Transport Area Working Group
Internet-Draft
Intended status: Experimental
Expires: September 12, 2019

M. Amend
Deutsche Telekom
A. Brunstrom
A. Kassler
Karlstad University
V. Rakocevic
City University of London
March 11, 2019

Lossless and overhead free DCCP - UDP header conversion (U-DCCP)
draft-amend-tsvwg-dccp-udp-header-conversion-00

Abstract

The Datagram Congestion Control protocol (DCCP) is a non-widely deployed transport protocol in the Internet. The reason for that is a typical chicken-egg problem. Even if there would be a use for application developer to rely on DCCP, middle-boxes like firewalls and NATs will prevent DCCP end-to-end since they lack support for DCCP. However, as long as the protocol penetration of DCCP will not increase, middle-boxes will not handle DCCP properly. To overcome this challenge NAT/NATP traversal and UDP encapsulation for DCCP is already defined. However the first requires special middle-box support and the latter introduces overhead. The proposal of a multipath extension for DCCP further stresses the question of efficient middle-box passing as its main goal is to be applied over the Internet, traversing numerous uncontrolled middle-boxes. This document introduces a new approach, disguising DCCP during transmission as UDP without requiring middle-box modification nor introducing any overhead.

Copyright Notice

Table of Contents

1. Introduction

The Datagram Congestion Control Protocol (DCCP) [RFC4340] is a transport-layer protocol that provides upper layers with the ability to use non-reliable congestion-controlled flows. The current specification for DCCP [RFC4340] specifies a direct native encapsulation in IPv4 or IPv6 packets.

DCCP support has been specified for devices that use Network Address Translation (NAT) or Network Address and Port Translation (NAPT) [RFC5597]. However, there is a significant installed base of NAT/ NAPT devices that do not support [RFC5597]. An UDP encapsulation for

DCCP [RFC6773] circumvents such limitations and makes DCCP compatible with any UDP [RFC768] compliant device that support [RFC4787] but do not support [RFC5597]. For convenience, the standard encapsulation for DCCP [RFC4340] (including [RFC5596] and [RFC5597] as required) is referred to as DCCP-STD, whereas the UDP encapsulation for DCCP [RFC6773] is referred to as DCCP-UDP.

It can be stated, that DCCP-STD and DCCP-UDP are techniques, which increases the success rate of DCCP transmissions significantly. However, DCCP-STD fails on devices that blocks DCCP for any reasons. On the other hand, DCCP-UDP uses the well-accepted UDP to let devices assume they handle the UDP protocol, but for the cost of a reduced goodput/throughput ratio.

To compensate the inefficiency of DCCP-STD (device blocking) and DCCP-UDP (overhead), this document proposes a beneficial modification scheme relying like DCCP-UDP on UDP, but for no overhead. This goal is reached by re-arranging DCCP's extended header in that it looks like UDP, without losing critical information and is referred to as U-DCCP.

U-DCCP is limited to DCCP's extended header, requiring X is set to 1. Otherwise U-DCCP relies on the NAT/NATP functionalities specified for UDP in [RFC4787], [RFC6888] and [RFC7857].

2. Terminology

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. U-DCCP

3.1. Overview

The basic approach of U-DCCP is to modify the extended header of a DCCP packet such that it appears like UDP [RFC768]. In particular, this takes place without losing information, but requires a U-DCCP termination before the packet is delivered to the DCCP end system (!!! Auto-detection via e.g. SDP required??? !!!). The method does not change the 4-tuple of IP and port addressing, however it changes the protocol carried over IP to UDP. In consequence, the length of the packet remains unchanged and behaves like DCCP-STD. The solution is not a tunneling approach. It requires that the same port as for DCCP can be used by UDP.

The method is designed to support use when these addresses are modified by a device that implements NAT/NAPT. A NAT translates the

IP addresses, which impacts the transport-layer checksum. A NAPT device may also translate the port values (usually the source port). In both cases, the outer transport header that includes these values would need to be updated by the NAT/NAPT.

U-DCCP supports IPv4 and IPv6.

The basic format of a U-DCCP packet is:

```
+------------------------------------+
|     IP Header (IPv4 or IPv6)       |  Variable length
+------------------------------------+
|UDP like arranged DCCP ext. Header |  8 bytes \
+------------------------------------+           ) U-DCCP header
|Rest of rearranged DCCP ext. Header|  8 bytes /
+------------------------------------+
| Additional (type-specific) Fields |  Variable length (could be 0)
+------------------------------------+
|           DCCP Options             |  Variable length (could be 0)
+------------------------------------+
|      Application Data Area         |  Variable length (could be 0)
+------------------------------------+
```

Figure 1: Format of U-DCCP packet

The U-DCCP header is described in Section 3.4 after introducing the traditional DCCP header in Section 3.1 and its target appearance of a UDP header in Section 3.2. Section 3.3 discusses considerations for building the U-DCCP header upfront.

3.2. The DCCP Generic header

The DCCP Generic Header [RFC4340] takes two forms, one with long sequence numbers (48 bits) and the other with short sequence numbers (24 bits), which is not part of U-DCCP's modification.

```
      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
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |          Source Port          |           Dest Port           |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |  Data Offset  | CCVal | CsCov |           Checksum            |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |     |       |X|               |                               .
     | Res | Type  |=|   Reserved    |  Sequence Number (high bits)  .
     |     |       |1|               |                               .
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                  Sequence Number (low bits)                   |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: The extended DCCP Header with Long Sequence Numbers [RFC4340]

```
      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
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |          Source Port          |           Dest Port           |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |  Data Offset  | CCVal | CsCov |           Checksum            |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |     |       |X|                                               |
     | Res | Type  |=|   Sequence Number (low bits)                  |
     |     |       |0|                                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: The short DCCP Header with Short Sequence Numbers [RFC4340]

All generic header fields have the meaning specified in [RFC4340], updated by [RFC5596].

3.3. UDP header

```
      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
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |          Source Port          |           Dest Port           |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |             Length            |           Checksum            |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: The UDP Header [RFC768]

All header fields have the meaning specified in [RFC768].

3.4. U-DCCP conversion considerations

The U-DCCP header has the goal to merge the information of DCCP's extended header (Section 3.1) and imitates in the beginning the UDP header (Section 3.2). Information, to restore a DCCP header from any conversion, which must not be lost are source and destination port, Data Offset, CCVal, CsCov, Checksum, Type, X and the Sequence Number.

Compared with the UDP header, the DCCP ext. header shows similarities in source and destination port and checksum. The length field of UDP (bits 33-48) is not part of the DCCP header and contains in case of DCCP the fields Data Offset, CCVal and CsCov.

For the goal of imitating UDP, the checksum must cover the whole datagram, which renders any limitation by CsCov useless. The checksum itself is required to re-calculate after conversion anyway.

If the conversion is limited to DCCP'S extended header only, X is always "1".

Thus, Data Offset, CCVal, Type and Sequence Number must be re-arranged in a way that the Length field of UDP can be applied.

3.5. U-DCCP header

The considerations of Section 3.3 leads to the following header, denoted as U-DCCP header.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
U  |          Source Port          |           Dest Port           |
D  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
P  |          Length               |           Checksum            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Type  | CCVal |  Data Offset  |  Sequence Number (high bits)  .
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   .                   Sequence Number (low bits)                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: The U-DCCDP Header

The first 8 bytes of the U-DCCP header corresponds to [RFC768] and the fields are interpreted as follows:

Source and Dest(ination) Ports: 16 bits each

These fields identify the UDP ports on which the source and destination (respectively) of the packet are listening for incoming UDP packets. The UDP port values do identify the DCCP source and destination ports.

Length: 16 bits

This field is the length of the UDP datagram, including the UDP header and the payload (for U-DCCP, the payload comprises the payload of the original DCCP datagram and part of its header).

Checksum: 16 bits

This field is the Internet checksum of a network-layer pseudoheader and Length bytes of the UDP packet [RFC768]. The UDP checksum MUST NOT be zero for a U-DCCP packet.

The remaining 8 bytes of the U-DCCP header contains:

Type, CCVal, Data Offset, Seq. Number: As specified in [RFC4340]

In case U-DCCP is applied, the IP network layer must be instructed to carry an UDP datagram and its checksum must be re-calculated, for detailed information see section 4.

3.6. Implementation

The process of applying U-DCCP behaves as follows and requires:

DCCP generation -> U-DCCP conversion -> UDP transmission -> U-DCCP reception and restoration -> DCCP reception

The conversion can be integrated into DCCP endpoints directly or as an additional component on the way along the transmission route. Depending on the degree of integration, especially the process of checksum calculation and validation can be optimized. Section 4.1 and 4.2 provides a possible pseudo-code for the conversion without any optimized integration into the sender's network stack nor in the receiver's network stack. The pseudo-code assumes explicit knowledge about which U-DCCP flows need conversion between sender and receiver.

3.7. Pseudo-code DCCP to U-DCCP conversion

A possible processing of an already generated DCCP datagram for U-DCCP conversion:

1. Receive DCCP datagram.

2. Check eligibility for conversion otherwise bypass conversion.

3. Verify consistency, e.g. checksum otherwise drop.

4. Shift Type and CCVal field to the ninth octet.

5. Shift Data Offset field to the tenth octet.

6. Place a length information at octet 5+6 corresponding to [RFC768].

7. Modify the IP header's encapsulated protocol from DCCP to UDP.

8. Re-calculate IP header checksum.

9. Reset DCCP checksum field: octet 7+8 = 0.

10. Generate new checksum at octet 7+8 as described in [RFC768].

11. Forward to destination based on the unmodified 4-tuple of IP-addresses and ports.

3.8. Pseudo-code U-DCCP to DCCP restoration

A possible processing of an already converted U-DCCP datagram for DCCP restoration:

1. Receive UDP datagram.

2. Check eligibility for restoration otherwise bypass restoration

3. Validate UDP checksum otherwise drop.

4. Restore Data Offset field according to [RFC4340].

5. Restore CCVal field according to [RFC4340].

6. Set CsCov field according to [RFC4340] to "0".

7. Restore Type field according to [RFC4340].

8. Set Reserved bits according to [RFC4340] to "0".

9. Set X according to [RFC4340] to "1".

10. Modify the IP header's encapsulated protocol from UDP to DCCP.

11. Re-calculate IP header checksum.

12. Reset DCCP checksum field: octet 7+8 = 0.

13. Generate new checksum at octet 7+8 as described in [RFC768].

14. Forward to destination based on the unmodified 4-tuple of IP-addresses and ports.

3.9. U-DCCP negotiation (required????)

Tbd later if required. Otherwise assumes explicit knowledge about the U-DCCP conversion between sender and receiver.

4. Security Considerations

TBD.

5. IANA Considerations

6. Notes

This document is inspired by [RFC6773] and some text passages for the -00 version are copied unmodified.

7. Acknowledgments

8. Informative References

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

[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>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <https://www.rfc-editor.org/info/rfc4340>.

[RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <https://www.rfc-editor.org/info/rfc4787>.

[RFC5596] Fairhurst, G., "Datagram Congestion Control Protocol (DCCP) Simultaneous-Open Technique to Facilitate NAT/Middlebox Traversal", RFC 5596, DOI 10.17487/RFC5596, September 2009, <https://www.rfc-editor.org/info/rfc5596>.

[RFC5597] Denis-Courmont, R., "Network Address Translation (NAT) Behavioral Requirements for the Datagram Congestion Control Protocol", BCP 150, RFC 5597, DOI 10.17487/RFC5597, September 2009, <https://www.rfc-editor.org/info/rfc5597>.

[RFC6773] Phelan, T., Fairhurst, G., and C. Perkins, "DCCP-UDP: A Datagram Congestion Control Protocol UDP Encapsulation for NAT Traversal", RFC 6773, DOI 10.17487/RFC6773, November 2012, <https://www.rfc-editor.org/info/rfc6773>.

[RFC6888] Perreault, S., Ed., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common Requirements for Carrier-Grade NATs (CGNs)", BCP 127, RFC 6888, DOI 10.17487/RFC6888, April 2013, <https://www.rfc-editor.org/info/rfc6888>.

[RFC7857] Penno, R., Perreault, S., Boucadair, M., Ed., Sivakumar, S., and K. Naito, "Updates to Network Address Translation (NAT) Behavioral Requirements", BCP 127, RFC 7857, DOI 10.17487/RFC7857, April 2016, <https://www.rfc-editor.org/info/rfc7857>.

Authors' Addresses

Markus Amend
Deutsche Telekom
T-Online-Allee 1
Darmstadt
Germany

Email: Markus.Amend@telekom.de

Anna Brunstrom
Karlstad University
Universitetsgatan 2
651 88 Karlstad
Sweden

Email: anna.brunstrom@kau.se

Andreas Kassler
Karlstad University
Universitetsgatan 2
651 88 Karlstad
Sweden

Email: andreas.kassler@kau.se

Veselin Rakocevic
City University of London
Northampton Square
London
United Kingdom

Email: veselin.rakocevic.1@city.ac.uk