocol

6.1. Introduction

This section describes the protocol-specific information that is relevant for SCTP transport providers. It also defines data structures and constants required for SCTP transport providers which are exposed through the <xti_sctp.h> header file.

6.2. Protocol Features

T_MORE Flag and TSDUs

The notion of TSDU is supported by SCTP transport provider, so the T_MORE flag is interpreted correctly by the SCTP transport provider.

Expedited Data

SCTP does not have a notion of expedited data in a sense comparable to ISO expedited data. SCTP defines an out-of-order delivery mechanism which can be implemented similar to TCP's urgent mechanism, by which DATA chunks which are marked for out-of-order delivery could be transmitted in advance of normal in-order deliver DATA chunks on a given stream. See the SCTP Draft Standard for more detailed information.

Orderly Release

The orderly release functions **t_sndrel()** and **t_rcvrel()** which were defined to support the orderly release facility of TCP will also be used for the orderly release facility of SCTP. The specification states that only established connections may be closed with orderly release: that is, on an endpoint in T_DATAXFER or T_INREL state.

Abortive Release

Functions **t_sdnddis()** and **t_rcvdis()** may be used to perform abortive release over SCTP transport. However, their use is not recommended as the abortive release primitive (ABORT chunk) is not transmitted reliably by the SCTP protocol.

Connection Establishment

Unlike TCP, SCTP partially allows the possibility of refusing a connection indication. This is peformed by sending an ABORT chunk instead of a COOKIE-ACK chunk in response to a COOKIE-ECHO. Therfore, unlike TCP, **t_listen()** and **t_accept()** have similar semantics to that for ISO providers.

Connection Release

After a connection has been released, the local address(es) bound to the endpoint may be qualified with the local IP address (rather than having a wildcard IP address). Also, the port number itself may have been changed during the **t_accept()** processing. If the endpoint is not being reused immediately then it is recommended that it should be unbound or closed so that other users can successfully bind to the address.

6.3. Options

Options are formatted according to the structure **t_opthdr** as described in *The Use of Options in XTI*[3]. A transport provider compliant to this specification supports none, all or any subset of the options defined in *SCTP-Level Options* and *IP-Level Options*[3]. An implementation may restrict the use of any of these options by offering them only in privileged or read-only mode.

6.3.1. SCTP-Level Options

The protocol level is **T_INET_SCTP**. For this level, *SCTP-Level Options* shows the options that are defined.

Option Name	Type of Option Value	Legal Option Value
T_SCTP_NODELAY	t_uscalar_t	T_YES/T_NO
T_SCTP_MAXSEG	t_uscalar_t	T_YES/T_NO
T_SCTP_CORK	t_uscalar_t	T_YES/T_NO
T_SCTP_SID	t_uscalar_t	stream id
T_SCTP_PPI	t_uscalar_t	payld proto id
T_SCTP_RECVSID	t_uscalar_t	T_YES/T_NO
T_SCTP_RECVPPI	t_uscalar_t	T_YES/T_NO
T_SCTP_RECVSSN	t_uscalar_t	T_YES/T_NO
T_SCTP_HB	struct t_sctp_hb	(see text)
T_SCTP_RTO	struct t_sctp_rto	(see text)

Table 4. SCTP-level Options

These options do **not** have end-to-end significance. The may be negotiated in all XTI states except **T_UNBIND** and T_UNINIT. They are read-only in state **T_UNBND**. See *The Use of Options in XTI*[3] for the differences between options that have end-to-end significance and those that do not.

6.3.2. Absolute Requirements

A request for T_SCTP_NODELAY and a request to activate T_SCTP_KEEPALIVE is an absolute requirement. T_SCTP_MAXSEG is a read-only option.

6.3.3. Further Remarks

T_SCTP_KEEPALIVE

If this option is set, the heartbeat mechanism of SCTP is activated on all destination addresses to monitor idle

destination addresses which may no longer be reachable. If a destination has been idle for the last heartbeat timeout interval, a HEARTBEAT chunk is sent to check if the destination is still alive or broken.

HEARTBEAT chunks are an explicit feature of SCTP, and this practice is part of the SCTP Standard. The option value consists of a structure **t_sctp_hb** declared as:

```
typedef struct sctp_hb
    struct sctp_addr hb_dest;  /* destination address      */
    t_scalar_t       hb_onoff; /* activation flag          */
    t_scalar_t       hb_itvl;  /* interval in milliseconds */
} t_sctp_hb_t;
```

Legal values for the field *hb_dest* is a valid IP address which indicates the destination addresses of the SCTP association for which the heartbeat parameters are to be set.

Legal values for the field *hb_onoff* are:

T_NO switch the heartbeat for destination *hb_dest* off

T_YES activate the heartbeat for destination *hb_dest*

The field *hb_itvl* determines the frequency of HEARTBEAT chunks being sent, in milliseconds. The transport user can request the default value by setting the field to **T_UNSPEC**. The default is implementation-dependent, but at least 60 seconds (see the referenced RFC 2960). Legal values for this field are **T_UNSPEC** and all positive numbers.

The timeout value is not an absolute requirement. The implementation may pose upper and lower limits to this value. Requests that fall short of the lower limit may be negotiated to the lower limit.

The use of this option might be restricted to priviledged users.

T_SCTP_MAXSEG

This option is read-only. It is used to retrieve the maximum SCTP segment size.

T_SCTP_NODELAY

Under some circumstances, SCTP sends data as soon as iti is presented. When outstanding data has not yet been acknowledged, it gathers small amounts of output to be sent in a single packet once an acknowledgement is received. For a small number of clients, such as window systems (for example, MIT X Window System) that send a stream of mouse events which receive no replies, this packetization may cause significant delays. T_SCTP_NODELAY is used to defeat this algorithm. Legal option values are **T_YES** ("don't delay") and **T_NO** ("delay").

T_SCTP_RTO

6.4. Functions

t_accept()

Issuing a **t_accept**() assigns an already established connection to *resfd*. Since user data may be exchanged during the connection establishment phase, *call->udata.len* can be larger than 0.

When (*resfd != fd*), the function **t_accept**() is recommended to be called with *resfd* in **T_UNBND** state, though an endpoint which is bound to any local address (in **T_IDLE**) state) can also be used.

After **t_accept**() completes, the endpoint *resfd* will represent a connected SCTP endpoint whose complete binding essentially has both local and remote address components.

If file descriptor *resfd* was unbound before calling **t_accept**(), after the call completes its local address binding would be to the same protocol address bound to *fd*. If file descriptor *resfd* was bound to a local address before calling **t_accept**(), that local address binding is dissolved and the local address part of the binding after **t_accept**() completes would become the same as the address bound to *fd*.

If options with end-to-end significance (T_IP_OPTIONS, T_IP_TOS) are to be sent with the connection confirmation, the values of these options must be set with **t_optmgmt**() before the T_LISTEN event occurs. When the transport user detects a T_LISTEN, SCTP has already established the connection. Association-related options passed with **t_accept**() become effective at once, but since the connection is already established, they are

transmitted with subsequent IP datagrams sent out in the T_DATAXFER state.

t_bind() The *addr* field of the **t_bind** structure represents the local socket; that is, an address which specifically includes a port identifier.

Some implementations treat port number 0 as a request to bind to any unused port. Other than that value, a port number part of the binding is specific. The IP address part of the binding can represent a single IP address or a list of IP addresses or a wildcard binding to an address that could represent multiple IP addresses that are legal for the host.

t_close()The **t_close**() call will result in a **close** call on the descriptor of this XTI communication endpoint. If there are no other descriptors in this process or any other process which reference this communication endpoint, the **close**() call will perform an orderly connection termination according to the rules of a SCTP SHUTDOWN call on this connection endpoint as specified in standard RFC 2960[1]. If the XTI_LINGER options is supported and is used to enable the *linger* option, the linger time will affect the time an implementation lingers in the execution of **t_close**() or **close**(). A linger time of 0 specified with the XTI_LINGER option may cause an abortive release of a SCTP connection, resulting in lost data.

t_connect()

The *sndcall->addr* struct specifies the remote socket. In the present version, the returned address set in *rcvcall->addr* will have the same value. Since user data can be exchanged during the connection establishment phase, *sndcall->udata.len* can be set to non-zero.

Note that the likely the peer SCTP, and not the peer transport user, will confirm the connection.

t_listen()

Upon successful return, **t_lister**() indicates an connection indication and not an existing connection. Since user data may be exchanged during the connection establishment phase, *sndcall->udata.len* may be non-zero before the call to **listen**(). The *call->addr* structure contains the remote calling socket.

t_open()**t_open**() is called as the first step in the initialization of a transport endpoint. This function returns various default characteristics of the underlying transport protocol by setting fields in the *t_info* structure. The following should be the values returned by the call to **t_open**() and **t_getinfo**() with the indicated transport provider.

Parameters	Before call	After call
name	x	/
oflag	x	/
info->addr	/	x
info->options	/	x
info->tsdu	/	T_INFINITE
info->etsdu	/	T_INFINITE
info->connect	/	x
info->discon	/	0
info->servtype	/	T_COTS_ORD
info->flags	/	0

Table 5. t_info parameters
'x' equals T_INVALID (-2) or an integral number greater than zero

t_rcv() The T_MORE flag should not be ignored if normal data is delivered. Out-of-order data chunks are received with the T_EXPEDITED flag set. If the buffer supplied by the user is too small to hold all the data, the T_MORE flag will be set, indicating that more data still remains to be read.

t_rcvconnect()

Since user data can be exchanged during the connection establishment phase, *call->udata.maxlen* can be set to non-zero before the call to this function. On return, the *call->addr* structure contains the address of the remote calling endpoint.

t_rcvdis()

Since data may not be sent with a disconnect, the *discon->udata* structure will not be meaningful.

t_snd() The T_MORE flag should be used. If **t_snd**() is called with the T_EXPEDITED flag set, the data will be sent out-of-order on the specified SCTP stream id.

t_snddis()
Sincde data may not be sent with a disconnect, *call->udata.len* must be set to zero.

t_sndudata()
Be aware that the maximum size of a connectionless-mode TSDU varies among implementations.

Security Considerations

There are no security considerations for this draft.

Acknowledgements

References

1. Randall Stewart, Qiaobing Xie, Ken Morneault, Chip Sharp, Hanns Juergen Schwarzbauer, Tom Taylor, Ian Rytina, Hallewar Kalla, Lixia Zhang, and Vern Paxson, “Stream Control Transmission Protocol (SCTP),” RFC 2960, The Internet Society (February 2000).
2. Open Group, “Transport Provider Interface Specification,” TPI Version 2, Draft 2, Open Group Publication (1999). http://www.opengroup.org/onlinepubs/
3. X Programmer’s Group, “XOpen Transport Interface,” Revision 1.0, X Programmer’s Group (no date).
4. Y. Pouffray and A. Young, “ISO Transport Service on top of TCP (ITOT),” RFC 2126, The Internet Society (March 1997).
5. Unix International, “Transport Provider Interface Specification,” Revision 1.5, Unix International Press (December 10, 1992). (defunct)

Author’s Addresses

Brian F. G. Bidulock
OpenSS7 Corporation
4701 Preston Park Boulevard, Suite 424
Plano, TX 75093
USA

Tel: +1-972-839-4489
EMail: bidulock@openss7.org

This Internet Draft expires September 2001.

List of Illustrations

Table of Contents