Network Working Group
Internet-Draft
Obsoletes: 6951 (if approved)
Intended status: Standards Track
Expires: 3 September 2025

M. T端xen
M端nster Univ. of Appl. Sciences
R. Stewart

2 March 2025

UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication

draft-tuexen-tsvwg-rfc6951-bis-07

Abstract

This document describes a simple method of encapsulating Stream Control Transmission Protocol (SCTP) packets into UDP packets and its limitations. This allows the usage of SCTP in networks with legacy NATs that do not support 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 needed 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 out of scope for this document.

This document covers only end-hosts and not tunneling (egress or ingress) endpoints. It obsoletes RFC 6951.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 September 2025.

Table of Contents

1. Introduction

This document describes a simple method of encapsulating SCTP packets into UDP packets. SCTP, as defined in [RFC9260], runs directly over IPv4 or IPv6. There are two main reasons for encapsulating SCTP packets:

* To allow SCTP traffic to pass through NATs, which do not provide native SCTP support.

* To allow SCTP to be implemented on hosts that 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", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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, a kernel implementation might not 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 that 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 needed for NAT traversal, described in Section 5.9.

3.2. Legacy NAT Traversal

Using UDP encapsulation allows SCTP communication when traversing NATs not supporting SCTP. 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 (Address Configuration Change Chunks) in combination with [RFC4895].

For multihomed SCTP associations, the address management as described in Section 5.9 MUST be performed.

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

If multiple SCTP endpoints are operating be a NAT, the local SCTP port numbers used by the SCTP endpoints MUST all be different.

4. Unilateral Self-Address Fixing (UNSAF) Considerations

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

Obviously, the exit strategy is to use hosts supporting SCTP natively and middleboxes supporting SCTP.

5. SCTP over UDP

5.1. Architectural Considerations

UDP-encapsulated SCTP is normally communicated between SCTP stacks using the IANA-assigned UDP port number 9899 (sctp-tunneling) on both ends. There are circumstances where other ports MAY be used on either end: As stated earlier, implementations in the application space might be need to use ports other than the registered port. Since NAT boxes might change UDP port numbers, the receiver might observe other UDP port numbers than were used by the sender. Discovery of alternate ports is outside of the scope of this document, but this section describes considerations for SCTP stack

design in light of their potential use.

Each SCTP stack uses a single local UDP encapsulation port number as the destination port for all its incoming SCTP packets. While the uniqueness of the local UDP encapsulation port number is not necessarily needed for the protocol, this greatly simplifies implementation design, since different ports for each address would require a sender implementation to choose the appropriate port while doing source address selection. 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 maintain a remote UDP encapsulation port number per destination address for each SCTP association. Again, because the remote stack MAY be using ports other than the well-known port, each port MAY be different from each stack. However, because of remapping of ports by NATs, the remote ports associated with different remote IP addresses MAY not be identical, even if they are associated with the same stack.

Implementation note: Because the well-known port might not be used, implementations need to allow other port numbers to be specified as a local or remote UDP encapsulation port number through APIs.

5.2. Packet Format

5.2.1. SCTP/UDP/IPv4 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 [RFC9260].

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

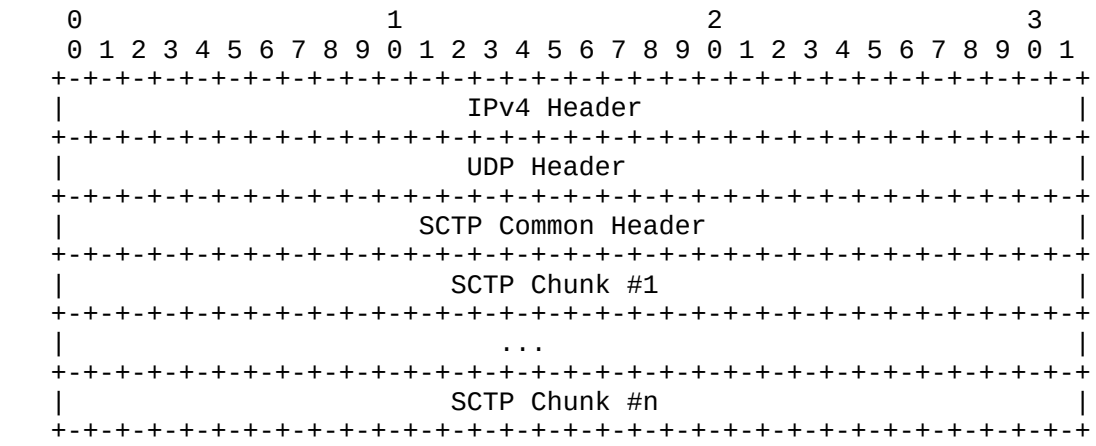

Figure 1: An SCTP/UDP/IPv4 Packet

5.2.2. SCTP/UDP/IPv6 Packet Format

The packet format for an encapsulated SCTP packet when using IPv6 as defined in [RFC8200] is shown in Figure 2. Please note that the number m of IPv6 extension headers can be 0.

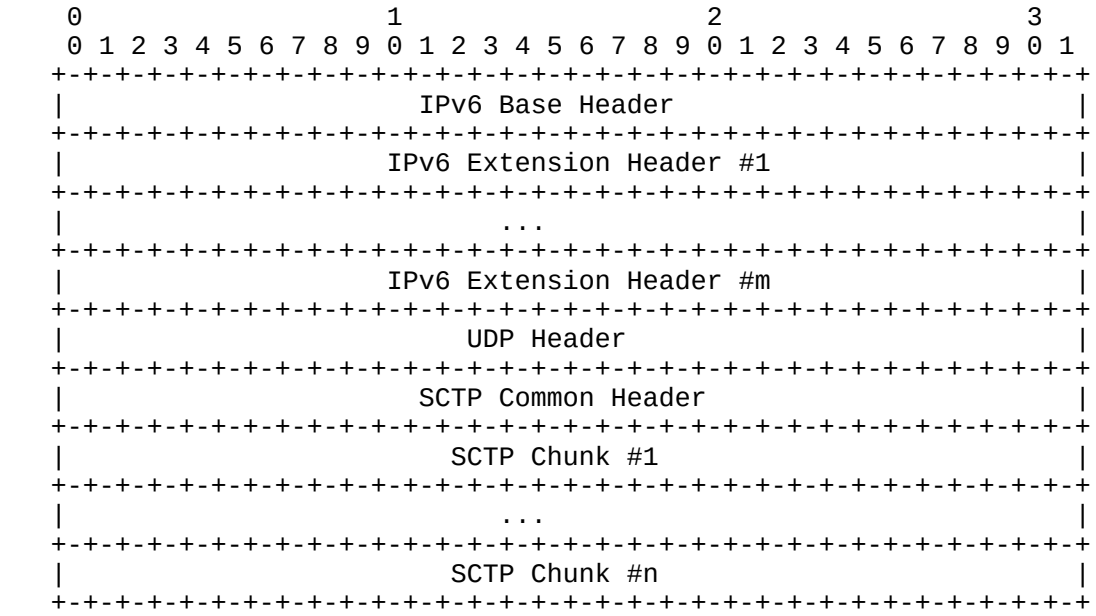

Figure 2: An SCTP/UDP/IPv6 Packet

5.2.3. A New Error Cause

The error cause indicating an "Restart of an Association with New Encapsulation Port" is defined by the following figure.

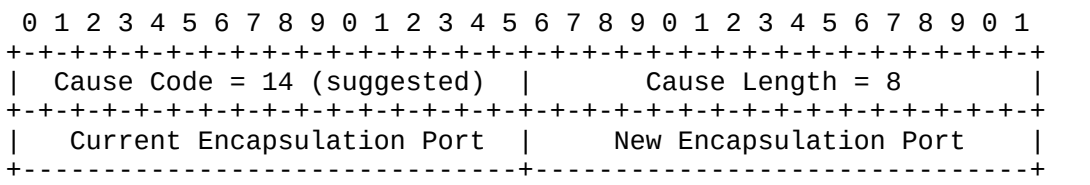

Figure 3: Restart of an Association with New Encapsulation Port Error Cause

Cause Code: 2 bytes (unsigned integer)
: This field holds the IANA defined cause code for the "Restart of an Association with New Encapsulation Port" error cause. IANA is requested to assign the value 14 (suggested) for this cause code.

Cause Length: 2 bytes (unsigned integer)
: This field holds the length in bytes of the error cause; the value MUST be 8.

Current Encapsulation Port: 2 bytes (unsigned integer)
: This field holds the remote encapsulation port currently being used for the destination address the received packet containing the INIT chunk was sent from. If the UDP encapsulation for destination address is currently disabled, 0 is used.

New Encapsulation Port: 2 bytes (unsigned integer)
: If the received SCTP packet containing the INIT chunk is encapsulated in UDP, this field holds the UDP source port number of the UDP packet. If the received SCTP packet is not encapsulated in UDP, this field is 0.

All transported integer numbers are in "network byte order" a.k.a., Big Endian.

5.3. Sending Packets

Within the UDP header, the source port MUST be the local UDP encapsulation port number of the SCTP stack, and the destination port MUST be the remote UDP encapsulation port number maintained 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 for IPv4 and IPv6, and the UDP checksum SHOULD be computed for IPv4 and IPv6. (See [RFC0768] regarding IPv4; see [RFC8200] and [RFC6936] regarding IPv6.) Although UDP with a zero checksum over IPv6 is allowed under certain constraints [RFC6936], this document does not specify mechanisms for this mode. Deployed support might be limited; also, at the time of writing, the use of a zero UDP checksum would be counter to the goal of legacy NAT traversal.

5.4. Receiving Packets

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.

If the verification tag can’t be checked, the procedures described in one of the following sections MUST be followed.

5.5. Handling of SCTP Packets Containing an INIT Chunk Matching an Existing Association

SCTP packets containing an INIT chunk have the verification tag 0 in the common header. Therefore the verification tag can’t be checked.

The following rules apply when processing the received packet:

1. The remote UDP encapsulation port for the source address of the received SCTP packet MUST NOT be updated if the encapsulation of outgoing packets is enabled and the received SCTP packet is encapsulated.

2. The UDP encapsulation for outgoing packets towards the source address of the received SCTP packet MUST NOT be enabled, if it is disabled and the received SCTP packet is encapsulated.

3. The UDP encapsulation for outgoing packets towards the source address of the received SCTP packet MUST NOT be disabled, if it is enabled and the received SCTP packet is not encapsulated.

4. If the UDP encapsulation for outgoing packets towards the source address of the received SCTP packet is disabled and the received SCTP packet is encapsulated, an SCTP packet containing an ABORT chunk MUST be sent. The ABORT chunk MAY include the error cause defined below indicating an "Restart of an Association with New Encapsulation Port". This packet containing the ABORT chunk MUST be encapsulated in UDP. The UDP source port and UDP destination port used for sending the packet containing the ABORT chunk are the UDP destination port and UDP source port of the received packet containing the INIT chunk.

5. If the UDP encapsulation for outgoing packets towards the source address of the received SCTP packet is disabled and the received SCTP packet is not encapsulated, the processing defined in [RFC9260] MUST be performed. If a packet is sent in response, it MUST NOT be encapsulated.

6. If the UDP encapsulation for outgoing packets towards the source address of the received SCTP packet is enabled and the received SCTP packet is not encapsulated, an SCTP packet containing an ABORT chunk MUST be sent. The ABORT chunk MAY include the error cause defined in Section 5.2.3 indicating an "Restart of an Association with New Encapsulation Port". This packet containing the ABORT chunk MUST NOT be encapsulated in UDP.

7. If the UDP encapsulation for outgoing packets towards the source address of the received SCTP packet is enabled and the received SCTP packet is encapsulated, but the UDP source port of the received SCTP packet is not equal to the remote UDP encapsulation port for the source address of the received SCTP packet, an SCTP packet containing an ABORT chunk MUST be sent. The ABORT chunk MAY include the error cause defined in Section 5.2.3 indicating an "Restart of an Association with New Encapsulation Port". This packet containing the ABORT chunk MUST be encapsulated in UDP. The UDP source port and UDP destination port used for sending the packet containing the ABORT chunk are the UDP destination port and UDP source port of the received packet containing the INIT chunk.

8. If the UDP encapsulation for outgoing packets towards the source address of the received SCTP packet is enabled and the received SCTP packet is encapsulated and the UDP source port of the received SCTP packet is equal to the remote UDP encapsulation port for the source address of the received SCTP packet, the

processing defined in [RFC9260] MUST be performed. If a packet is sent in response, it MUST be encapsulated. The UDP source port and UDP destination port used for sending the packet containing the ABORT chunk are the UDP destination port and UDP source port of the received packet containing the INIT chunk.

5.6. Handling of Out of the Blue Packets

If the processing of an out of the blue packet requires the sending of a packet in response according to the rules specified in Section 8.4 of [RFC9260], the following rules apply:

1. If the received packet was encapsulated in UDP, the response packets MUST also be encapsulated in UDP. The UDP source port and UDP destination port used for sending the response packet are the UDP destination port and UDP source port of the received packet.

2. If the received packet was not encapsulated in UDP, the response packet MUST NOT be encapsulated in UDP.

Please note that in these cases a check of the verification tag is not possible.

5.7. ICMP Considerations

When receiving ICMP or ICMPv6 response packets, there might not be enough bytes in the payload to identify the SCTP association that the SCTP packet triggering the ICMP or ICMPv6 packet belongs to. If a received ICMP or ICMPv6 packet cannot be related to a specific SCTP association or the verification tag cannot be verified, it MUST be discarded silently. In particular, this means 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 'Destination Unreachable' and code 'Protocol 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.

5.8. 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.

Since one cannot rely on the feedback provided by ICMP or ICMPv6 due to the limitation laid out in Section 5.7, [RFC8899] MUST be followed, when performing Path MTU discovery.

If the implementation does not allow control of the Don’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.

5.9. 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 multihomed SCTP association is set up initially as a single-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, the protocol extension [RFC5061] MUST be used with wildcard addresses only. If an SCTP endpoint 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.

5.10. Explicit Congestion Notification (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 might 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. Middlebox Considerations

Middleboxes often use different timeouts for UDP based flows than for other flows. Therefore the HEARTBEAT.Interval parameter SHOULD be lowered to 15 seconds when UDP encapsulation is used.

8. IANA Considerations

[NOTE to RFC-Editor: "RFCXXXX" is to be replaced by the RFC number you assign this document.]

[NOTE to RFC-Editor: The requested values for the cause code are tentative and to be confirmed by IANA.]

A new error cause code has to be assigned by IANA. This requires an additional line in the "Error Cause Codes" registry for SCTP:

Value	Cause Code	Reference
14 (suggested)	Restart of an Association with New Encapsulation Port	[RFCXXXX]

Table 1: New Entry in Error Cause Codes Registry

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

9. Security Considerations

Encapsulating SCTP into UDP does not add any additional security considerations to the ones given in [RFC9260] and [RFC5061].

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

An attacker might send a malicious UDP packet towards an SCTP endpoint 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 of the two negotiated verification tags. This protects against blind attackers the same way as described in [RFC9260] 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.

10. References

10.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <https://www.rfc-editor.org/info/rfc792>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, DOI 10.17487/RFC4895, August 2007, <https://www.rfc-editor.org/info/rfc4895>.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, DOI 10.17487/RFC5061, September 2007, <https://www.rfc-editor.org/info/rfc5061>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8899] Fairhurst, G., Jones, T., T端xen, M., R端ngeler, I., and T. V旦lker, "Packetization Layer Path MTU Discovery for Datagram Transports", RFC 8899, DOI 10.17487/RFC8899, September 2020, <https://www.rfc-editor.org/info/rfc8899>.

[RFC9260] Stewart, R., T端xen, M., and K. Nielsen, "Stream Control Transmission Protocol", RFC 9260, DOI 10.17487/RFC9260, June 2022, <https://www.rfc-editor.org/info/rfc9260>.

10.2. Informative References

[RFC3424] Daigle, L., Ed. and IAB, "IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation", RFC 3424, DOI 10.17487/RFC3424, November 2002, <https://www.rfc-editor.org/info/rfc3424>.

[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, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, DOI 10.17487/RFC6458, December 2011, <https://www.rfc-editor.org/info/rfc6458>.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <https://www.rfc-editor.org/info/rfc6936>.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, DOI 10.17487/RFC6951, May 2013, <https://www.rfc-editor.org/info/rfc6951>.

Acknowledgments

The authors wish to thank Stewart Bryant, Dave Crocker, Gorry Fairhurst, Tero Kivinen, Barry Leiba, Pete Resnick, Martin Stiemerling, Irene R端ngeler, and Dan Wing for their invaluable comments on [RFC6951].

The authors wish to thank Georgios Papastergiou for the initial problem report resulting in the work on this document and Irene R端ngeler and Felix Weinrank for their invaluable comments on this document.

Part of this work has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 644334 (NEAT). The views expressed are solely those of the author(s).

Authors’ Addresses

Michael T端xen
M端nster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
Germany

Email: tuexen@fh-muenster.de

Randall R. Stewart
15214 Pendio Drive
Bella Collina, FL 34756
United States of America
Email: randall@lakerest.net