Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 6, 2013

M. Tuexen
Muenster Univ. of Appl. Sciences
R. Stewart
Adara Networks

July 5, 2012

UDP Encapsulation of SCTP Packets
draft-ietf-tsvwg-sctp-udp-encaps-04.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.

Status of this Memo

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:

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

- 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.

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 proving 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-land is controlling the source address of outgoing packets. This is not an issue when using all available addresses. However, this is not the case when also using the address management required for NAT traversal described in Section 4.7.

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]). It is important to realize that for single homed associations it is only necessary that no IP addresses are listed in the INIT and INIT-ACK chunks. To use multiple addresses, the dynamic address reconfiguration extension described in [RFC5061] MUST be used with wildcard addresses in combination with [RFC4895].

For multi-homed SCTP association the address management as described in Section 4.7 MUST be performed.

4. SCTP over UDP

4.1. Architectural Considerations

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

Each SCTP stack uses a single local UDP encapsulation port number as the destination port for all its incoming SCTP packets. The IANA assigned value of 9989 MAY be used as this port number. 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 UDP encapsulation port number per host can be advantageous (e.g., this reduces the number of mappings in firewalls and NATs, among other things). However, this is not possible if the SCTP stack is implemented as part of an application.

4.2. 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

The UDP checksum MUST NOT be zero.

4.3. Encapsulation Procedure

When inserting the UDP header, the source port is the local UDP encapsulation port number of the SCTP stack, the destination port is the remote UDP encapsulation port number stored for the destination address the packet is sent to (see Section 4.1).

The length of the UDP packet is the length of the SCTP packet plus the size of the UDP header.

The UDP checksum SHOULD be computed and the SCTP checksum MUST be computed.

4.4. Decapsulation Procedure

When an encapsulated packet is received, the UDP header is removed. Then a lookup is performed to find the association the received SCTP packet belongs to. The UDP source port is stored as the encapsulation port for the destination address the SCTP packet is received from (see Section 4.1).

Please note that when a non-encapsulated SCTP packet is received, the encapsulation of outgoing packets belonging to the same association and the corresponding destination address is disabled.

4.5. 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, it MUST be discarded silently. This means in particular that the SCTP stack MUST NOT rely on receiving ICMP or ICMPv6 messages. There MAY be implementation constraints not allowing to process received ICMP or ICMPv6 messages at all.

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.

4.6. 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 4.5.

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

4.7. 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.

4.8. 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 encapsulation and decapsulation. In the other case, ECN MUST NOT be used for such an SCTP association.

5. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is 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.

5.1. 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.

6. IANA Considerations

This document does not require any actions from IANA.

7. Security Considerations

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

8. Acknowledgments

The authors wish to thank Irene Ruengeler and Dan Wing for their invaluable comments.

9. References

9.1. 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. and R. 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.

9.2. Informative References

[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-behave-sctpnat]
Stewart, R., Tuexen, M., and I. Ruengeler, "Stream Control

Transmission Protocol (SCTP) Network Address Translation", draft-ietf-behave-sctpnat-06 (work in progress), March 2012.

[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-02 (work in progress), March 2012.

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