UDP Encapsulation of SCTP Packets for End-Host to End-Host Communication
                draft-ietf-tsvwg-sctp-udp-encaps-13.txt

Abstract

   This document describes a simple method of encapsulating SCTP Packets
   into UDP packets and its limitations. This allows the usage of SCTP
   in networks with legacy NAT not supporting SCTP. It can also be used
   to implement SCTP on hosts without directly accessing the IP-layer,
   for example implementing it as part of the application without
   requiring special privileges.

   Please note that this document only describes the functionality
   required within an SCTP stack to add on UDP encapsulation, providing
   only those mechanisms for two end-hosts to communicate with each
   other over UDP ports. In particular, it does not provide mechanisms
   to determine whether UDP encapsulation is being used by the peer, nor
   the mechanisms for determining which remote UDP port number can be
   used. These functions are are out of scope for this document.

   This document covers only end-hosts and not tunneling (egress or
   ingress) end-points.

Copyright Notice

Table of Contents

1. Introduction

This document describes a simple method of encapsulating SCTP packets into UDP packets. SCTP as defined in [RFC4960] runs directly over IPv4 or IPv6. There are two main reasons for encapsulating SCTP packets:

o Allow SCTP traffic to pass through legacy NATs, which do not provide native SCTP support as specified in [I-D.ietf-behave-sctpnat] and [I-D.ietf-tsvwg-natsupp].

o Allow SCTP to be implemented on hosts which do not provide direct access to the IP-layer. In particular, applications can use their own SCTP implementation if the operating system does not provide one.

SCTP provides the necessary congestion control and reliability service that UDP does not perform.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Use Cases

This section discusses two important use cases for encapsulating SCTP into UDP.

3.1. Portable SCTP Implementations

Some operating systems support SCTP natively. For other operating systems implementations are available, but require special privileges to install and/or use them. In some cases no kernel implementation might be available at all. When providing an SCTP implementation as part of a user process, most operating systems require special privileges to access the IP layer directly.

Using UDP encapsulation makes it possible to provide an SCTP implementation as part of a user process which does not require any special privileges.

A crucial point for implementing SCTP in user space is that the
source address of outgoing packets needs to be controlled. This is
not an issue if the SCTP stack can use all addresses configured at
the IP-layer as source addresses. However, it is an issue when also
using the address management required for NAT traversal, described in
Section 5.7.

3.2. Legacy NAT Traversal

Using UDP encapsulation allows SCTP communication when traversing
legacy NATs (i.e those NATs not supporting SCTP as described in
[I-D.ietf-behave-sctpnat] and [I-D.ietf-tsvwg-natsupp]). For single-
homed associations IP addresses MUST NOT be listed in the INIT and
INIT-ACK chunks. To use multiple addresses, the dynamic address
reconfiguration extension described in [RFC5061] MUST be used only
with wildcard addresses in the ASCONF chunks in combination with
[RFC4895].

For multi-homed SCTP association the address management as described
in Section 5.7 MUST be performed.

SCTP sends periodic HEARTBEAT chunks on all idle paths. These can
keep the NAT state alive.

4. Unilateral Self-Address Fixing (UNSAF) Considerations

As [RFC3424] requires a limited scope, this document only covers SCTP
end-points dealing with legacy constraints as described in Section 3.
It doesn't cover generic tunneling end-points.

Obviously, the exit strategy is to use hosts supporting SCTP natively
and middleboxes supporting SCTP as specified in
[I-D.ietf-behave-sctpnat] and [I-D.ietf-tsvwg-natsupp]).

5. SCTP over UDP

5.1. Architectural Considerations

Each SCTP stack uses a single local UDP encapsulation port number as
the destination port for all its incoming SCTP packets.

If there is only a single SCTP implementation on a host (for example,
a kernel implementation being part of the operating system), using a
single local UDP encapsulation port number per host can be
advantageous (e.g., this reduces the number of mappings in firewalls
and NATs, among other things). Using a single local UDP
encapsulation port number per host is not possible if the SCTP stack
is implemented as part of each application, there are multiple

applications, and some of the applications want to use the same IP-
address.

An SCTP implementation supporting UDP encapsulation MUST store a
remote UDP encapsulation port number per destination address for each
SCTP association.

UDP encapsulated SCTP is communicated over the IANA-assigned UDP port
number 9899 (sctp-tunneling). However, implementations SHOULD allow
other port numbers to be specified as a local or remote UDP
encapsulation port number through APIs, as applications may have the
need to communicate over different port numbers.

5.2. Packet Format

To encapsulate an SCTP packet, a UDP header as defined in [RFC0768]
is inserted between the IP header as defined in [RFC0791] and the
SCTP common header as defined in [RFC4960].

Figure 1 shows the packet format of an encapsulated SCTP packet when
IPv4 is used.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         IPv4 Header                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         UDP Header                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     SCTP Common Header                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        SCTP Chunk #1                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        SCTP Chunk #n                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
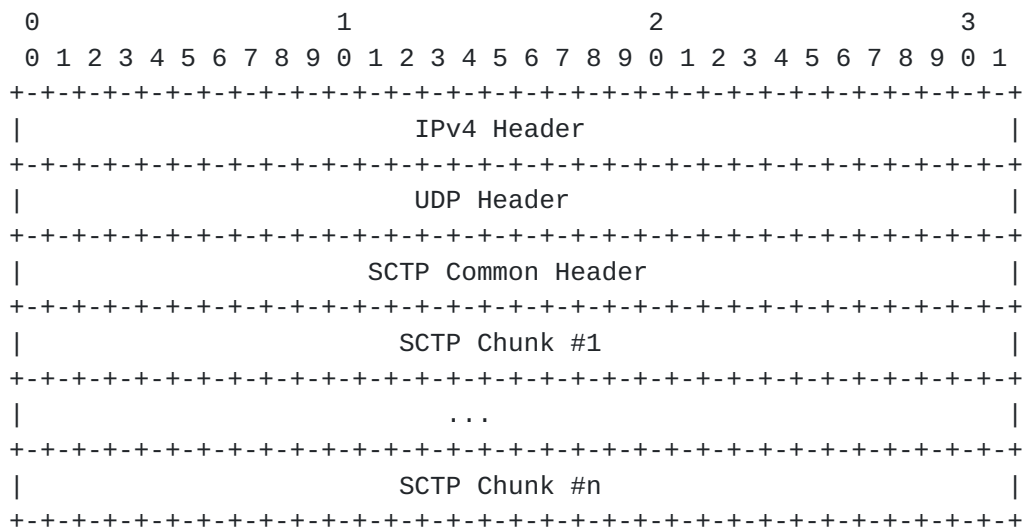
Figure 1: An SCTP/UDP/IPv4 packet

The packet format for an encapsulated SCTP packet when using IPv6 as
defined in [RFC2460] is shown in Figure 2. Please note the the
number m of IPv6 extension headers can be 0.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       IPv6 Base Header                        |
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    IPv6 Extension Header #1                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            ...                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    IPv6 Extension Header #m                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         UDP Header                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      SCTP Common Header                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         SCTP Chunk #1                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            ...                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         SCTP Chunk #n                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
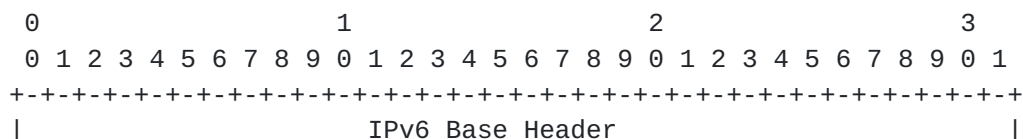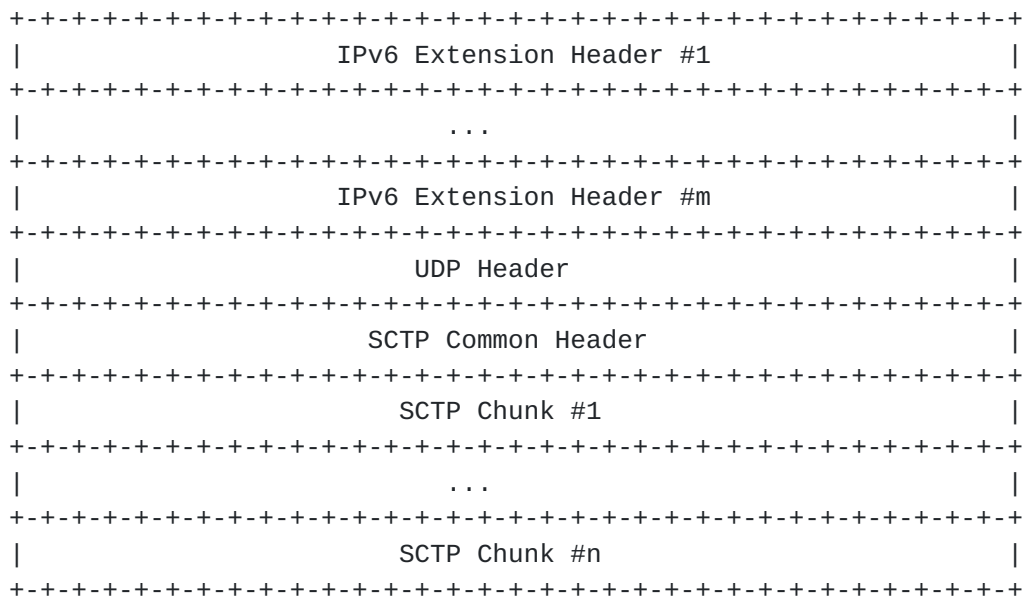
Figure 2: An SCTP/UDP/IPv6 packet

5.3. Encapsulation Procedure

Within the UDP header, the source port MUST be the local UDP
encapsulation port number of the SCTP stack, the destination port
MUST be the remote UDP encapsulation port number stored for the
association and the destination address to which the packet is sent
(see Section 5.1).

Because the SCTP packet is the UDP payload, the length of the UDP
packet MUST be the length of the SCTP packet plus the size of the UDP
header.

The SCTP checksum MUST be computed and the UDP checksum SHOULD be
computed for IPv4 (see [RFC0768]) and IPv6 (see [RFC2460] and
[I-D.ietf-6man-udpzero]). Although UDP with a zero checksum over
IPv6 is allowed under certain constraints [I-D.ietf-6man-udpzero],
this document does not specify mechanisms for this mode. Deployed
support may be limited and at the time of writing, the use of a zero
UDP checksum would also be counter to the goal of legacy NAT
traversal.

5.4. Decapsulation Procedure

When an encapsulated packet is received, the UDP header is removed.
Then the generic lookup is performed, as done by an SCTP stack
whenever a packet is received, to find the association for the
received SCTP packet. After finding the SCTP association (which
includes checking the verification tag), the UDP source port MUST be

stored as the encapsulation port for the destination address the SCTP
packet is received from (see Section 5.1).

When a non-encapsulated SCTP packet is received by the SCTP stack,
the encapsulation of outgoing packets belonging to the same
association and the corresponding destination address MUST be
disabled.

5.5. ICMP Considerations

When receiving ICMP or ICMPv6 response packets, there might not be
enough bytes in the payload to identify the SCTP association which
the SCTP packet triggering the ICMP or ICMPv6 packet belongs to. If
a received ICMP or ICMPv6 packet can not be related to a specific
SCTP association or the verification tag can't be verified, it MUST
be discarded silently. This means in particular that the SCTP stack
MUST NOT rely on receiving ICMP or ICMPv6 messages. Implementation
constraints could prevent processing received ICMP or ICMPv6
messages.

If received ICMP or ICMPv6 messages are processed, the following
mapping SHOULD apply:

1. ICMP messages with type 'Destination Unreachable' and code 'Port
    Unreachable' SHOULD be treated as ICMP messages with type
    'Protocol Unreachable' and code 'Destination Port unreachable.
    See [RFC0792] for more details.

2. ICMPv6 messages with type 'Destination Unreachable' and code
    'Port unreachable' SHOULD be treated as ICMPv6 messages with type
    'Parameter Problem' and code 'Unrecognized Next Header type
    encountered'. See [RFC4443] for more details.

5.6. Path MTU Considerations

If an SCTP endpoint starts to encapsulate the packets of a path, it
MUST decrease the Path MTU of that path by the size of the UDP
header. If it stops encapsulating them, the Path MTU SHOULD be
increased by the size of the UDP header.

When performing Path MTU discovery as described in [RFC4820] and
[RFC4821] it MUST be taken into account that one cannot rely on the
feedback provided by ICMP or ICMPv6 due to the limitation laid out in
Section 5.5.

If the implementation does not allow control of the dont't fragment
(DF)-bit contained in the IPv4 header, then Path MTU discovery can't
be used. In this case, an implementation specific value should be
used instead.

5.7. Handling of Embedded IP-addresses

When using UDP encapsulation for legacy NAT traversal, IP addresses
that might require translation MUST NOT be put into any SCTP packet.

This means that a multi homed SCTP association is setup initially as
a singled homed one and the protocol extension [RFC5061] in
combination with [RFC4895] is used to add the other addresses. Only
wildcard addresses are put into the SCTP packet.

When addresses are changed during the lifetime of an association
[RFC5061] MUST be used with wildcard addresses only. If an SCTP end-
point receives an ABORT with the T-bit set, it MAY use this as an
indication that the addresses seen by the peer might have changed.

5.8. ECN Considerations

If the implementation supports the sending and receiving of the ECN
bits for the IP protocols being used by an SCTP association, the ECN
bits MUST NOT be changed during sending and receiving.

6. Socket API Considerations

This section describes how the socket API defined in [RFC6458] needs
to be extended to provide a way for the application to control the
UDP encapsulation.

Please note that this section is informational only.

A socket API implementation based on [RFC6458] is extended by
supporting one new read/write socket option.

6.1. Get or Set the Remote UDP Encapsulation Port Number
      (SCTP_REMOTE_UDP_ENCAPS_PORT)

This socket option can be used to set and retrieve the UDP
encapsulation port number. This allows an endpoint to encapsulate
initial packets.

```
struct sctp_udpencaps {
  sctp_assoc_t sue_assoc_id;
  struct sockaddr_storage sue_address;
  uint16_t sue_port;
```

```
};
```

sue_assoc_id: This parameter is ignored for one-to-one style
   sockets. For one-to-many style sockets the application may fill
   in an association identifier or SCTP_FUTURE_ASSOC for this query.
   It is an error to use SCTP_{CURRENT|ALL}_ASSOC in sue_assoc_id.

sue_address: This specifies which address is of interest. If a
   wildcard address is provided it applies only to future paths.

sue_port: The UDP port number in network byte order used as the
   destination port number for UDP encapsulation. Providing a value
   of 0 disables UDP encapsulation.

7. IANA Considerations

This document refers to the already assigned UDP port 9899 (sctp-
tunneling). IANA is requested to update this assignment to refer to
this document. As per [RFC6335] the Assignee should be [IESG] and
the Contact should be [IETF_Chair].

Please note that the TCP port 9899 (sctp-tunneling) assignment is not
needed anymore and IANA is asked to remove this TCP port number
assignment.

8. Security Considerations

Encapsulating SCTP into UDP does not add any additional security
considerations to the ones given in [RFC4960] and [RFC5061].

Firewalls inspecting SCTP packets must also be aware of the
encapsulation and apply corresponding rules to the encapsulated
packets.

An attacker might send a malicious UDP packet towards an SCTP end-
point to change the encapsulation port for a single remote address of
a particular SCTP association. However, as specified in Section 5.4,
this requires the usage of one the two negotiated verification tags.
This protects against blind attackers the same way as described in
[RFC4960] for SCTP over IPv4 or IPv6. Non-blind attackers can affect
SCTP association using the UDP encapsulation described in this
document in the same way as SCTP associations not using the UDP
encapsulation of SCTP described here.

9. Acknowledgments

10. References

10.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768,
           August 1980.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September
           1981.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5,
           RFC 792, September 1981.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
           Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2460] Deering, S.E. and R.M. Hinden, "Internet Protocol, Version
           6 (IPv6) Specification", RFC 2460, December 1998.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control
           Message Protocol (ICMPv6) for the Internet Protocol
           Version 6 (IPv6) Specification", RFC 4443, March 2006.

[RFC4820] Tuexen, M., Stewart, R., and P. Lei, "Padding Chunk and
           Parameter for the Stream Control Transmission Protocol
           (SCTP)", RFC 4820, March 2007.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU
           Discovery", RFC 4821, March 2007.

[RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla,
           "Authenticated Chunks for the Stream Control Transmission
           Protocol (SCTP)", RFC 4895, August 2007.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC
           4960, September 2007.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M.
           Kozuka, "Stream Control Transmission Protocol (SCTP)
           Dynamic Address Reconfiguration", RFC 5061, September
           2007.

10.2. Informative References

[RFC3424]   Daigle, L. IAB, "IAB Considerations for UNilateral Self-
            Address Fixing (UNSAF) Across Network Address
            Translation", RFC 3424, November 2002.

[RFC6335]   Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S.
            Cheshire, "Internet Assigned Numbers Authority (IANA)
            Procedures for the Management of the Service Name and
            Transport Protocol Port Number Registry", BCP 165, RFC
            6335, August 2011.

[RFC6458]   Stewart, R., Tuexen, M., Poon, K., Lei, P., and V.
            Yasevich, "Sockets API Extensions for the Stream Control
            Transmission Protocol (SCTP)", RFC 6458, December 2011.

[I-D.ietf-6man-udpzero]
            Fairhurst, G. and M. Westerlund, "Applicability Statement
            for the use of IPv6 UDP Datagrams with Zero Checksums",
            draft-ietf-6man-udpzero-12 (work in progress), February
            2013.

[I-D.ietf-behave-sctpnat]
            Stewart, R., Tuexen, M., and I. Ruengeler, "Stream Control
            Transmission Protocol (SCTP) Network Address Translation",
            draft-ietf-behave-sctpnat-08 (work in progress), February
            2013.

[I-D.ietf-tsvwg-natsupp]
            Stewart, R., Tuexen, M., and I. Ruengeler, "Stream Control
            Transmission Protocol (SCTP) Network Address Translation
            Support", draft-ietf-tsvwg-natsupp-05 (work in progress),
            February 2013.

Authors' Addresses

   Michael Tuexen
   Muenster University of Applied Sciences
   Stegerwaldstrasse 39
   48565 Steinfurt
   DE

   Email: tuexen@fh-muenster.de

      Randall R. Stewart
      Adara Networks
      Chapin, SC 29036
      US

      Email: randall@lakerest.net