NAME

dtrace_sctp - a DTrace provider for tracing events related to the sctp(4) protocol

SYNOPSIS

sctp:cwnd::init(*uint32_t*, *uint32_t*, *uintptr_t*, *int*, *int*);

sctp:cwnd::ack(*uint32_t*, *uint32_t*, *uintptr_t*, *int*, *int*);

sctp:cwnd::rttvar(*uint64_t*, *uint64_t*, *uint64_t*, *uint64_t*, *uint64_t*);

sctp:cwnd::rttstep(*uint64_t*, *uint64_t*, *uint64_t*, *uint64_t*, *uint64_t*);

sctp:cwnd::fr(*uint32_t*, *uint32_t*, *uintptr_t*, *int*, *int*);

sctp:cwnd::to(*uint32_t*, *uint32_t*, *uintptr_t*, *int*, *int*);

sctp:cwnd::bl(*uint32_t*, *uint32_t*, *uintptr_t*, *int*, *int*);

sctp:cwnd::ecn(*uint32_t*, *uint32_t*, *uintptr_t*, *int*, *int*);

sctp:cwnd::pd(*uint32_t*, *uint32_t*, *uintptr_t*, *int*, *int*);

sctp:rwnd:assoc:val(*uint32_t*, *uint32_t*, *int*, *int*);

sctp:flightsize:net:val(*uint32_t*, *uint32_t*, *uintptr_t*, *int*, *int*);

sctp:flightsize:assoc:val(*uint32_t*, *uint32_t*, *int*, *int*);

sctp:::receive(*pktinfo_t *, csinfo_t *, ipinfo_t *, sctpsinfo_t *, sctpinfo_t **);

sctp:::send(*pktinfo_t *, csinfo_t *, ipinfo_t *, sctpsinfo_t *, sctpinfo_t **);

sctp:::state-change(*void *, csinfo_t *, void *, sctpsinfo_t *, void *, sctplsinfo_t **);

DESCRIPTION

The DTrace **sctp** provider allows users to trace events in the sctp(4) protocol implementation. This provider is similar to the dtrace_ip(4) and dtrace_udp(4) providers, but additionally contains probes corresponding to protocol events at a level higher than packet reception and transmission.

The **sctp:cwnd::**() probes track changes in the congestion window on a netp. The **sctp:rwnd::**() probes track changes in the receiver window for an assoc. The **sctp:flightsize:net:val**() probe tracks changes in the flight size on a net or assoc and the **sctp:flightsize:assoc:val**() probe provides the total flight version.

The arguments of all **sctp** probes except for **sctp:cwnd::rtt***() and **sctp::assoc:val**() are the Vtag for this end, the port number of the local side, the pointer to struct sctp_nets *changing, the old value of the cwnd, and the new value of the cwnd.

The arguments of **sctp:::val**() are similar to the above except the fourth argument is the up/down amount.

The **sctp:cwnd::rtt***() probe arguments are a bitmap of Vtag << 32 | localport << 16 | remoteport, a bitmap of obw | nbw, a bitmap of bwrtt | newrtt, flight, and a bitmap of (cwnd << 32) | point << 16 | retval(0/1).

The **sctp:cwnd::init**() probe fires when a remotely-initiated active SCTP open succeeds. At this point the new connection is in the ESTABLISHED state, and the probe arguments expose the headers associated with the final ACK of the four-way handshake.

The **sctp:::send**() and **sctp:::receive**() probes fire when the host sends or receives an SCTP packet, respectively. As with the dtrace_udp(4) provider, **sctp** probes fire only for packets sent by or to the local host; forwarded packets are handled in the IP layer and are only visible to the dtrace_ip(4) provider.

The **sctp:::state-change**() probe fires upon local SCTP association state transitions. Its first, third and fifth arguments are currently always NULL. Its last argument describes the from-state in the transition, and the to-state can be obtained from args[3]->sctps_state.

FILES

/usr/lib/dtrace/sctp.d DTrace type and translator definitions for the **sctp** provider.

EXAMPLES

A script that logs SCTP packets in real time:

```
#pragma D option quiet
#pragma D option switchrate=10hz

dtrace:::BEGIN
{
        printf(" %3s %15s:%-5s      %15s:%-5s\n", "CPU",
            "LADDR", "LPORT", "RADDR", "RPORT");
}
```

```
      sctp:::send
      {
              printf(" %3d %16s:%-5d -> %16s:%-5d\n", cpu,
                  args[2]->ip_saddr, args[4]->sctp_sport,
                  args[2]->ip_daddr, args[4]->sctp_dport);
      }

      sctp:::receive
      {
              printf(" %3d %16s:%-5d <- %16s:%-5d\n", cpu,
                  args[2]->ip_daddr, args[4]->sctp_dport,
                  args[2]->ip_saddr, args[4]->sctp_sport);
      }
```

A script that logs SCTP association state changes as they occur:

```
      #pragma D option quiet
      #pragma D option switchrate=10

      int last[int];

      dtrace:::BEGIN
      {
              printf(" %3s %12s  %-25s    %-25s\n",
                  "CPU", "DELTA(us)", "OLD", "NEW");
      }

      sctp:::state-change
      / last[args[1]->cs_cid] /
      {
              this->elapsed = (timestamp - last[args[1]->cs_cid]) / 1000;
              printf(" %3d %12d  %-25s -> %-25s\n", cpu, this->elapsed,
                  sctp_state_string[args[5]->sctps_state],
                  sctp_state_string[args[3]->sctps_state]);
              last[args[1]->cs_cid] = timestamp;
      }

      sctp:::state-change
      / last[args[1]->cs_cid] == 0 /
      {
              printf(" %3d %12s  %-25s -> %-25s\n", cpu, "-",
                  sctp_state_string[args[5]->sctps_state],
                  sctp_state_string[args[3]->sctps_state]);
              last[args[1]->cs_cid] = timestamp;
      }
```

COMPATIBILITY

The **sctp:::send**(), **sctp:::receive**(), and **sctp:::state-change**() probes are compatible with the **sctp** provider in Solaris. All other probes are only available in FreeBSD.

SEE ALSO

dtrace(1), dtrace_ip(4), dtrace_udp(4), dtrace_udplite(4), sctp(4), SDT(9)

AUTHORS

This manual page was written by Devin Teske <*dteske@FreeBSD.org*>.