Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'sctp.7' command

$ man sctp.7

SCTP(7) Linux Programmer's Manual SCTP(7)

NAME

sctp - SCTP protocol.

SYNOPSIS

#include <sys/socket.h>

#include <netinet/in.h>

#include <netinet/sctp.h>

sctp_socket = socket(PF_INET, SOCK_STREAM, IPPROTO_SCTP);

sctp_socket = socket(PF_INET, SOCK_SEQPACKET, IPPROTO_SCTP);

DESCRIPTION

This is an implementation of the SCTP protocol as defined in RFC2960 and RFC3309. It is a message oriented, reliable transport protocol with direct support for multihoming that runs on top of ip(7), and supports both v4 and v6 versions.

Like TCP, SCTP provides reliable, connection oriented data delivery with congestion control. Unlike TCP, SCTP also provides message bound? ary preservation, ordered and unordered message delivery, multi-stream? ing and multi-homing. Detection of data corruption, loss of data and duplication of data is achieved by using checksums and sequence num? bers. A selective retransmission mechanism is applied to correct loss or corruption of data.

This implementation supports a mapping of SCTP into sockets API as de? fined in the draft-ietf-tsvwg-sctpsocket-10.txt(Sockets API extensions

for SCTP). Two styles of interfaces are supported.

A one-to-many style interface with 1 to MANY relationship between socket and associations where the outbound association setup is im? plicit. The syntax of a one-to-many style socket() call is

sd = socket(PF_INET, SOCK_SEQPACKET, IPPROTO_SCTP);

A typical server in this style uses the following socket calls in se? quence to prepare an endpoint for servicing requests.

1. socket()
2. bind()
3. listen()
4. recvmsg()
5. sendmsg()
6. close()

A typical client uses the following calls in sequence to setup an asso? ciation with a server to request services.

1. socket()
2. sendmsg()
3. recvmsg()
4. close()

A one-to-one style interface with a 1 to 1 relationship between socket and association which enables existing TCP applications to be ported to SCTP with very little effort. The syntax of a one-to-one style socket() call is

sd = socket(PF_INET, SOCK_STREAM, IPPROTO_SCTP);

A typical server in one-to-one style uses the following system call se? quence to prepare an SCTP endpoint for servicing requests:

1. socket()
2. bind()
3. listen()
4. accept()

The accept() call blocks until a new association is set up. It returns with a new socket descriptor. The server then uses the new socket de? scriptor to communicate with the client, using recv() and send() calls

to get requests and send back responses. Then it calls

5. close()

to terminate the association. A typical client uses the following sys?

tem call sequence to setup an association with a server to request ser?

vices:

1. socket()

2. connect()

After returning from connect(), the client uses send() and recv() calls

to send out requests and receive responses from the server. The client

calls

3. close()

to terminate this association when done.

ADDRESS FORMATS

SCTP is built on top of IP (see ip(7)). The address formats defined by

ip(7) apply to SCTP. SCTP only supports point-to-point communication;

broadcasting and multicasting are not supported.

SYSCTLS

These variables can be accessed by the /proc/sys/net/sctp/* files or

with the sysctl(2) interface. In addition, most IP sysctls also apply

to SCTP. See ip(7).

Please check kernel documentation for this, at Documentation/network?

ing/ip-sysctl.txt.

STATISTICS

These variables can be accessed by the /proc/net/sctp/* files.

assocs Displays the following information about the active associa?

tions. assoc ptr, sock ptr, socket style, sock state, associa?

tion state, hash bucket, association id, bytes in transmit

queue, bytes in receive queue, user id, inode, local port, re?

mote port, local addresses and remote addresses.

eps Displays the following information about the active endpoints.

endpoint ptr, sock ptr, socket style, sock state, hash bucket,

local port, user id, inode and local addresses.

snmp Displays the following statistics related to SCTP states, pack?

ets and chunks.

SctpCurrEstab

The number of associations for which the current state is either ESTABLISHED, SHUTDOWN-RECEIVED or SHUTDOWN-PENDING.

SctpActiveEstabs

The number of times that associations have made a direct transi? tion to the ESTABLISHED state from the COOKIE-ECHOED state. The upper layer initiated the association attempt.

SctpPassiveEstabs

The number of times that associations have made a direct transi? tion to the ESTABLISHED state from the CLOSED state. The remote endpoint initiated the association attempt.

SctpAborteds

The number of times that associations have made a direct transi? tion to the CLOSED state from any state using the primitive 'ABORT'. Ungraceful termination of the association.

SctpShutdowns

The number of times that associations have made a direct transi? tion to the CLOSED state from either the SHUTDOWN-SENT state or the SHUTDOWN-ACK-SENT state. Graceful termination of the associ? ation.

SctpOutOfBlues

The number of out of the blue packets received by the host. An out of the blue packet is an SCTP packet correctly formed, in? cluding the proper checksum, but for which the receiver was un? able to identify an appropriate association.

SctpChecksumErrors

The number of SCTP packets received with an invalid checksum.

SctpOutCtrlChunks

The number of SCTP control chunks sent (retransmissions are not included). Control chunks are those chunks different from DATA.

SctpOutOrderChunks

The number of SCTP ordered data chunks sent (retransmissions are

not included).

SctpOutUnorderChunks

The number of SCTP unordered chunks(data chunks in which the U bit is set to 1) sent (retransmissions are not included).

SctpInCtrlChunks

The number of SCTP control chunks received (no duplicate chunks included).

SctpInOrderChunks

The number of SCTP ordered data chunks received (no duplicate chunks included).

SctpInUnorderChunks

The number of SCTP unordered chunks(data chunks in which the U bit is set to 1) received (no duplicate chunks included).

SctpFragUsrMsgs

The number of user messages that have to be fragmented because of the MTU.

SctpReasmUsrMsgs

The number of user messages reassembled, after conversion into DATA chunks.

SctpOutSCTPPacks

The number of SCTP packets sent. Retransmitted DATA chunks are included.

SctpInSCTPPacks

The number of SCTP packets received. Duplicates are included.

SOCKET OPTIONS

To set or get a SCTP socket option, call getsockopt(2) to read or set? sockopt(2) to write the option with the option level argument set to SOL_SCTP.

SCTP_RTOINFO.

This option is used to get or set the protocol parameters used to initialize and bound retransmission timeout(RTO). The struc? ture sctp_rtoinfo defined in /usr/include/netinet/sctp.h is used to access and modify these parameters.

SCTP_ASSOCINFO

This option is used to both examine and set various association and endpoint parameters. The structure sctp_assocparams defined in /usr/include/netinet/sctp.h is used to access and modify these parameters.

SCTP_INITMSG

This option is used to get or set the protocol parameters for the default association initialization. The structure sctp_initmsg defined in /usr/include/netinet/sctp.h is used to access and modify these parameters.

Setting initialization parameters is effective only on an uncon? nected socket (for one-to-many style sockets only future associ? ations are effected by the change). With one-to-one style sock? ets, this option is inherited by sockets derived from a listener socket.

SCTP_NODELAY

Turn on/off any Nagle-like algorithm. This means that packets are generally sent as soon as possible and no unnecessary delays are introduced, at the cost of more packets in the network. Ex? pects an integer boolean flag.

SCTP_AUTOCLOSE

This socket option is applicable to the one-to-many style socket only. When set it will cause associations that are idle for more than the specified number of seconds to automatically close. An association being idle is defined an association that has NOT sent or received user data. The special value of 0 indicates that no automatic close of any associations should be performed. The option expects an integer defining the number of seconds of idle time before an association is closed.

SCTP_SET_PEER_PRIMARY_ADDR

Requests that the peer mark the enclosed address as the associa? tion primary. The enclosed address must be one of the associa? tion's locally bound addresses. The structure sctp_setpeerprim

defined in /usr/include/netinet/sctp.h is used to make a set peer primary request.

SCTP_PRIMARY_ADDR

Requests that the local SCTP stack use the enclosed peer address as the association primary. The enclosed address must be one of the association peer's addresses. The structure sctp_prim de? fined in /usr/include/netinet/sctp.h is used to make a get/set primary request.

SCTP_DISABLE_FRAGMENTS

This option is a on/off flag and is passed an integer where a non-zero is on and a zero is off. If enabled no SCTP message fragmentation will be performed. Instead if a message being sent exceeds the current PMTU size, the message will NOT be sent and an error will be indicated to the user.

SCTP_PEER_ADDR_PARAMS

Using this option, applications can enable or disable heartbeats for any peer address of an association, modify an address's heartbeat interval, force a heartbeat to be sent immediately, and adjust the address's maximum number of retransmissions sent before an address is considered unreachable. The structure sctp_paddrparams defined in /usr/include/netinet/sctp.h is used to access and modify an address's parameters.

SCTP_DEFAULT_SEND_PARAM

Applications that wish to use the sendto() system call may wish to specify a default set of parameters that would normally be supplied through the inclusion of ancillary data. This socket option allows such an application to set the default sctp_sndr? cvinfo structure. The application that wishes to use this socket option simply passes in to this call the sctp_sndrcvinfo struc? ture defined in /usr/include/netinet/sctp.h. The input parame? ters accepted by this call include sinfo_stream, sinfo_flags, sinfo_ppid, sinfo_context, sinfo_timetolive. The user must set the sinfo_assoc_id field to identify the association to affect

if the caller is using the one-to-many style.

SCTP_EVENTS

This socket option is used to specify various notifications and ancillary data the user wishes to receive. The structure sctp_event_subscribe defined in /usr/include/netinet/sctp.h is used to access or modify the events of interest to the user.

SCTP_I_WANT_MAPPED_V4_ADDR

This socket option is a boolean flag which turns on or off mapped V4 addresses. If this option is turned on and the socket is type PF_INET6, then IPv4 addresses will be mapped to V6 rep? resentation. If this option is turned off, then no mapping will be done of V4 addresses and a user will receive both PF_INET6 and PF_INET type addresses on the socket.

By default this option is turned on and expects an integer to be passed where non-zero turns on the option and zero turns off the option.

SCTP_MAXSEG

This socket option specifies the maximum size to put in any out? going SCTP DATA chunk. If a message is larger than this size it will be fragmented by SCTP into the specified size. Note that the underlying SCTP implementation may fragment into smaller sized chunks when the PMTU of the underlying association is smaller than the value set by the user. The option expects an integer.

The default value for this option is 0 which indicates the user is NOT limiting fragmentation and only the PMTU will effect SCTP's choice of DATA chunk size.

SCTP_STATUS

Applications can retrieve current status information about an association, including association state, peer receiver window size, number of unacked data chunks, and number of data chunks pending receipt. This information is read-only. The structure sctp_status defined in /usr/include/netinet/sctp.h is used to

access this information.

SCTP_GET_PEER_ADDR_INFO

Applications can retrieve information about a specific peer ad? dress of an association, including its reachability state, con? gestion window, and retransmission timer values. This informa? tion is read-only. The structure sctp_paddrinfo defined in /usr/include/netinet/sctp.h is used to access this information.

SCTP_GET_ASSOC_STATS

Applications can retrieve current statistics about an associa? tion, including SACKs sent and received, SCTP packets sent and received. The complete list can be found in /usr/in? clude/netinet/sctp.h in struct sctp_assoc_stats.

AUTHORS

Sridhar Samudrala <sri@us.ibm.com>

SEE ALSO

socket(7), socket(2), ip(7), bind(2), listen(2), accept(2), connect(2), sendmsg(2), recvmsg(2), sysctl(2), getsockopt(2), sctp_bindx(3), sctp_connectx(3), sctp_sendmsg(3), sctp_sendv(3), sctp_send(3), sctp_recvmsg(3), sctp_recvv(3), sctp_peeloff(3), sctp_getladdrs(3), sctp_getpaddrs(3), sctp_opt_info(3).

RFC2960, RFC3309 for the SCTP specification.