Linux SCTP is catching up and going above

Red Hat, Inc.

Marcelo Ricardo Leitner, Xin Long

Linux Plumber Conference in Vancouver, 2018

redhat.

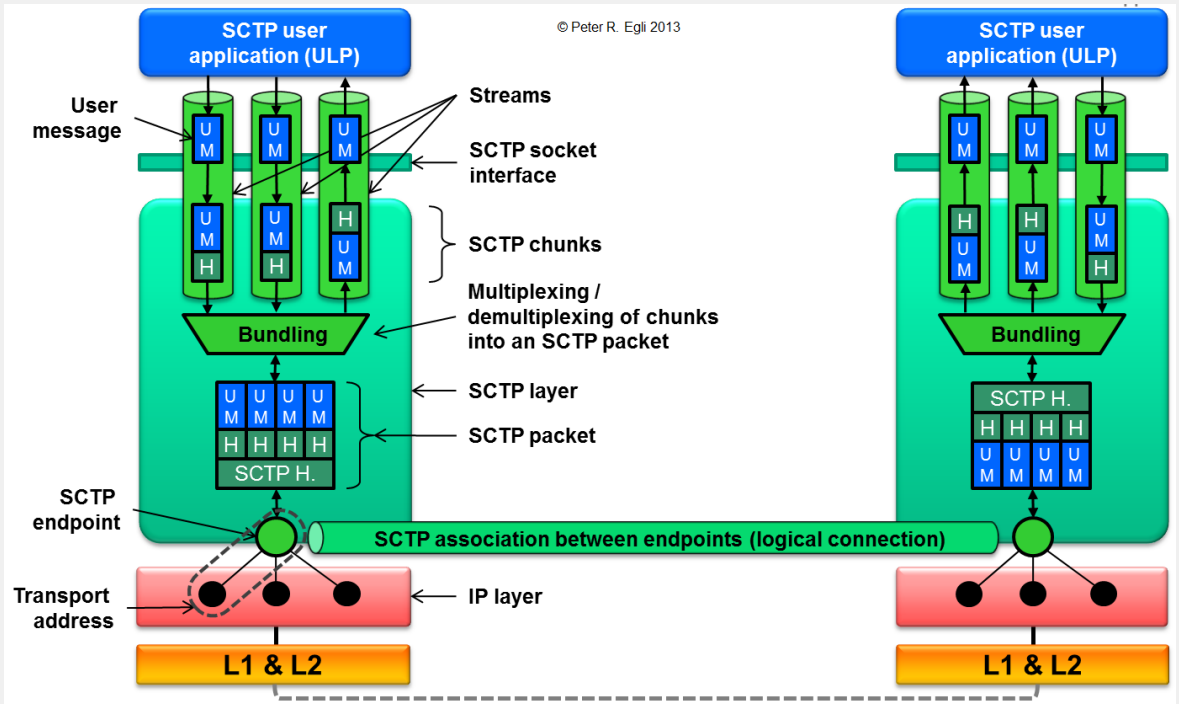
Outline

redhat.

Structures

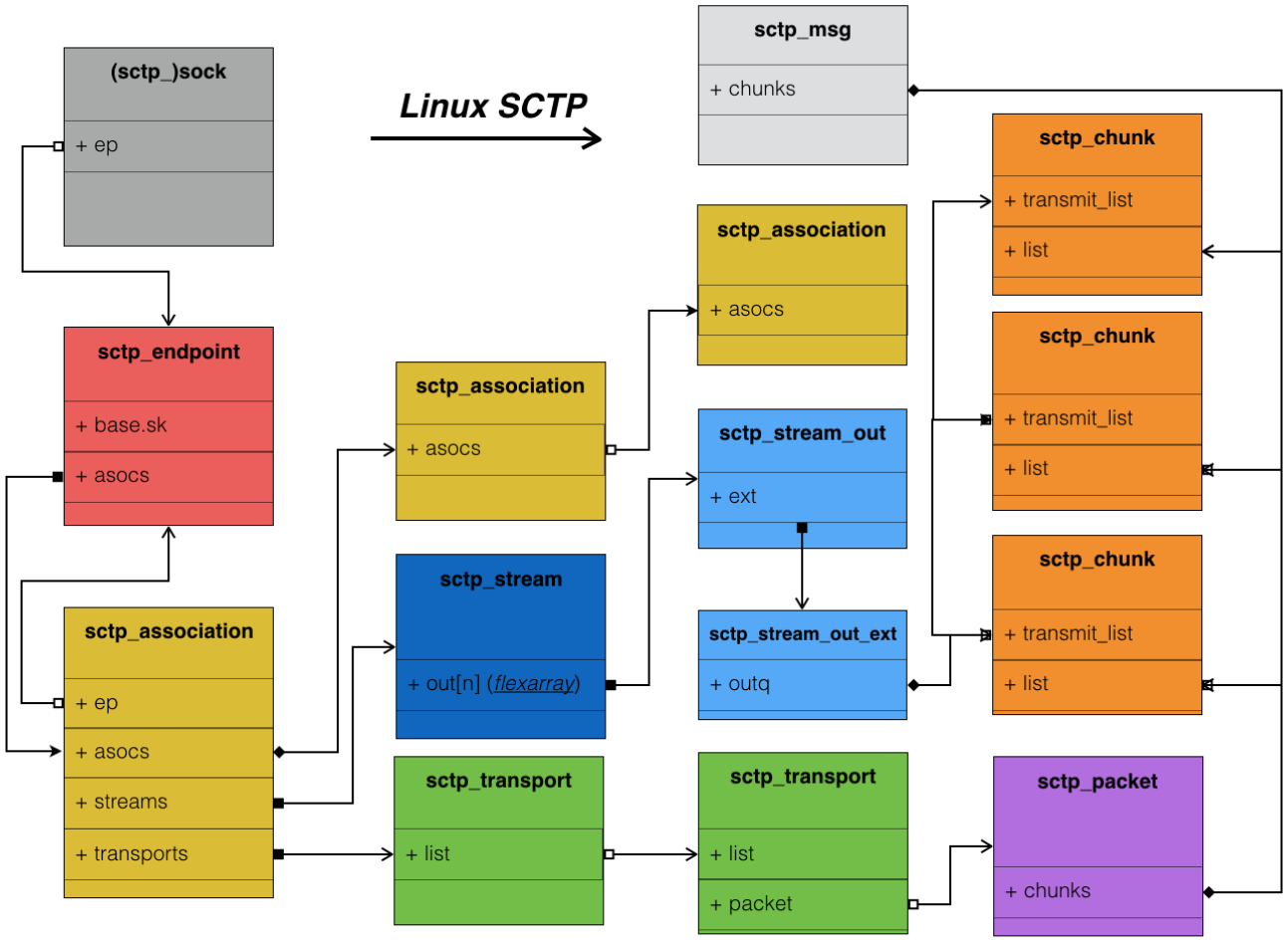
1. Endpoint
2. Association
3. Transport
4. Stream
5. Msg
6. Packet
7. Chunk

overview



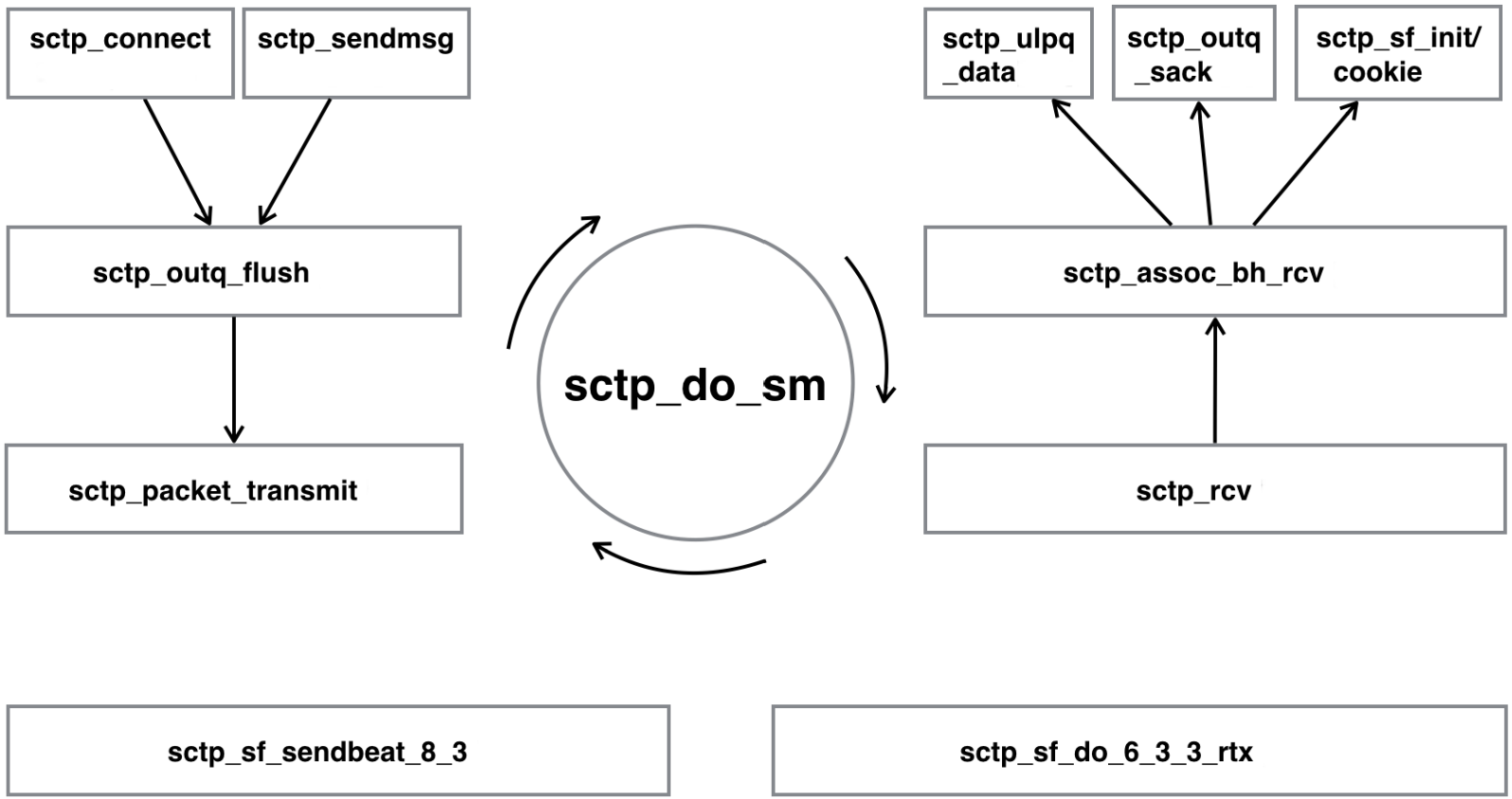
redhat.

SCTP Structures in Linux



redhat

SCTP Procedures in Linux



redhat

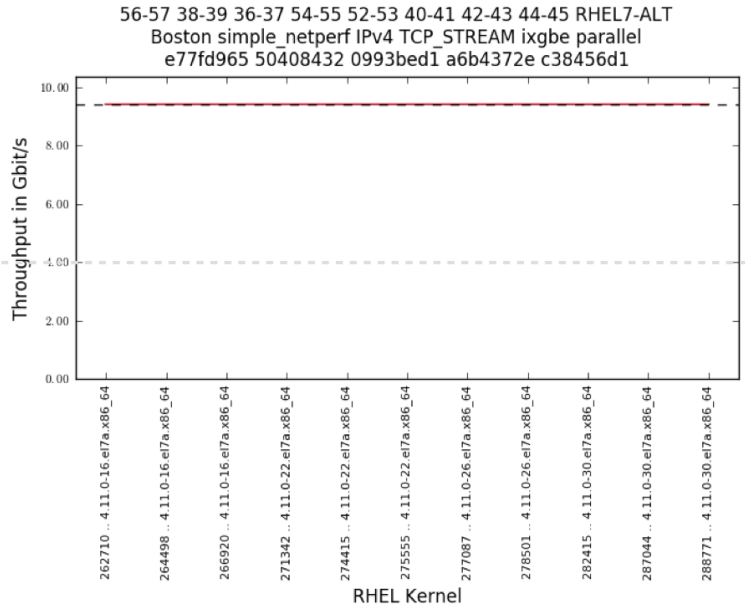
SCTP vs TCP/UDP on Feature

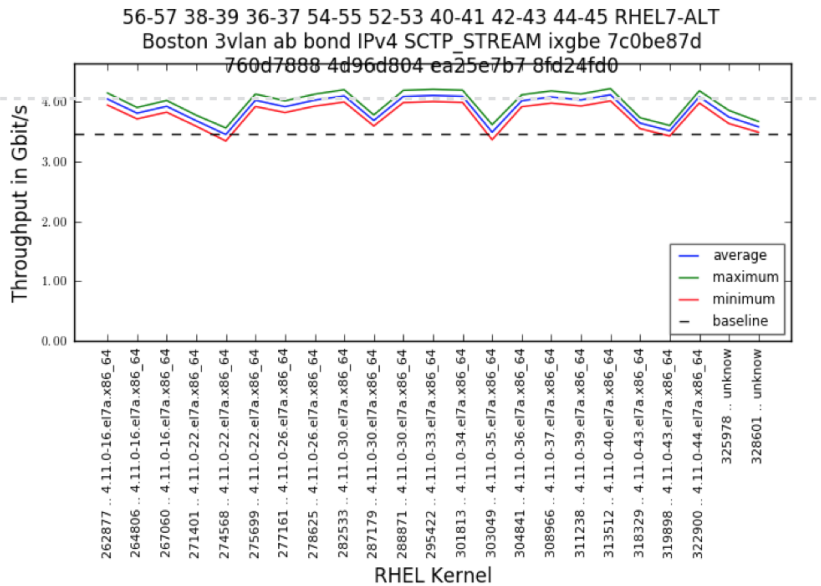
	SCTP	TCP	UDP
• Full-duplex	yes	yes	yes
• Connection oriented	yes	yes	no
• Reliable data transmission	yes	yes	no
• Unreliable data transmission	yes	no	yes
• Partially reliable data transmission	yes	no	no
• In order delivery	yes	yes	no
• Out of order delivery	yes	no	yes
• Flow- and Congestion Control	yes	yes	no
• ECN support	yes	yes	no
• Selective ACKs	yes	yes	no
• Protection of message boundaries	yes	no	yes
• Fragmentations	yes	yes	no
• Multistreaming	yes	no	no
• Multihoming	yes	no	no
• Protection against SYN flooding	yes	no	n/a
• Half-closed connection	no	yes	n/a

redhat.

SCTP vs TCP on Performance

Performance ?





redhat.

Outline

redhat

lksctp-tools (lib and unit test)

MANIFEST ———

.

|– bin

|– doc

|– man

|– src

... |– **apps**

... |– **func_tests**

... |– include

... |... '– netinet

... |– **lib**

... |– testlib

... '– withsctp

- sctp_darn, sctp_test
- sctp_status, sctp_xconnect
- peel_client, peel_server
- bindx_test, myftp, nagle_rcv, nagle_snd

"""

Unit Test: Look in src/func_tests and in lksctp-tests package for examples of of tests. Please do not submit code that fails its own tests or any of the unit tests. If it fails a functional test, please document that with the submission.

"""

- sctp_send, sctp_sendmsg, sctp_recvmsg
- sctp_connectx_orig, sctp_connectx2, sctp_connectx3
- sctp_bindx, sctp_opt_info
- sctp_peeloff, sctp_peeloff_flags

redhat

sctp-tests (regression test): 27 test cases so far

```
.
├── libst
│   ├── bin
│   ├── keys
│   ├── logging.sh
│   ├── network.sh
│   └── system.sh
├── st.cfg
├── sctp-tests
├── testcase
│   ├── misc
│   │   └── demo
│   ├── conformance
│   │   ├── streamreconf
│   │   ├── streamndata
│   │   ├── interleave
│   │   └── prsctp
│   ├── performance
│   │   ├── repeatability
│   │   └── sctphashtable
│   ├── regression
│   │   ├── extoverflow
│   │   ├── gsomtuchange
│   │   │   ├── issue
│   │   │   └── test.sh
│   │   └── peeloffbusy
│   ├── stress
│   │   ├── procdumps
│   │   └── sctpdiag
│   └── testsuit
│       └── lksctptools
├── testplan
│   ├── phostcase.list
│   └── upstream.list
└── topos
    ├── netns.sh
    ├── netns_cs.sh
    ├── netns_crs.sh
    ├── netns_sit_cs.sh
    ├── netns_ipsec_cs.sh
    ├── netns_vlan_cs.sh
    ├── netns_team_cs.sh
    ├── netns_bridge_cs.sh
    ├── ph.cfg
    ├── phost.sh
    ├── phost_cs.sh
```

```
# ./sctp-tests help
sctp-tests v1.0

usage: sctp-tests [ OPTIONS ] COMMAND
where  OPTIONS:= { -e {TestCase | TestPlan}    |
                   -o  Prefix    | -l LogLevel |
                   -O  Outfile   | -l LogFile  |
                   -T  TOPOOPT   | -f Confile }
       COMMAND:= { init | run | shell }

       TOPOOPT:= { st_team_opt=xxx  | st_ipsec_opt=xxx }
```

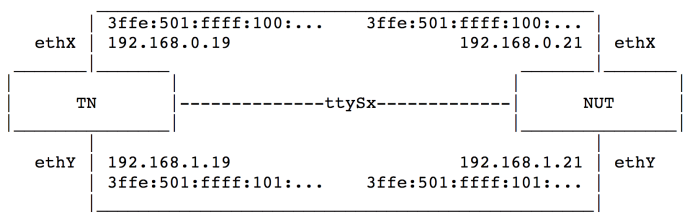
```
# cat upstream.list
testcase/testsuit/lksctptools/test.sh
testcase/performance/repeatability/sctp_rr.sh
testcase/performance/repeatability/sctp_stream.sh
testcase/performance/repeatability/sctp_rr~on_ipsec.sh
testcase/performance/repeatability/sctp_stream~on_ipsec.sh
testcase/performance/repeatability/sctp_stream~on_sit.sh
testcase/stress/procdumps/test.sh
testcase/stress/sctpdiag/test.sh
testcase/regression/gsomtuchange/test.sh
testcase/regression/peeloffbusy/test.sh
testcase/regression/extoverflow/test.sh
testcase/performance/sctphashtable/test~on_netns.sh
testcase/performance/sctphashtable/test~on_team.sh
```

```
# cat netns_ipsec_cs.sh
source "$ROOTDIR/topos/netns.sh"

st_topo_setup()
{
        local def_opt="proto esp spi 0x1000 mode transport enc blowfish i
                       proto esp spi 0x2000 mode transport enc blowfish :
        local ipsec_opt=${Topo_opt:-"$st_ipsec_opt"}

        ipsec_opt=${ipsec:-"$def_opt"}

        st_c_eth=(eth1 eth2)
        st_s_eth=(eth1 eth2)

        st_c_ip6=(2000::1 1000::1)
        st_s_ip6=(2000::2 1000::2)
        st_c_ip4=(192.0.0.1 172.0.0.1)
        st_s_ip4=(192.0.0.2 172.0.0.2)

        st_netns_ns_create c s
        st_netns_veth_create c "${st_c_eth[*]}" s "${st_s_eth[*]}"

        st_netns_addr_assign c "${st_c_eth[*]}" "${st_c_ip4[*]}" "24 16"
        st_netns_addr_assign s "${st_s_eth[*]}" "${st_s_ip4[*]}" "24 16"
        st_netns_addr_assign c "${st_c_eth[*]}" "${st_c_ip6[*]}" "64 128"
        st_netns_addr_assign s "${st_s_eth[*]}" "${st_s_ip6[*]}" "64 128"

        st_netns_ipsec_create c "${st_c_ip4[*]}" "${st_s_ip4[*]}" \
                "$ipsec_opt" "${st_c_ip4[*]}" "${st_s_ip4[*]}" "out"
        st_netns_ipsec_create c "${st_c_ip6[*]}" "${st_s_ip6[*]}" \
                "$ipsec_opt" "${st_c_ip6[*]}" "${st_s_ip6[*]}" "out"
        st_netns_ipsec_create s "${st_c_ip4[*]}" "${st_s_ip4[*]}" \
                "$ipsec_opt" "${st_c_ip4[*]}" "${st_s_ip4[*]}" "in"
        st_netns_ipsec_create s "${st_c_ip6[*]}" "${st_s_ip6[*]}" \
                "$ipsec_opt" "${st_c_ip6[*]}" "${st_s_ip6[*]}" "in"
}

st_topo_clean()
{
        st_netns_ipsec_destroy c
        st_netns_ipsec_destroy s
        st_netns_ns_destroy c s
}
```

```
# cat test.sh
Name="gsomtuchange"
Topo="netns_crs"

do_setup()
{
        st_s_run sctp_test -H :: -P 33331 -l -T > /dev/null 2>&1 &
}

do_clean()
{
        pkill sctp_test
        rm -rf *.log
}

do_test()
{
        local mtus="1490 1480 1470 1460 1450 1440 1430 1420 1410 1400"
        local logf="$(st_o)_send.log"
        local pids=""
        local i=0

        st_log DBG "sctp_test start"
        for mtu in $mtus; do
                st_c_run sctp_test -H ${st_c_ip4[0]} -P 3333$i \
                                   -h ${st_s_ip4[0]} -p 33331 -s -c 5 \
                                   -x 1000 -T 2>&1 >> $logf &
                pids="$pids $!"
                st_c_run sctp_test -H ${st_c_ip6[0]} -P 3333$i \
                                   -h ${st_s_ip6[0]} -p 33331 -s -c 5 \
                                   -x 1000 -T 2>&1 >> $logf &
                pids="$pids $!"
                sleep 2
                st_r_run ip link set dev ${st_r_eth[0]} mtu $mtu
                st_r_run ip link set dev ${st_r_eth[2]} mtu $mtu
                let i++
        done

        st_log DBG "wait for children $pids"
        wait $pids

        st_log INFO " - PASS - No Panic"
}
```

redhat.

tahi-sctp (conformance test)



RFC4960: Association Initialization
RFC4960: Association Termination
RFC4960: Fault Management
RFC4960: Error Cause
RFC4960: Chunk Bundling
RFC4960: User Data Transfer
RFC4960: Retransmission Timer
RFC4960: Congestion Control
RFC4960: Path MTU Discovery
RFC4960: Multi-Homed Endpoints
RFC4960: Explicit Congestion Notification
RFC4960: Packet Format
RFC4960: Miscellaneous Test
RFC4895: Authentication Chunks
RFC5061: Dynamic Address Reconfiguration
RFC3758: Partial Reliability Extension
RFC3554: Internet Protocol Security

SCTP Conformance Test For Dynamic Address Reconfiguration

Tool Version : REL_3_3_2
Test Program Version : REL_1_0_3

Start: 2017/05/12 13:00:44
End : 2017/05/12 13:32:18

No.	Title	Result	Log	Script	Packet	Dump (bin)
	Initialize					
1	Initialization Test Environment	-	X	X	-	Link0 Link1
	ASCONF Chunk Procedures					
	Sequence Number					
2	ASCONF chunk is received with duplicate Peer Sequence Number and cached ASCONF-ACK response exists	PASS	X	X	X	Link0 Link1
3	ASCONF chunk is received with duplicate Peer Sequence Number but no cached ASCONF-ACK is outstanding	PASS	X	X	X	Link0 Link1
4	ASCONF chunk is received with Sequence Number greater than next expected Sequence Number	PASS	X	X	X	Link0 Link1
	Parameter Type					
5	ASCONF chunk is received with unrecognized parameter which does not understand	FAIL, Why	X	X	X	Link0 Link1
	Add IP Address					
6	ASCONF chunk is received with Add IP Address Parameter which contain a broadcast or multicast address	PASS	X	X	X	Link0 Link1
7	ASCONF chunk is received with Add IP Address Parameter which contain a unspecified address	FAIL, Why	X	X	X	Link0 Link1
8	ASCONF chunk is received with Add IP Address Parameter which contain a different type address	PASS	X	X	X	Link0 Link1
9	ASCONF chunk is received with Add IP Address Parameter which contain a valid address	PASS	X	X	X	Link0 Link1
	Delete IP Address					
10	ASCONF chunk is received with Delete IP Address Parameter which contain a broadcast or multicast address	PASS	X	X	X	Link0 Link1
11	ASCONF chunk is received with Delete IP Address Parameter which contain a unspecified address	PASS	X	X	X	Link0 Link1
12	ASCONF chunk is received with Delete IP Address to delete last remaining IP address from an association	PASS	X	X	X	Link0 Link1
13	ASCONF chunk is received with Delete IP Address to delete an IP address that is also the source address	PASS	X	X	X	Link0 Link1
14	ASCONF chunk is received with Delete IP Address Parameter which contain a address not part of the association	PASS	X	X	X	Link0 Link1

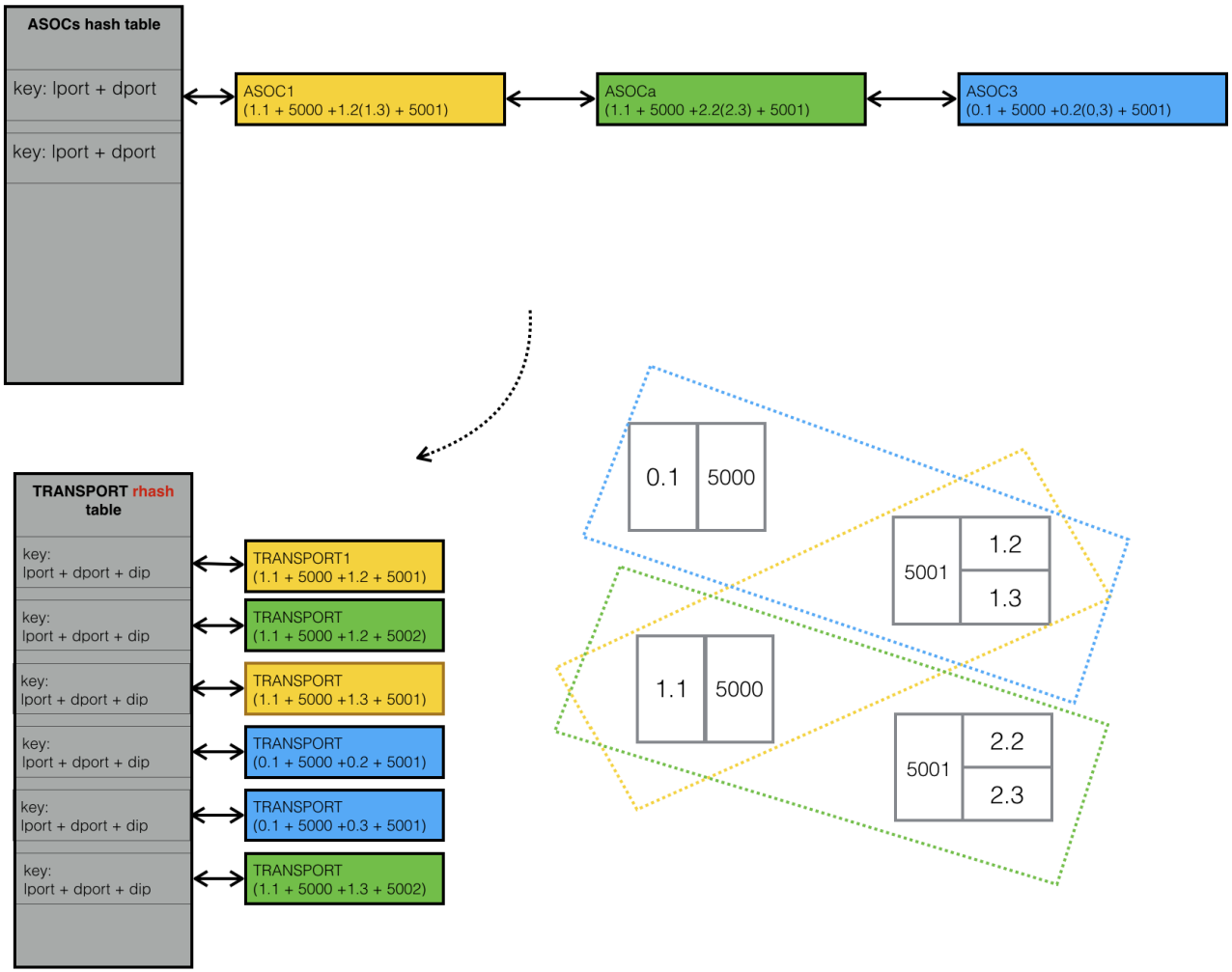
redhat.

Others

- Syzkaller (fuzz test)
- Codenomicon (fuzz test)
- Packetdrill (conformance test)
- Scapy (packet generating)
- More ?

redhat.

Transport Rhashtable 1



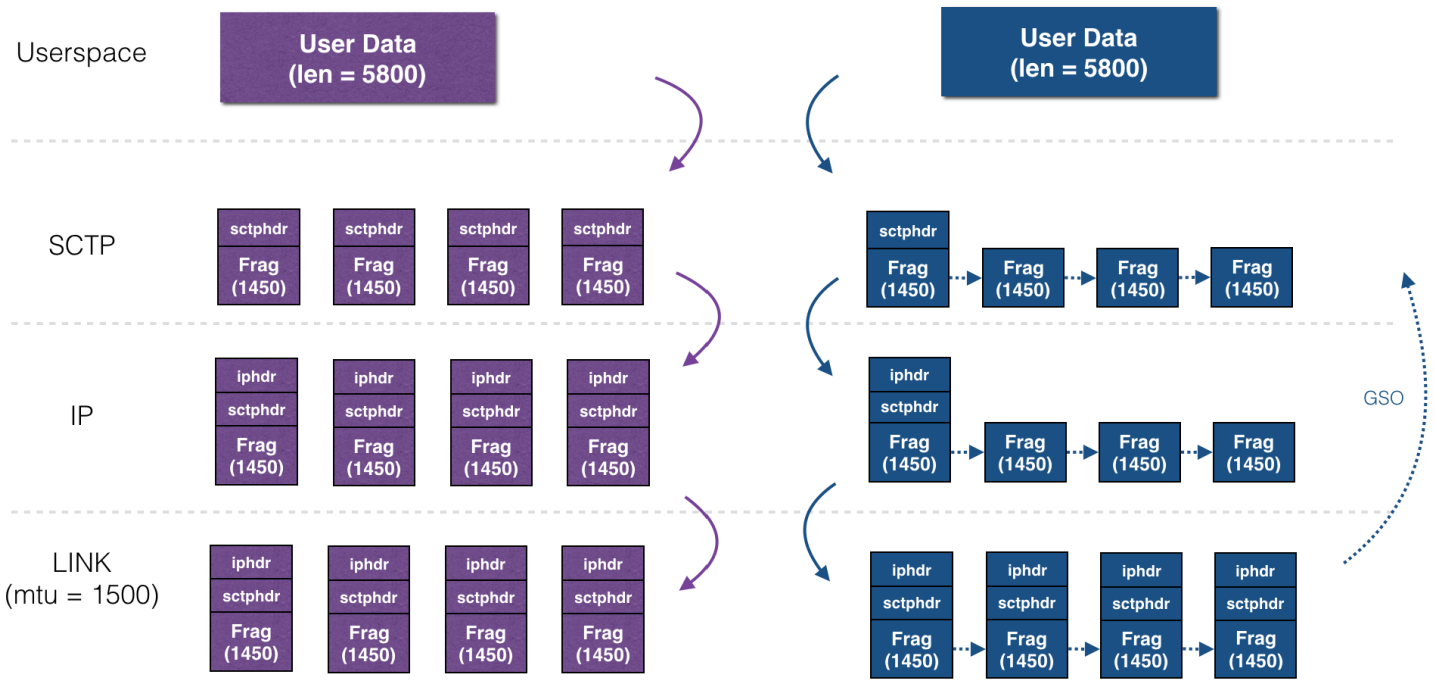
redhat.

Transport Rhashtable 2

1. 1-to-many(with "the same dport and different dip" lookup fast
2. 1-step to find both transport and asoc
3. Rhashtable (rhlist) features: rcu_lock and resize memory
4. Why not use the key with hash(dport, lport, dip, lip) ?
5. Why not make rhashtable per endpoint/socket ?

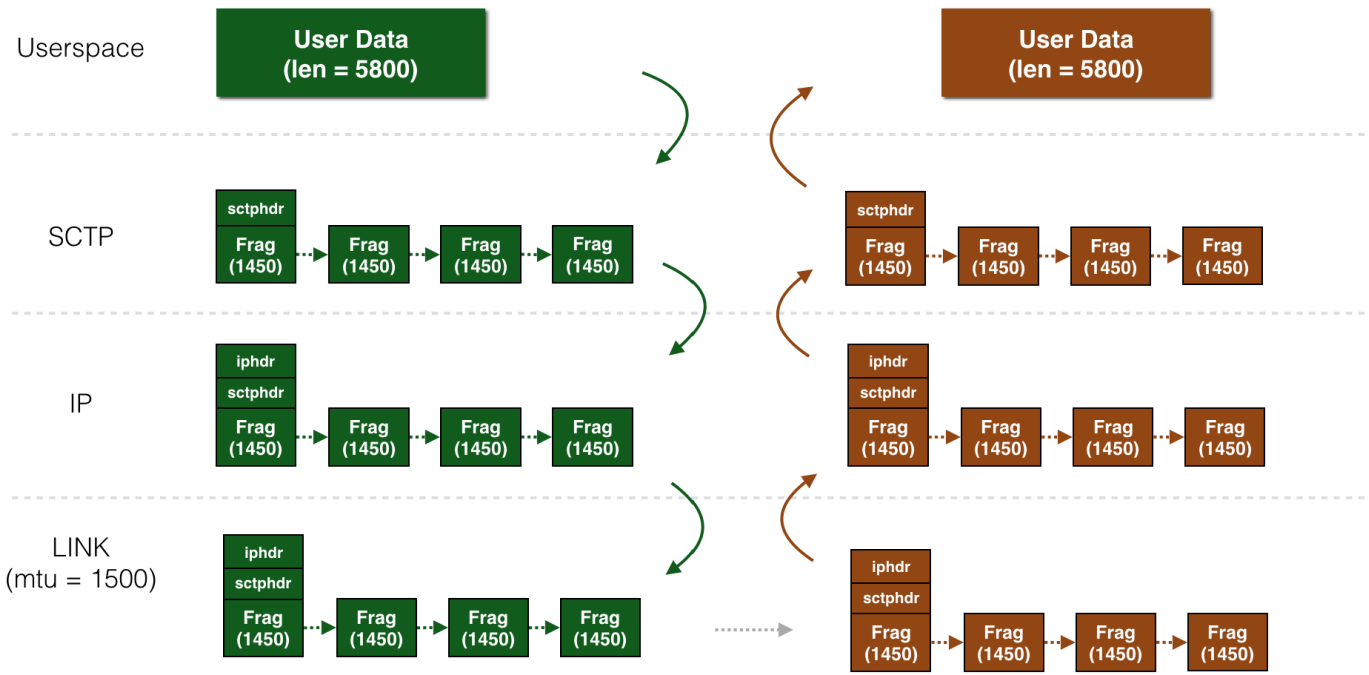
redhat.

SCTP Offload 1



redhat.

SCTP Offload 2



redhat

SCTP Diag 1

```
[iproute2]# ss --sctp  -n -l
State      Recv-Q Send-Q   Local Address:Port       Peer Address:Port
LISTEN     0      128      172.16.254.254:8888      *:*
LISTEN     0      5        127.0.0.1:1234           *:*
LISTEN     0      5        127.0.0.1:1234           *:*
  - ESTAB  0      0        127.0.0.1%lo:1234        127.0.0.1:4321
LISTEN     0      128      172.16.254.254:8888      *:*
  - ESTAB  0      0        172.16.254.254%eth1:8888 172.16.253.253:8888
  - ESTAB  0      0        172.16.254.254%eth1:8888 172.16.1.1:8888
  - ESTAB  0      0        172.16.254.254%eth1:8888 172.16.1.2:8888
  - ESTAB  0      0        172.16.254.254%eth1:8888 172.16.2.1:8888
  - ESTAB  0      0        172.16.254.254%eth1:8888 172.16.2.2:8888
  - ESTAB  0      0        172.16.254.254%eth1:8888 172.16.3.1:8888
  - ESTAB  0      0        172.16.254.254%eth1:8888 172.16.3.2:8888
LISTEN     0      0        127.0.0.1:4321           *:*
  - ESTAB  0      0        127.0.0.1%lo:4321        127.0.0.1:1234
```

```
[iproute2]# ss -Snai
State       Recv-Q Send-Q  Local Address:Port     Peer Address:Port
LISTEN      0      1           127.0.0.1:27375               *:*
locals:127.0.0.1,192.168.42.2, v4mapped:1
ESTAB       0      0           127.0.0.1:37636       127.0.0.1:27375
locals:0.0.0.0, v4mapped:1
```

redhat.

SCTP Diag 2

```
# cat /proc/net/sctp/assocs
 ASSOC     SOCK   STY SST ST HBKT ASSOC-ID TX_QUEUE RX_QUEUE UID INODE LPORT RPORT
LADDRS <-> RADDRS HBINT INS OUTS MAXRT T1X T2X RTXC wmema wmemq sndbuf rcvbuf
```


```
struct sctp_info {
        __u32   sctpi_tag;
        __u32   sctpi_state;
        __u32   sctpi_rwnd;
        __u16   sctpi_unackdata;
        __u16   sctpi_penddata;
        __u16   sctpi_instrms;
        __u16   sctpi_outstrms;
        __u32   sctpi_fragmentation_point;
        __u32   sctpi_inqueue;
        __u32   sctpi_outqueue;
        __u32   sctpi_overall_error;
        __u32   sctpi_max_burst;
        __u32   sctpi_maxseg;
        __u32   sctpi_peer_rwnd;
        __u32   sctpi_peer_tag;
        __u8    sctpi_peer_capable;
        __u8    sctpi_peer_sack;
        __u16   __reserved1;

        /* assoc status info */
        __u64   sctpi_isacks;
        __u64   sctpi_osacks;
        __u64   sctpi_opackets;
        __u64   sctpi_ipackets;
        __u64   sctpi_rtxchunks;
        __u64   sctpi_outofseqtsns;
        __u64   sctpi_idupchunks;
        __u64   sctpi_gapcnt;
        __u64   sctpi_ouodchunks;
        __u64   sctpi_iuodchunks;
        __u64   sctpi_oodchunks;
        __u64   sctpi_iodchunks;
        __u64   sctpi_octrlchunks;
        __u64   sctpi_ictrlchunks;

        /* primary transport info */
        struct sockaddr_storage sctpi_p_address;
        __s32   sctpi_p_state;
        __u32   sctpi_p_cwnd;
        __u32   sctpi_p_srtt;
        __u32   sctpi_p_rto;
        __u32   sctpi_p_hbinterval;
        __u32   sctpi_p_pathmaxrxt;
        __u32   sctpi_p_sackdelay;
        __u32   sctpi_p_sackfreq;
        __u32   sctpi_p_ssthresh;
        __u32   sctpi_p_partial_bytes_acked;
        __u32   sctpi_p_flight_size;
        __u16   sctpi_p_error;
        __u16   __reserved2;

        /* sctp sock info */
        __u32   sctpi_s_autoclose;
        __u32   sctpi_s_adaptation_ind;
        __u32   sctpi_s_pd_point;
        __u8    sctpi_s_nodelay;
        __u8    sctpi_s_disable_fragments;
        __u8    sctpi_s_v4mapped;
        __u8    sctpi_s_frag_interleave;
        __u32   sctpi_s_type;
        __u32   __reserved3;
};
```

redhat

Others

- Dst source addr selection
- Rwnd improvements
- Partial reliability fixes
- MTU handling refactor
- PMTU discovery (critical) fixes
- CRC32c offloading on virtual interfaces
- Some codes cleaning up
- More ...

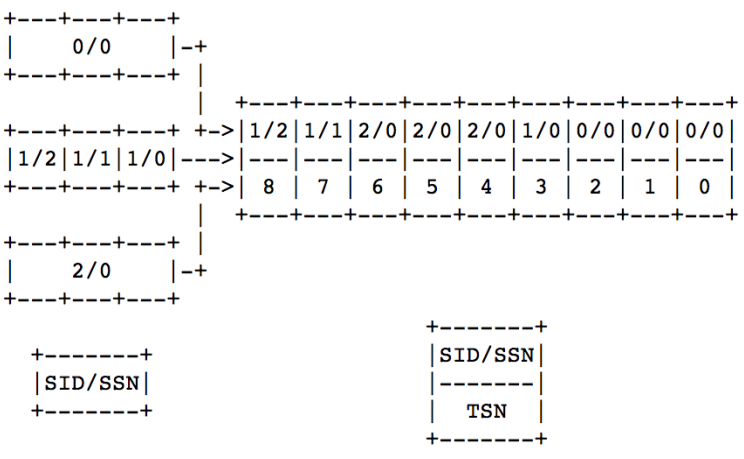
redhat

Overview

- Stream Schedulers and User Message Interleaving for the Stream Control Transmission Protocol [RFC8260]
- Additional Policies for the Partially Reliable Stream Control Transmission Protocol Extension [RFC7496]
- Stream Control Transmission Protocol (SCTP) Stream Reconfiguration [RFC6525]
- Sockets API Extensions for the Stream Control Transmission Protocol (SCTP) [RFC6458]
- Full SELinux support
- More ...

redhat.

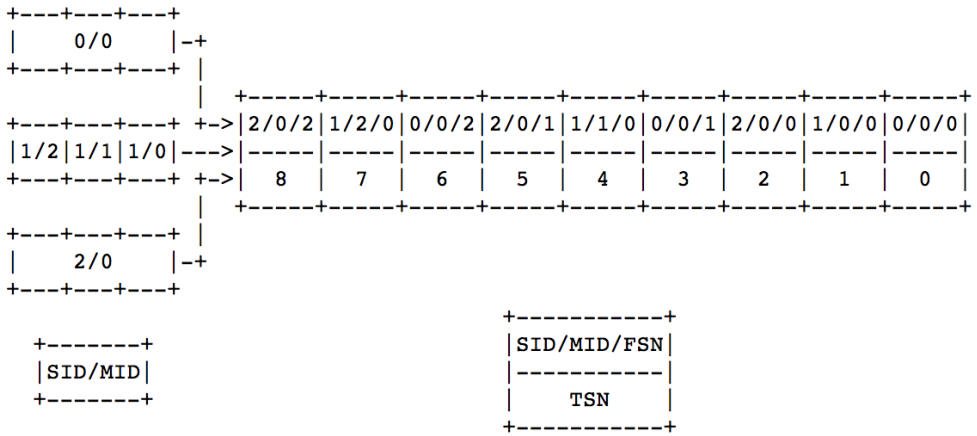
Stream Schedulers




```
struct sctp_sched_ops {
        /* Property handling for a given stream */
        int (*set)(struct sctp_stream *stream, __u16 sid, __u16 value,
                   gfp_t gfp);
        int (*get)(struct sctp_stream *stream, __u16 sid, __u16 *value);

        /* Init the specific scheduler */
        int (*init)(struct sctp_stream *stream);
        /* Init a stream */
        int (*init_sid)(struct sctp_stream *stream, __u16 sid, gfp_t gfp);
        /* Frees the entire thing */
        void (*free)(struct sctp_stream *stream);

        /* Enqueue a chunk */
        void (*enqueue)(struct sctp_outq *q, struct sctp_datamsg *msg);
        /* Dequeue a chunk */
        struct sctp_chunk *(*dequeue)(struct sctp_outq *q);
        /* Called only if the chunk fit the packet */
        void (*dequeue_done)(struct sctp_outq *q, struct sctp_chunk *chunk);
        /* Sched all chunks already enqueued */
        void (*sched_all)(struct sctp_stream *steam);
        /* Unched all chunks already enqueued */
        void (*unsched_all)(struct sctp_stream *steam);
};
```

redhat.

Message Interleaving

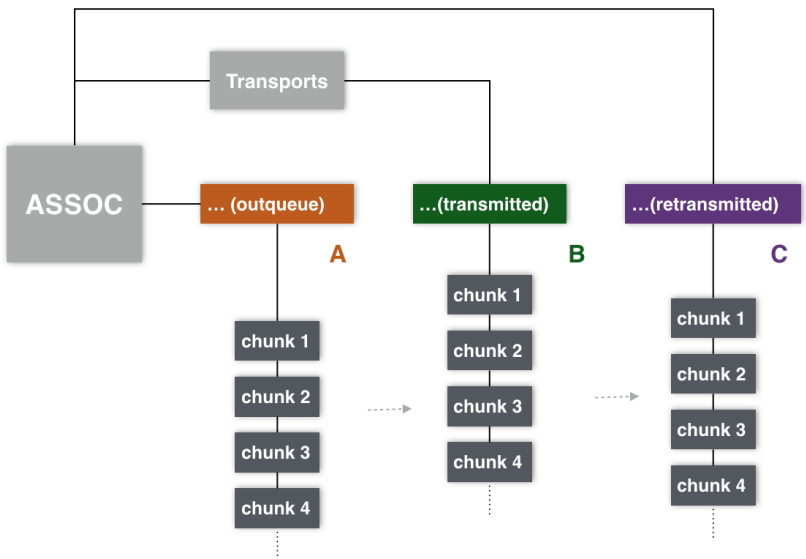



```
struct sctp_stream_interleave {
        __u16   data_chunk_len;
        __u16   ftsn_chunk_len;
        /* (I-)DATA process */
        struct sctp_chunk *(*make_datafrag)(const struct sctp_association *asoc,
                                            const struct sctp_sndrcvinfo *sinfo,
                                            int len, __u8 flags, gfp_t gfp);
        void    (*assign_number)(struct sctp_chunk *chunk);
        bool    (*validate_data)(struct sctp_chunk *chunk);
        int     (*ulpevent_data)(struct sctp_ulpq *ulpq,
                                 struct sctp_chunk *chunk, gfp_t gfp);
        int     (*enqueue_event)(struct sctp_ulpq *ulpq,
                                 struct sctp_ulpevent *event);
        void    (*renege_events)(struct sctp_ulpq *ulpq,
                                 struct sctp_chunk *chunk, gfp_t gfp);
        void    (*start_pd)(struct sctp_ulpq *ulpq, gfp_t gfp);
        void    (*abort_pd)(struct sctp_ulpq *ulpq, gfp_t gfp);
        /* (I-)FORWARD-TSN process */
        void    (*generate_ftsn)(struct sctp_outq *q, __u32 ctsn);
        bool    (*validate_ftsn)(struct sctp_chunk *chunk);
        void    (*report_ftsn)(struct sctp_ulpq *ulpq, __u32 ftsn);
        void    (*handle_ftsn)(struct sctp_ulpq *ulpq,
                               struct sctp_chunk *chunk);
};
```

redhat

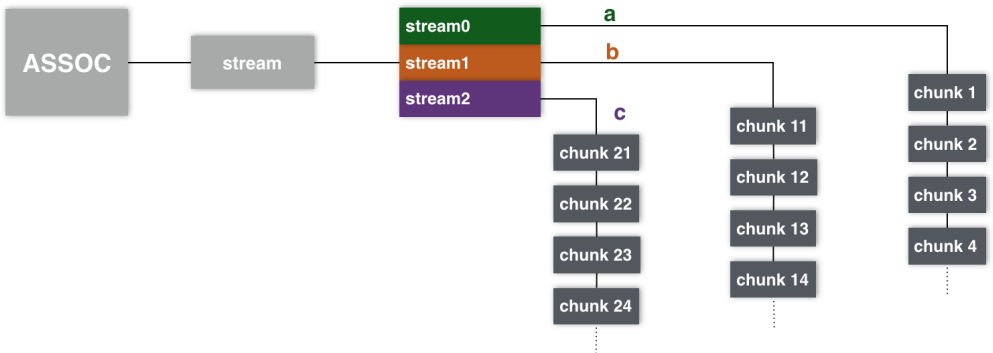
PR_SCTP policies

1. Timed Reliability
   SCTP_PR_SCTP_TTL
2. Limited Retransmissions Policy
   SCTP_PR_SCTP_RTX
   - When dequeuing chunks from A
   - When dequeuing chunks from C
   - When moving chunks from B to C
   - After receiving a SACK, check B and C
3. Priority Policy
   SCTP_PR_SCTP_PRIO
   - Before enqueuing chunk into A
   - And No Enough TX Buffer
   - Then try to drop C -> B -> A.



redhat.

Stream Reconfig



1. Add Outgoing Streams:
   No restrictions
2. Add Incoming Streams:
   No restrictions
3. Reset Outgoing Streams:
   Reset stream 1, b have to be empty
4. Reset Incoming Streams:
   Peer will send Outgoing Stream request for which it has to follow the above rule
5. Reset SSN/TSN:
   All queues have to be empty: A, B, C, a, b, c

redhat.

Socket APIs

1. User APIs
   - sctp_sendv
   - sctp_recvv
2. Snd Info Flags
   - SENDALL
   - MSG_MORE
3. Cmsgs
   - PR_INFO
   - AUTH_INFO
   - DSTv4
   - DSTv6

redhat.

Linux vs BSD on Features

Chunks

```
LINUX:
 ongoing

BSD:
 SCTP_NR_SELECTIVE_ACK (draft)
 SCTP_PACKET_DROPPED (draft)
 SCTP_PAD_CHUNK
```

Others

```
LINUX:
 sctp_do_sm()
 transport rhashtable
 offload
 diag

BSD:
 sctp_cc_functions
```

redhat.

Outline

redhat

Features Development

- Support more Chunks, Apis, Sockopts, Notifications.
- Other features from Draft RFC, like SCTP NAT and CMT.
- SCTP Performance Improvement (including sndbuf auto-tuning)
- Add more test cases in sctp-tests.

redhat.

Code Refactor

- Some huge and messy functions.
- Congestion framework.
- Refactor lksctp-tools.

redhat.

Hardware Support

- GSO x frag_list x frags.
- Checksum.
- Offload.

The end.

Thanks for listening.