

Transport Area Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: January 9, 2020

M. Amend  
Deutsche Telekom  
A. Brunstrom  
A. Kassler  
Karlstad University  
V. Rakocevic  
City University of London  
July 08, 2019

Lossless and overhead free DCCP - UDP header conversion (U-DCCP)  
draft-amend-tsvwg-dccp-udp-header-conversion-01

## Abstract

The Datagram Congestion Control Protocol (DCCP) is a transport-layer protocol that provides upper layers with the ability to use non-reliable congestion-controlled flows. DCCP is not widely deployed in the Internet, and the reason for that can be defined as a typical example of a chicken-egg problem. Even if an application developer decided to use DCCP, the middle-boxes like firewalls and NATs would prevent DCCP end-to-end since they lack support for DCCP. Moreover, as long as the protocol penetration of DCCP does not increase, the middle-boxes will not handle DCCP properly. To overcome this challenge, NAT/NATP traversal and UDP encapsulation for DCCP is already defined. However, the former requires special middle-box support and the latter introduces overhead. The recent proposal of a multipath extension for DCCP further underlines the challenge 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 solution which disguises DCCP during transmission as UDP without requiring middle-box modification or introducing any overhead.

## 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 January 9, 2020.

#### Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

#### Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. U-DCCP . . . . .	3
3.1. Overview . . . . .	3
3.2. The DCCP Generic header . . . . .	4
3.3. UDP header . . . . .	5
3.4. U-DCCP conversion considerations . . . . .	6
3.5. U-DCCP header . . . . .	6
3.6. Implementation . . . . .	7
3.7. Pseudo-code DCCP to U-DCCP conversion . . . . .	7
3.8. Pseudo-code U-DCCP to DCCP restoration . . . . .	8
3.9. U-DCCP negotiation (required???) . . . . .	9
4. Security Considerations . . . . .	9
5. IANA Considerations . . . . .	9
6. Notes . . . . .	9
7. Acknowledgments . . . . .	9
8. Informative References . . . . .	9
Authors' Addresses . . . . .	10

#### 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 [RFC0768] compliant device that supports [RFC4787] but does 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 increase the success rate of DCCP transmissions significantly. However, DCCP-STD fails on devices that block DCCP for any reasons. On the other hand, DCCP-UDP uses the well-accepted UDP to let devices assume they are handling the UDP protocol, but at the cost of a reduced goodput/throughput ratio.

To compensate for the inefficiency of DCCP-STD (device blocking) and DCCP-UDP (overhead), this document proposes a beneficial modification scheme relying on UDP (like DCCP-UDP), but with no overhead. This goal is reached by re-arranging DCCP's extended header to make it look like UDP, without losing critical information. This solution 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 so that it appears like UDP [RFC0768]. In particular, this takes place without losing any header information, but requires a U-DCCP termination before the packet is delivered to the DCCP end system. This method does not change the 4-tuple of IP and port addressing, however it changes the protocol carried over IP from DCCP to UDP. As a 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 used by DCCP can be used by UDP.

The method is designed to support use when the IP 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 \
+-----+		
	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). The short one is not part of U-DCCP's modification.

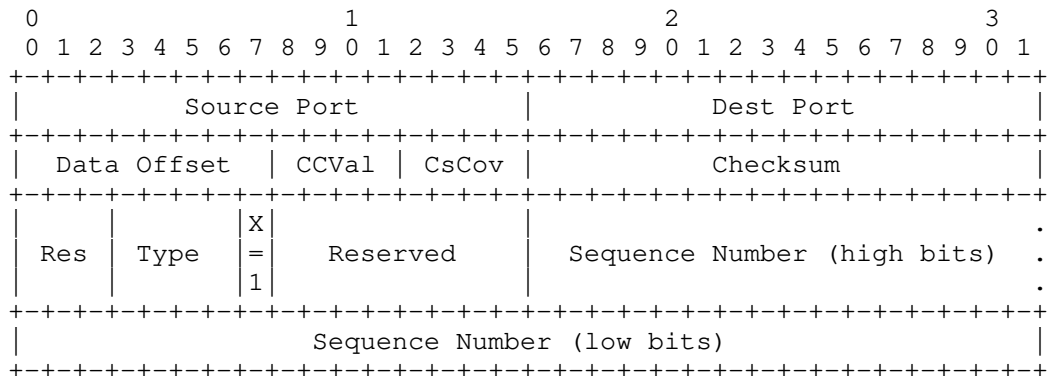


Figure 2: The extended DCCP Header with Long Sequence Numbers  
[RFC4340]

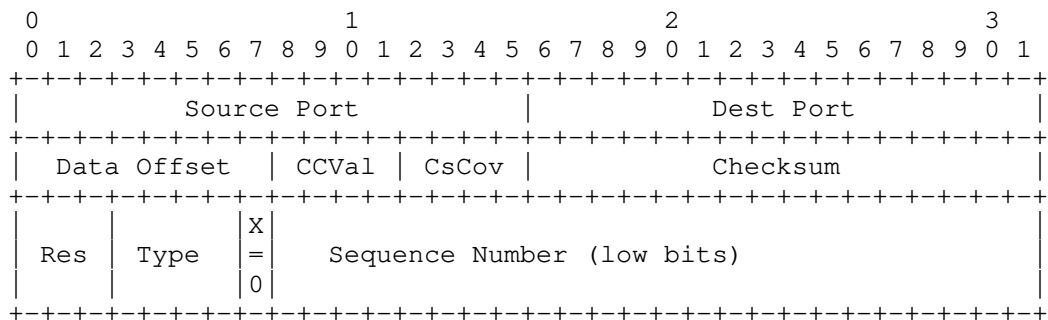


Figure 3: The short DCCP Header with Short Sequence Numbers [RFC4340]

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

### 3.3. UDP header

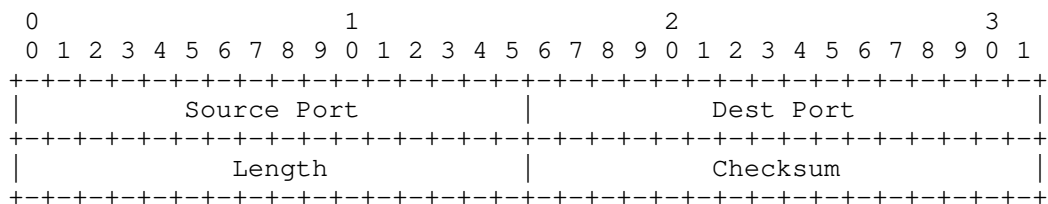


Figure 4: The UDP Header [RFC768]

All header fields have the meaning specified in [RFC0768].

### 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 first 64 bits the UDP header (Section 3.2). Information required to restore a DCCP header from any conversion, which must not be lost, includes: source and destination port, Data Offset, CCVal, CsCov, Checksum, Type, X and the Sequence Number.

Compared with the UDP header, the DCCP extended 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.

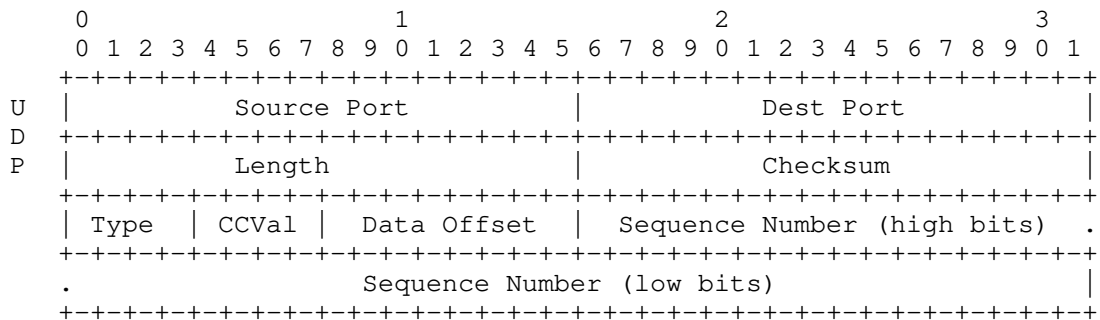


Figure 5: The U-DCCDP Header

The first 8 bytes of the U-DCCP header corresponds to [RFC0768] and the fields are interpreted as follows:

Source and Dest(ination) Ports: 16 bits each

These fields identify the UDP ports used by the source and destination (respectively) of the packet to listen for incoming UDP packets. The UDP port values 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 [RFC0768]. 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 layer must be instructed to carry an UDP datagram and its checksum must be re-calculated. For detailed information see Section 3.7.

### 3.6. Implementation

The process of applying U-DCCP is defined as follows:

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 3.7 and Section 3.8 provide a possible pseudo-code for the conversion without any optimized integration into the sender's network stack or into the receiver's network stack. The pseudo-code assumes explicit knowledge on which U-DCCP flows need conversion between the sender and the 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 [RFC0768].
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 [RFC0768].
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 [RFC0768].
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  
Deutsche-Telekom-Allee 7  
64295 Darmstadt  
Germany

Email: [Markus.Amend@telekom.de](mailto:Markus.Amend@telekom.de)

Anna Brunstrom  
Karlstad University  
Universitetsgatan 2  
651 88 Karlstad  
Sweden

Email: [anna.brunstrom@kau.se](mailto: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

Transport Area Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: 13 January 2022

M. Amend  
D. Hugo  
DT  
A. Brunstrom  
A. Kassler  
Karlstad University  
V. Rakocevic  
City University of London  
S. Johnson  
BT  
12 July 2021

DCCP Extensions for Multipath Operation with Multiple Addresses  
draft-amend-tsvwg-multipath-dccp-05

Abstract

DCCP communication is currently restricted to a single path per connection, yet multiple paths often exist between peers. The simultaneous use of these multiple paths for a DCCP session could improve resource usage within the network and, thus, improve user experience through higher throughput and improved resilience to network failures. Use cases for a Multipath DCCP (MP-DCCP) are mobile devices (handsets, vehicles) and residential home gateways simultaneously connected to distinct paths as, e.g., a cellular link and a WiFi link or to a mobile radio station and a fixed access network. Compared to existing multipath protocols such as MPTCP, MP-DCCP provides specific support for non-TCP user traffic as UDP or plain IP. More details on potential use cases are provided in [website], [slide] and [paper]. All these use cases profit from an Open Source Linux reference implementation provided under [website].

This document presents a set of extensions to traditional DCCP to support multipath operation. Multipath DCCP provides the ability to simultaneously use multiple paths between peers. The protocol offers the same type of service to applications as DCCP and it provides the components necessary to establish and use multiple DCCP flows across potentially disjoint paths.

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 13 January 2022.

#### Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

#### Table of Contents

1. Introduction . . . . .	3
1.1. Multipath DCCP in the Networking Stack . . . . .	4
1.2. Terminology . . . . .	4
1.3. MP-DCCP Concept . . . . .	5
1.4. Differences from Multipath TCP . . . . .	5
1.5. Requirements Language . . . . .	9
2. Operation Overview . . . . .	9
3. MP-DCCP Protocol . . . . .	9
3.1. Multipath Capable Feature . . . . .	12
3.2. Multipath Option . . . . .	12
3.2.1. MP_CONFIRM . . . . .	13
3.2.2. MP_JOIN . . . . .	13
3.2.3. MP_FAST_CLOSE . . . . .	14
3.2.4. MP_KEY . . . . .	14
3.2.5. MP_SEQ . . . . .	15
3.2.6. MP_HMAC . . . . .	15
3.2.7. MP_RTT . . . . .	16
3.2.8. MP_ADDADDR . . . . .	17
3.2.9. MP_REMOVEADDR . . . . .	18
3.2.10. MP_PRIO . . . . .	19

3.3. MP-DCCP Handshaking Procedure . . . . .	19
4. Security Considerations . . . . .	21
5. Interactions with Middleboxes . . . . .	22
6. Implementation . . . . .	22
7. Acknowledgments . . . . .	22
8. IANA Considerations . . . . .	23
9. Informative References . . . . .	25
Authors' Addresses . . . . .	28

## 1. Introduction

Multipath DCCP (MP-DCCP) is a set of extensions to regular DCCP [RFC4340], i.e. the Datagram Congestion Control Protocol denoting a transport protocol that provides bidirectional unicast connections of congestion-controlled unreliable datagrams. A multipath extension to DCCP enables the transport of user data across multiple paths simultaneously. This is beneficial to applications that transfer fairly large amounts of data, due to the possibility to aggregate capacity of the multiple paths. In addition, it enables to tradeoff timeliness and reliability, which is important for low latency applications that do not require guaranteed delivery services such as Audio/Video streaming. DCCP multipath operation is suggested in the context of ongoing 3GPP work on 5G multi-access solutions [I-D.amend-tsvwg-multipath-framework-mpdccp] and for hybrid access networks [I-D.lhwxyz-hybrid-access-network-architecture][I-D.muley-net-work-based-bonding-hybrid-access]. It can be applied for load-balancing, seamless session handover, and aggregation purposes (referred to as ATSSS; Access steering, switching, and splitting in 3GPP terminology [TS23.501]).

This document presents the protocol changes required to add multipath capability to DCCP; specifically, those for signaling and setting up multiple paths ("subflows"), managing these subflows, re-assembly of data, and termination of sessions. DCCP, as stated in [RFC4340] does not provide reliable and ordered delivery. Consequently, multiple application subflows may be multiplexed over a single DCCP connection with no inherent performance penalty for flows that do not require in-ordered delivery. DCCP does not provide built-in support for those multiple application subflows.

In the following, use of the term subflow will refer to physical separate DCCP subflows transmitted via different paths, but not to application subflows. Application subflows are differing content-wise by source and destination port per application as, for example, enabled by Service Codes introduced to DCCP in [RFC5595], and those subflows can be multiplexed over a single DCCP connection. For sake of consistency we assume that only a single application is served by a DCCP connection here as shown in Figure 1 while use of that feature should not impact DCCP operation on each single path as noted in ([RFC5595], sect. 2.4).

### 1.1. Multipath DCCP in the Networking Stack

MP-DCCP operates at the transport layer and aims to be transparent to both higher and lower layers. It is a set of additional features on top of standard DCCP; Figure 1 illustrates this layering. MP-DCCP is designed to be used by applications in the same way as DCCP with no changes to the application itself.

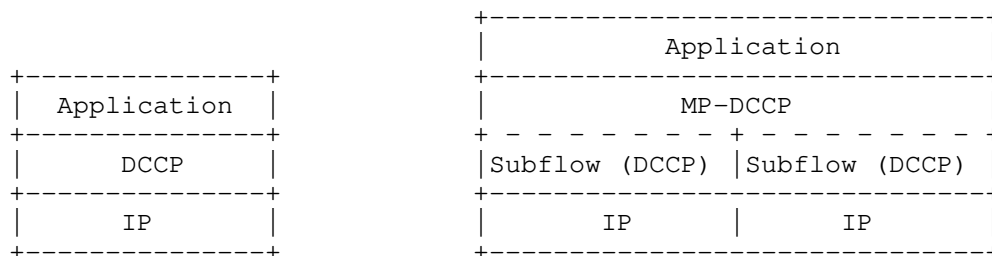


Figure 1: Comparison of Standard DCCP and MP-DCCP Protocol Stacks

### 1.2. Terminology

Throughout this document we make use of terms that are either specific for multipath transport or are defined in the context of MP-DCCP, similar to [RFC8684], as follows:

**Path:** A sequence of links between a sender and a receiver, defined in this context by a 4-tuple of source and destination address/ port pairs.

**Subflow:** A flow of DCCP segments operating over an individual path, which forms part of a larger MP-DCCP connection. A subflow is started and terminated similar to a regular (single-path) DCCP connection.

(MP-DCCP) Connection: A set of one or more subflows, over which an application can communicate between two hosts. There is a one-to-one mapping between a connection and an application socket.

Token: A locally unique identifier given to a multipath connection by a host. May also be referred to as a "Connection ID".

Host: An end host operating an MP-DCCP implementation, and either initiating or accepting an MP-DCCP connection. In addition to these terms, within framework of MP-DCCP the interpretation of, and effect on, regular single-path DCCP semantics is discussed in Section 3.

### 1.3. MP-DCCP Concept

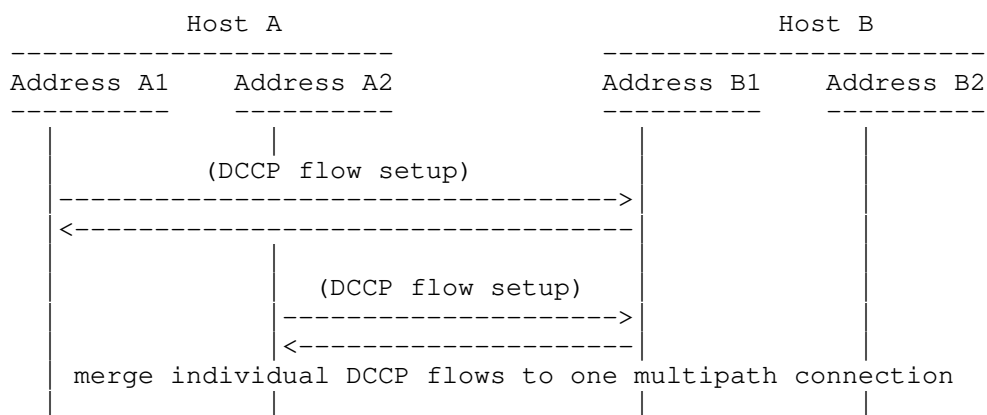


Figure 2: Example MP-DCCP Usage Scenario

### 1.4. Differences from Multipath TCP

Multipath DCCP is similar to Multipath TCP [RFC8684], in that it extends the related basic DCCP transport protocol [RFC4340] with multipath capabilities in the same way as Multipath TCP extends TCP [RFC0793]. However, because of the differences between the underlying TCP and DCCP protocols, the transport characteristics of MPTCP and MP-DCCP are different.



Table 1 compares the protocol characteristics of TCP and DCCP, which are by nature inherited by their respective multipath extensions. A major difference lies in the delivery of payload, which is for TCP an exact copy of the generated byte-stream. DCCP behaves in a different way and does not guarantee to deliver any payload nor the order of delivery. Since this is mainly affecting the receiving endpoint of a TCP or DCCP communication, many similarities on the sender side can be identified. Both transport protocols share the 3-way initiation of a communication and both employ congestion control to adapt the sending rate to the path characteristics.

Feature	TCP	DCCP
Full-Duplex	yes	yes
Connection-Oriented	yes	yes
Header option space	40 bytes	< 1008 bytes or PMTU
Data transfer	reliable	unreliable
Packet-loss handling	re-transmission	report only
Ordered data delivery	yes	no
Sequence numbers	one per byte	one per PDU
Flow control	yes	no
Congestion control	yes	yes
ECN support	yes	yes
Selective ACK	yes	depends on congestion control
Fix message boundaries	no	yes
Path MTU discovery	yes	yes
Fragmentation	yes	no
SYN flood protection	yes	no
Half-open connections	yes	no

Table 1: TCP and DCCP protocol comparison

Consequently, the multipath features, shown in Table 2, are the same, supporting volatile paths having varying capacity and latency, session handover and path aggregation capabilities. All of them profit by the existence of congestion control.

Feature	MPTCP	MP-DCCP
Volatile paths	yes	yes
Session handover	yes	yes
Path aggregation	yes	yes
Robust session establishment	no	yes
Data re-assembly	yes	optional / modular
Expandability	limited by TCP header	flexible

Table 2: MPTCP and MP-DCCP protocol comparison

Therefore, the sender logic is not much different between MP-DCCP and MPTCP, even if the multipath session initiation differs. MP-DCCP inherits a robust session establishment feature, which guarantees communication establishment if at least one functional path is available. MPTCP relies on an initial path, which has to work; otherwise no communication can be established.

The receiver side for MP-DCCP has to deal with the unreliable transport character of DCCP and a possible re-assembly of the data stream while not advocating it. As many unreliable applications have built-in application support for reordering (such as adaptive audio and video buffers), those applications might not need support for re-assembly. However, for applications that benefit from partial or full support of reordering, MP-DCCP can provide flexible support for re-assembly, even if for DCCP the order of delivery is unreliable by nature. Such optional re-assembly mechanisms may account for the fact that packet loss may occur for any of the DCCP subflows. Another issue may occur as packet reordering may happen when the different DCCP subflows are routed across paths with different latencies. In theory, applications using DCCP are aware that packet reordering might happen, since DCCP has no mechanisms to prevent it.

The receiving process for MPTCP is on the other hand a rigid "just wait" approach, since TCP guarantees reliable delivery.

### 1.5. Requirements Language

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].

## 2. Operation Overview

RFC 4340 states that some applications might want to share congestion control state among multiple DCCP flows between same source and destination addresses. This functionality could be provided by the Congestion Manager (CM) [RFC3124], a generic multiplexing facility. However, the CM would not fully support MP-DCCP without change; it does not gracefully handle multiple congestion control mechanisms, for example.

The operation of MP-DCCP for data transfer takes one input data stream from an application, and splits it into one or more subflows, with sufficient control information to allow received data to be re-assembled and delivered in order to the recipient application. The following subsections define this behavior in detail.

The Multipath Capability for MP-DCCP can be negotiated with a new DCCP feature, as described in Section 3. Once negotiated, all subsequent MP-DCCP operations are signalled with a variable length multipath-related option, as described in Section 3.1.

## 3. MP-DCCP Protocol

The DCCP protocol feature list ([RFC4340] section 6.4) will be enhanced by a new Multipath related feature with Feature number 10, as shown in Table 3.

Number	Meaning	Rule	Rec'n Value	Initial Req'd
0	Reserved			
1	Congestion Control ID (CCID)	SP	2	Y
2	Allow Short Seqnos	SP	0	Y
3	Sequence Window	NN	100	Y
4	ECN Incapable	SP	0	N
5	Ack Ratio	NN	2	N
6	Send Ack Vector	SP	0	N
7	Send NDP Count	SP	0	N
8	Minimum Checksum Coverage	SP	0	N
9	Check Data Checksum	SP	0	N
10	Multipath Capable	SP	0	N
11-127	Reserved			
128-255	CCID-specific features			

Table 3: Proposed Feature Set

The DCCP protocol options as defined in ([RFC4340] section 5.8) and ([RFC5634] section 2.2.1) will be enhanced by a new Multipath related variable-length option with option type 46, as shown in Table 4.

Type	Option Length	Meaning	DCCP-Data?
0	1	Padding	Y
1	1	Mandatory	N
2	1	Slow Receiver	Y
3-31	1	Reserved	
32	variable	Change L	N
33	variable	Confirm L	N
34	variable	Change R	N
35	variable	Confirm R	N
36	variable	Init Cookie	N
37	3-8	NDP Count	Y
38	variable	Ack Vector [Nonce 0]	N
39	variable	Ack Vector [Nonce 1]	N
40	variable	Data Dropped	N
41	6	Timestamp	Y
42	6/8/10	Timestamp Echo	Y
43	4/6	Elapsed Time	N
44	6	Data Checksum	Y
45	8	Quick-Start Response	?
46	variable	Multipath	Y
47-127	variable	Reserved	
128-255	variable	CCID-specific options	-

Table 4: Proposed Option Set

[Tbd/tbv] In addition to the multipath option, MP-DCCP requires particular considerations for:

- \* The minimum PMTU of the individual paths must be announced to the application. Changes of individual path PMTUs must be re-announced to the application if they result in a value lower than the currently announced PMTU.
- \* Overall sequencing for optional re-assembly procedure
- \* Congestion control
- \* Robust MP-DCCP session establishment (no dependency on an initial path setup)

### 3.1. Multipath Capable Feature

DCCP endpoints are multipath-disabled by default and multipath capability can be negotiated with the Multipath Capable Feature.

Multipath Capable has feature number 10 and is server-priority. It takes one-byte values. The first four bits are used to specify compatible versions of the MP-DCCP implementation. The following four bits are reserved for further use.

### 3.2. Multipath Option

```

+-----+-----+-----+-----+-----+
|00101110| Length | MP_OPT | Value(s) ...
+-----+-----+-----+-----+-----+
Type=46

```

Type	Option Length	MP_OPT	Meaning
46	var	0 =MP_CONFIRM	Confirm reception and processing of an MP_OPT option
46	11	1 =MP_JOIN	Join path to an existing MP-DCCP flow
46	3	2 =MP_FAST_CLOSE	Close MP-DCCP flow
46	var	3 =MP_KEY	Exchange key material for MP_HMAC
46	7	4 =MP_SEQ	Multipath Sequence Number
46	23	5 =MP_HMAC	HMA Code for authentication
46	12	6 =MP_RTT	Transmit RTT values
46	var	7 =MP_ADDADDR	Advertise additional Address
46	var	8 =MP_REMOVEADDR	Remove Address
46	4	9 =MP_PRIO	Change Subflow Priority

Table 5: MP\_OPT Option Types

## 3.2.1. MP\_CONFIRM

```

+-----+-----+-----+-----+
|00101110| Length |00000000| List of options ...
+-----+-----+-----+-----+
Type=46      MP_OPT=0

```

MP\_CONFIRM can be used to send confirmation of received and processed options. Confirmed options are copied verbatim and appended as List of options. The length varies dependent on the amount of options.

[Tbd] Encoding "list of options"

## 3.2.2. MP\_JOIN



```

+-----+-----+-----+-----+-----+-----+
|00101110|00001011|00000001| Path Token |
+-----+-----+-----+-----+-----+-----+
| Nonce |
+-----+-----+-----+-----+
Type=46 Length=11 MP_OPT=1

```

The MP\_JOIN option is used to add a new path to an existing MP-DCCP flow. The Path Token is the SHA-1 HASH of the derived key (d-key), which was previously exchanged with the MP\_KEY option. MP\_HMAC MUST be set when using MP\_JOIN to provide authentication (See MP\_HMAC for details). Also MP\_KEY MUST be set to provide key material for authentication purposes.

### 3.2.3. MP\_FAST\_CLOSE

```

+-----+-----+-----+
|00101110|00000011|00000010|
+-----+-----+-----+
Type=46 Length=3 MP_OPT=2

```

MP\_FAST\_CLOSE terminates the MP-DCCP flow and all corresponding subflows.

### 3.2.4. MP\_KEY

```

+-----+-----+-----+-----+-----+-----+
|00101110| Length |00000011| Key Type | Key Data ...
+-----+-----+-----+-----+-----+-----+
Type=46 MP_OPT=3

```

The MP\_KEY suboption is used to exchange key material between hosts. The Length varies between 5 and 8 Bytes. The Key Type field is used to specify the key type. Key types are shown in Table 6.

Key Type	Key Length	Meaning
0 =Plain Text	8	Plain Text Key
1 =ECDHE-C25519-SHA256	32	ECDHE with SHA256 and Curve25519
2 =ECDHE-C25519-SHA512	32	ECDHE with SHA512 and Curve25519
3-255		Reserved

Table 6: MP\_KEY Key Types

#### Plain Text

Key Material is exchanged in plain text between hosts, and the key parts (key-a, key-b) are used by each host to generate the derived key (d-key) by concatenating the two parts with the local key in front (e.g. hostA d-key=(key-a+key-b), hostB d-key=(key-b+key-a)).

#### ECDHE-SHA256-C25519

Key Material is exchanged via ECDHE key exchange with SHA256 and Curve 25519 to generate the derived key (d-key).

#### ECDHE-SHA512-C25519

Key Material is exchanged via ECDHE key exchange with SHA512 and Curve 25519 to generate the derived key (d-key).

#### 3.2.5. MP\_SEQ

```

+-----+-----+-----+-----+
| 00101110 | 00000111 | 00000100 | Multipath Sequence Number |
+-----+-----+-----+-----+
Type=46 Length=7 MP_OPT=4

```

The MP\_SEQ option is used for end-to-end datagram-based sequence numbers of an MP-DCCP connection. The initial data sequence number (IDSN) SHOULD be set randomly. The MP\_SEQ number space is different from path individual sequence number space.

#### 3.2.6. MP\_HMAC

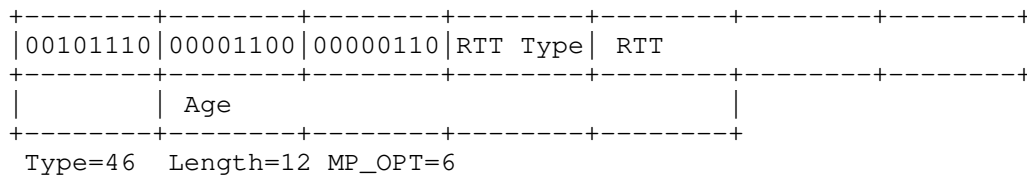
```

+-----+-----+-----+-----+
| 00101110 | 00001011 | 00000101 | HMAC-SHA1 (20 bytes) ... |
+-----+-----+-----+-----+
Type=46 Length=23 MP_OPT=5

```

The MP\_HMAC option is used to provide authentication for the MP\_JOIN option. The HMAC is built using the derived key (d-key) calculated previously from the handshake key material exchanged with the MP\_KEY option. The Message for the HMAC is the header of the MP\_JOIN for which authentication shall be performed. By including a nonce in these datagrams, possible replay-attacks are remedied.

### 3.2.7. MP\_RTT



The MP\_RTT option is used to transmit RTT values in milliseconds and MUST belong to the path over which this information is transmitted. Additionally, the age of the measurement is specified in milliseconds.

#### Raw RTT (=0)

Raw RTT value of the last Datagram Round-Trip. The Age parameter is set to the age of when the Ack for the datagram was received.

#### Min RTT (=1)

Min RTT value. The period for computing the Minimum can be specified by the Age parameter.

#### Max RTT (=2)

Max RTT value. The period for computing the Maximum can be specified by the Age parameter.

#### Smooth RTT (=3)

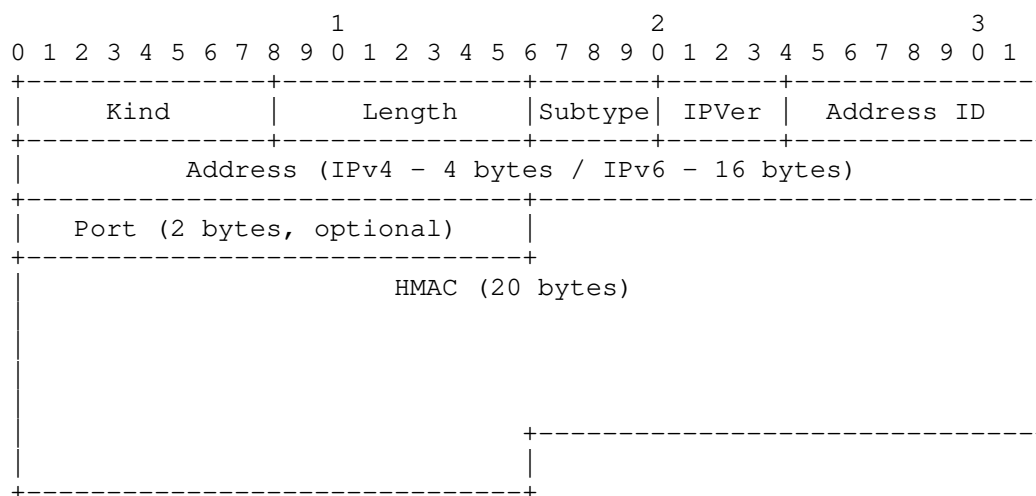
Averaged RTT value. The period for computing the smoothed RTT can be specified by the Age parameter.

#### Age (=4)

The Age parameter is a 4-byte value which is set to the age or timestamp when the Ack for the datagram was received in case of RTT type = 0 and may contain the periods for computing of derived RTT values depending on other RTT types, i.e., the Minimum (=1) and Maximum (=2) as well as the averaged smoothed RTT value (=3). [TBD/TBV]

## 3.2.8. MP\_ADDADDR

The MP\_ADDADDR option announces additional addresses (and, optionally, ports) on which a host can be reached. This option can be used at any time during an existing DCCP connection, when the sender wishes to enable multiple paths and/or when additional paths become available. Length is variable depending on IPv4 or IPv6 and whether port number is used and is in range between 28 and 42 bytes.



Every address has an Address ID that can be used for uniquely identifying the address within a connection for address removal. The Address ID is also used to identify MP\_JOIN options (see Section 3.2.2) relating to the same address, even when address translators are in use. The Address ID MUST uniquely identify the address for the sender of the option (within the scope of the connection); the mechanism for allocating such IDs is implementation specific.

All Address IDs learned via either MP\_JOIN or ADD\_ADDR SHOULD be stored by the receiver in a data structure that gathers all the Address-ID-to-address mappings for a connection (identified by a token pair). In this way, there is a stored mapping between the Address ID, observed source address, and token pair for future processing of control information for a connection.

Ideally, ADD\_ADDR and REMOVE\_ADDR options would be sent reliably, and in order, to the other end. This would ensure that this address management does not unnecessarily cause an outage in the connection when remove/add addresses are processed in reverse order, and also to ensure that all possible paths are used. Note, however, that losing

reliability and ordering will not break the multipath connections, it will just reduce the opportunity to open new paths and to survive different patterns of path failures.

Therefore, implementing reliability signals for these DCCP options is not necessary. In order to minimize the impact of the loss of these options, however, it is RECOMMENDED that a sender should send these options on all available subflows. If these options need to be received in order, an implementation SHOULD only send one ADD\_ADDR/REMOVE\_ADDR option per RTT, to minimize the risk of misordering. A host that receives an ADD\_ADDR but finds a connection set up to that IP address and port number is unsuccessful SHOULD NOT perform further connection attempts to this address/port combination for this connection. A sender that wants to trigger a new incoming connection attempt on a previously advertised address/port combination can therefore refresh ADD\_ADDR information by sending the option again.

[TBD/TBV]

### 3.2.9. MP\_REMOVEADDR

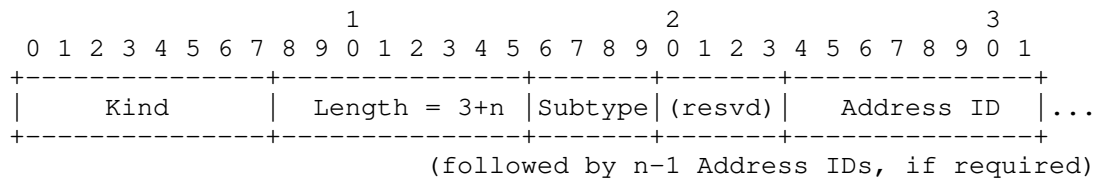
If, during the lifetime of an MP-DCCP connection, a previously announced address becomes invalid (e.g., if the interface disappears), the affected host SHOULD announce this so that the peer can remove subflows related to this address.

This is achieved through the Remove Address (REMOVE\_ADDR) option which will remove a previously added address (or list of addresses) from a connection and terminate any subflows currently using that address.

For security purposes, if a host receives a REMOVE\_ADDR option, it must ensure the affected path(s) are no longer in use before it instigates closure. Typical DCCP validity tests on the subflow (e.g., packet type specific sequence and acknowledgement number check) MUST also be undertaken. An implementation can use indications of these test failures as part of intrusion detection or error logging.

The sending and receipt of this message SHOULD trigger the sending of DCCP-Close and DCCP-Reset by client and server, respectively on the affected subflow(s) (if possible), as a courtesy to cleaning up middlebox state, before cleaning up any local state.

Address removal is undertaken by ID, so as to permit the use of NATs and other middleboxes that rewrite source addresses. If there is no address at the requested ID, the receiver will silently ignore the request.

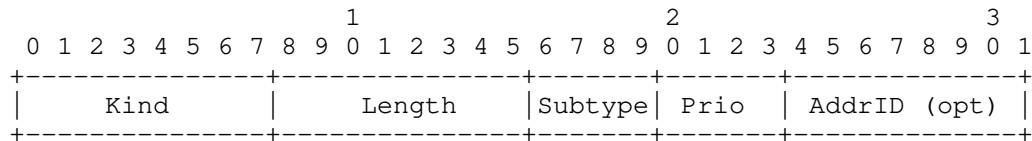


Minimum length of this option is 4 bytes (for one address to remove).

[TBD/TBV]

### 3.2.10. MP\_PRIO

In the event that a single specific path out of the set of available paths shall be treated with higher priority compared to the others, a host may wish to signal such change in priority of subflows to the peer. Therefore, the MP\_PRIO option, shown below, can be used to set a priority flag for the subflow on which it is sent.



Whether more than two values for priority (e.g., B for backup and P for prioritized path) are defined in case of more than two parallel paths is for further consideration.

[TBD/TBV]

### 3.3. MP-DCCP Handshaking Procedure

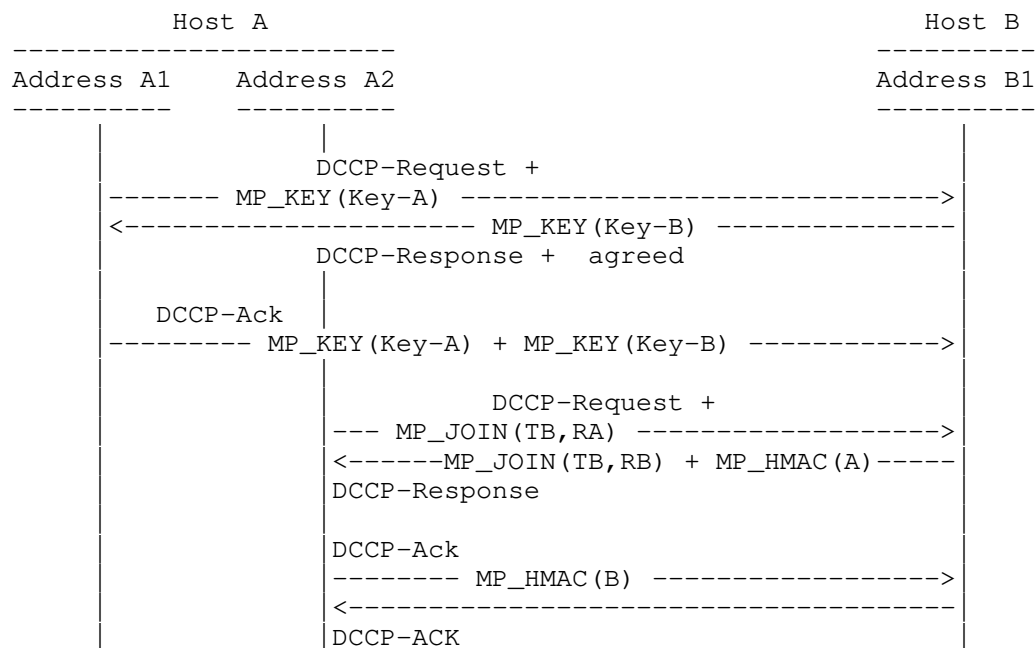


Figure 3: Example MP-DCCP Handshake

The basic initial handshake for the first flow is as follows:

- \* Host A sends a DCCP-Request with the MP-Capable feature Change request and the MP\_KEY option with Host-specific Key-A
- \* Host B sends a DCCP-Response with Confirm feature for MP-Capable and the MP\_Key option with Host-specific Key-B
- \* Host A sends a DCCP-Ack with both Keys echoed to Host B.

The handshake for subsequent flows based on a successful initial handshake is as follows:

- \* Host A sends a DCCP-Request with the MP-Capable feature Change request and the MP\_JOIN option with Host B's Token TB, generated from the derived key by applying a SHA-1 hash and truncating to the first 32 bits. Additionally, an own random nonce RA is transmitted with the MP\_JOIN.
- \* Host B computes the HMAC of the DCCP-Request and sends a DCCP-Response with Confirm feature option for MP-Capable and the MP\_JOIN option with the Token TB and a random nonce RB together with the computed MP\_HMAC.

- \* Host A sends a DCCP-Ack with the HMAC computed for the DCCP-Response.
- \* Host B sends a DCCP-Ack confirm the HMAC and to conclude the handshaking.

#### 4. Security Considerations

Similar to DCCP, MP-DCCP does not provide cryptographic security guarantees inherently. Thus, if applications need cryptographic security (integrity, authentication, confidentiality, access control, and anti-replay protection) the use of IPsec or some other kind of end-to-end security is recommended; Secure Real-time Transport Protocol (SRTP) [RFC3711] is one candidate protocol for authentication. Together with Encryption of Header Extensions in SRTP, as provided by [RFC6904], also integrity would be provided.

As described in [RFC4340], DCCP provides protection against hijacking and limits the potential impact of some denial-of-service attacks, but DCCP provides no inherent protection against attackers' snooping on data packets. Regarding the security of MP-DCCP no additional risks should be introduced compared to regular DCCP of today. Thereof derived are the following key security requirements to be fulfilled by MP-DCCP:

- \* Provide a mechanism to confirm that parties involved in a subflow handshake are identical to those in the original connection setup.
- \* Provide verification that the new address to be included in a MP connection is valid for a peer to receive traffic at before using it.
- \* Provide replay protection, i.e., ensure that a request to add/remove a subflow is 'fresh'.

In order to achieve these goals, MP-DCCP includes a hash-based handshake algorithm documented in Sections Section 3.2.4 and Section 3.3. The security of the MP-DCCP connection depends on the use of keys that are shared once at the start of the first subflow and are never sent again over the network. To ease demultiplexing while not giving away any cryptographic material, future subflows use a truncated cryptographic hash of this key as the connection identification "token". The keys are concatenated and used as keys for creating Hash-based Message Authentication Codes (HMACs) used on subflow setup, in order to verify that the parties in the handshake are the same as in the original connection setup. It also provides verification that the peer can receive traffic at this new address. Replay attacks would still be possible when only keys are used;



therefore, the handshakes use single-use random numbers (nonces) at both ends -- this ensures that the HMAC will never be the same on two handshakes. Guidance on generating random numbers suitable for use as keys is given in [RFC4086]. During normal operation, regular DCCP protection mechanisms (such as header checksum to protect DCCP headers against corruption) will provide the same level of protection against attacks on individual DCCP subflows as exists for regular DCCP today.

## 5. Interactions with Middleboxes

Issues from interaction with on-path middleboxes such as NATs, firewalls, proxies, intrusion detection systems (IDSs), and others have to be considered for all extensions to standard protocols since otherwise unexpected reactions of middleboxes may hinder its deployment. DCCP already provides means to mitigate the potential impact of middleboxes, also in comparison to TCP (see [RFC4043], sect. 16). In case, however, both hosts are located behind a NAT or firewall entity, specific measures have to be applied such as the [RFC5596]-specified simultaneous-open technique that update the (traditionally asymmetric) connection-establishment procedures for DCCP. Further standardized technologies addressing NAT type middleboxes are covered by [RFC5597].

[RFC6773] specifies UDP Encapsulation for NAT Traversal of DCCP sessions, similar to other UDP encapsulations such as for SCTP [RFC6951]. The alternative U-DCCP approach proposed in [I-D.amend-tsvwg-dccp-udp-header-conversion] would reduce tunneling overhead. The handshaking procedure for DCCP-UDP header conversion or use of a DCCP-UDP negotiation procedure to signal support for DCCP-UDP header conversion would require encapsulation during the handshakes and use of two additional port numbers out of the UDP port number space, but would require zero overhead afterwards.

## 6. Implementation

The approach described above has been implemented in open source across different testbeds and a new scheduling algorithm has been extensively tested. Also demonstrations of a laboratory setup have been executed and have been published at [website].

## 7. Acknowledgments

### 1. Notes

This document is inspired by Multipath TCP [RFC6824]/[RFC8684] and some text passages for the -00 version of the draft are copied almost unmodified.

## 8. IANA Considerations

This document defines one new value to DCCP feature list and one new DCCP Option with ten corresponding Subtypes as follows. This document defines a new DCCP feature parameter for negotiating the support of multipath capability for DCCP sessions between hosts as described in Section 3. The following entry in Table 7 should be added to the "Feature Numbers Registry" according to [RFC4340], Section 19.4. under the "DCCP Protocol" heading.

Value	Feature Name	Specification
0x10	MP-DCCP capability feature	Section 3.1

Table 7: Addition to DCCP Feature list Entries

This document defines a new DCCP protocol option of type=46 as described in Section 3.2 together with 10 additional sub-options. The following entries in Table 8 should be added to the "DCCP Protocol options" and assigned as "MP-DCCP sub-options", respectively.

Value	Symbol	Name	Reference
TBD or Type=46	MP_OPT	DCCP Multipath option	Section 3.2
TBD or MP_OPT=0	MP_CONFIRM	Confirm reception/ processing of an MP_OPT option	Section 3.2.1
TBD or MP_OPT=1	MP_JOIN	Join path to existing MP-DCCP flow	Section 3.2.2
TBD or MP_OPT=2	MP_FAST_CLOSE	Close MP-DCCP flow	Section 3.2.3
TBD or MP_OPT=3	MP_KEY	Exchange key material for MP_HMAC	Section 3.2.4
TBD or MP_OPT=4	MP_SEQ	Multipath Sequence Number	Section 3.2.5
TBD or MP_OPT=5	MP_HMAC	Hash-based Message Auth. Code for MP- DCCP	Section 3.2.6
TBD or MP_OPT=6	MP_RTT	Transmit RTT values and calculation parameters	Section 3.2.7
TBD or MP_OPT=7	MP_ADDADDR	Advertise additional Address(es)/Port(s)	Section 3.2.8
TBD or MP_OPT=8	MP_REMOVEADDR	Remove Address(es)/ Port(s)	Section 3.2.9
TBD or MP_OPT=9	MP_PRIO	Change Subflow Priority	Section 3.2.10

Table 8: Addition to DCCP Protocol options and  
corresponding sub-options

[Tbd], must include options for:

- \* handshaking procedure to indicate MP support
- \* handshaking procedure to indicate JOINING of an existing MP connection
- \* signaling of new or changed addresses
- \* setting handover or aggregation mode
- \* setting reordering on/off

should include options carrying:

- \* overall sequence number for restoring purposes
- \* sender time measurements for restoring purposes
- \* scheduler preferences
- \* reordering preferences

## 9. Informative References

[I-D.amend-tsvwg-dccp-udp-header-conversion]

Amend, M., Brunstrom, A., Kassler, A., and V. Rakocevic, "Lossless and overhead free DCCP - UDP header conversion (U-DCCP)", Work in Progress, Internet-Draft, draft-amend-tsvwg-dccp-udp-header-conversion-01, 8 July 2019, <<https://www.ietf.org/archive/id/draft-amend-tsvwg-dccp-udp-header-conversion-01.txt>>.

[I-D.amend-tsvwg-multipath-framework-mpdccp]

Amend, M., Bogenfeld, E., Brunstrom, A., Kassler, A., and V. Rakocevic, "A multipath framework for UDP traffic over heterogeneous access networks", Work in Progress, Internet-Draft, draft-amend-tsvwg-multipath-framework-mpdccp-01, 8 July 2019, <<https://www.ietf.org/archive/id/draft-amend-tsvwg-multipath-framework-mpdccp-01.txt>>.

[I-D.lhwxz-hybrid-access-network-architecture]

Leymann, N., Heidemann, C., Wesserman, M., Xue, L., and M. Zhang, "Hybrid Access Network Architecture", Work in Progress, Internet-Draft, draft-lhwxz-hybrid-access-network-architecture-02, 13 January 2015, <<https://www.ietf.org/archive/id/draft-lhwxz-hybrid-access-network-architecture-02.txt>>.

- [I-D.muley-network-based-bonding-hybrid-access]  
Muley, P., Henderickx, W., Liang, G., Liu, H., Cardullo, L., Newton, J., Seo, S., Draznin, S., and B. Patil, "Network based Bonding solution for Hybrid Access", Work in Progress, Internet-Draft, draft-muley-network-based-bonding-hybrid-access-03, 22 October 2018, <<https://www.ietf.org/archive/id/draft-muley-network-based-bonding-hybrid-access-03.txt>>.
- [paper] Amend, M., Bogenfeld, E., Cvjetkovic, M., Rakocevic, V., Pieska, M., Kassler, A., and A. Brunstrom, "A Framework for Multiaccess Support for Unreliable Internet Traffic using Multipath DCCP", DOI 10.1109/LCN44214.2019.8990746, October 2019, <<https://doi.org/10.1109/LCN44214.2019.8990746>>.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [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>>.
- [RFC3124] Balakrishnan, H. and S. Seshan, "The Congestion Manager", RFC 3124, DOI 10.17487/RFC3124, June 2001, <<https://www.rfc-editor.org/info/rfc3124>>.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <<https://www.rfc-editor.org/info/rfc3711>>.
- [RFC4043] Pinkas, D. and T. Gindin, "Internet X.509 Public Key Infrastructure Permanent Identifier", RFC 4043, DOI 10.17487/RFC4043, May 2005, <<https://www.rfc-editor.org/info/rfc4043>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [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>>.

- [RFC5595] Fairhurst, G., "The Datagram Congestion Control Protocol (DCCP) Service Codes", RFC 5595, DOI 10.17487/RFC5595, September 2009, <<https://www.rfc-editor.org/info/rfc5595>>.
- [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>>.
- [RFC5634] Fairhurst, G. and A. Sathiaselalan, "Quick-Start for the Datagram Congestion Control Protocol (DCCP)", RFC 5634, DOI 10.17487/RFC5634, August 2009, <<https://www.rfc-editor.org/info/rfc5634>>.
- [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>>.
- [RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <<https://www.rfc-editor.org/info/rfc6824>>.
- [RFC6904] Lennox, J., "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904, DOI 10.17487/RFC6904, April 2013, <<https://www.rfc-editor.org/info/rfc6904>>.
- [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>>.
- [RFC8684] Ford, A., Raiciu, C., Handley, M., Bonaventure, O., and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 8684, DOI 10.17487/RFC8684, March 2020, <<https://www.rfc-editor.org/info/rfc8684>>.

- [slide] Amend, M., "MP-DCCP for enabling transfer of UDP/IP traffic over multiple data paths in multi-connectivity networks", IETF105 , n.d.,  
<<https://datatracker.ietf.org/meeting/105/materials/slides-105-tsvwg-sessa-62-dccp-extensions-for-multipath-operation-00>>.
- [TS23.501] 3GPP, "System architecture for the 5G System; Stage 2; Release 16", December 2020,  
<[https://www.3gpp.org/ftp//Specs/archive/23\\_series/23.501/23501-g70.zip](https://www.3gpp.org/ftp//Specs/archive/23_series/23.501/23501-g70.zip)>.
- [website] "Multipath extension for DCCP", n.d.,  
<<https://multipath-dccp.org/>>.

## Authors' Addresses

Markus Amend  
Deutsche Telekom  
Deutsche-Telekom-Allee 9  
64295 Darmstadt  
Germany

Email: [Markus.Amend@telekom.de](mailto:Markus.Amend@telekom.de)

Dirk von Hugo  
Deutsche Telekom  
Deutsche-Telekom-Allee 9  
64295 Darmstadt  
Germany

Email: [Dirk.von-Hugo@telekom.de](mailto:Dirk.von-Hugo@telekom.de)

Anna Brunstrom  
Karlstad University  
Universitetsgatan 2  
SE-651 88 Karlstad  
Sweden

Email: [anna.brunstrom@kau.se](mailto:anna.brunstrom@kau.se)

Andreas Kassler  
Karlstad University  
Universitetsgatan 2  
SE-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

Stephen Johnson  
BT  
Adastral Park  
Martlesham Heath  
IP5 3RE  
United Kingdom

Email: stephen.h.johnson@bt.com



Transport Area Working Group  
Internet-Draft  
Intended status: Informational  
Expires: January 9, 2020

M. Amend  
E. Bogenfeld  
Deutsche Telekom  
A. Brunstrom  
A. Kassler  
Karlstad University  
V. Rakocevic  
City University of London  
July 08, 2019

A multipath framework for UDP traffic over heterogeneous access networks  
draft-amend-tsvwg-multipath-framework-mpdccp-01

## Abstract

More and more of today's devices are multi-homing capable, in particular 3GPP user equipment like smartphones. In the current standardization of the next upcoming mobile network generation 5G Rel.16, this is especially targeted in the study group Access Traffic Steering Switching Splitting [TR23.793]. ATSSS describes the flexible selection or combination of 3GPP untrusted access like Wi-Fi and cellular access, overcoming the single-access limitation of today's devices and services. Another multi-connectivity scenario is the Hybrid Access [I-D.lhwxyz-hybrid-access-network-architecture][I-D.muley-network-based-bonding-hybrid-access], providing multiple access for CPEs, which extends the traditional way of single access connectivity at home to dual-connectivity over 3GPP and fixed access. A missing piece in the ATSSS and Hybrid Access is the access and path measurement, which is required for efficient and beneficial traffic steering decisions. This becomes particularly important in heterogeneous access networks with a multitude of volatile access paths. While MP-TCP has been proposed to be used within ATSSS, there are drawbacks when being used to encapsulate unreliable traffic as it blindly retransmits each lost frame leading to excessive delay and potential head-of-line blocking. A decision for MP-TCP though leaves the increasing share of UDP in today's traffic mix (<<https://arxiv.org/abs/1801.05168>>) unconsidered. In this document, a multi-access framework is proposed leveraging the MP-DCCP network protocol, which enables flexible traffic steering, switching and splitting also for unreliable traffic. A benefit is the support for pluggable congestion control which enables our framework to be used either independent or complementary to MP-TCP.

## 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 January 9, 2020.

## Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Requirements . . . . .	3
3. IP compatible multipath framework based on MP-DCCP . . . . .	4
4. Application and placement . . . . .	6
5. Conclusion . . . . .	7
6. Security Considerations . . . . .	8
7. Acknowledgments . . . . .	8
8. Informative References . . . . .	8
Authors' Addresses . . . . .	9

## 1. Introduction

Multi-connectivity access networks are evolving. Ongoing standardization at 3GPP for 5G mobile networks [TR23.793] or the so called Hybrid Access networks [I-D.lhwxyz-hybrid-access-network-architecture][I-D.muley-network-based-bonding-hybrid-access] already provides or will enable in the near future the possibility to use multi-connectivity for a very large number of mobile users. Multi-connectivity solutions come with many user benefits including superior resilience against network outages, higher capacities for user traffic and network cost optimizations. Since multi-connectivity architectures are almost mature, new network protocols are required to fully exploit multi-connectivity and maximise its potential. In the simplest case, multi-connectivity is used for load-balancing decisions in order to balance the user flows over multiple paths. However, this has no effect on resilience or capacity gain on those load balanced individual flows. More complex scenarios include the dynamic shifting of traffic flows seamlessly between multiple paths or even aggregating those paths for leveraging the available capacity of multiple individual paths. Like [TR23.793] this document refers to the three distribution schemes Steering (load balancing), Switching (seamless handover) and Splitting (capacity aggregation).

MP-TCP [RFC6824] is a protocol, which can be applied in the above mentioned use cases and supports load-balancing, traffic shifting among the multiple paths and also capacity aggregation. Further, it leverages the inherent congestion control from TCP which adapts the sending rate by observing congestion signals from the network. By design, MP-TCP is limited to TCP services as it blindly re-transmits lost packets. Consequently, when MP-TCP is used as a framework for ATSSS, it may re-transmit packets sent from unreliable services such as e.g. UDP unnecessary. This may lead to head-of-line blocking and increased latency, which is detrimental to real-time services. As future multi-connectivity systems must support latency sensitive traffic that might be transported over unreliable transport, it is not sufficient anymore to rely on supporting only TCP. The increasing share of UDP traffic, mainly impacted by the QUIC introduction, may significantly reduce the share from TCP. It might be expected that with HTTP/3 carried over QUIC [I-D.ietf-quic-http], the previous strong dominance of TCP will be challenged by UDP.

## 2. Requirements

A multiaccess framework shall meet the following requirements:

- o IP compatibility: A multiaccess framework shall be able to transport IP packets and not make any assumptions on which transport protocol is encapsulated.
- o Support for unreliable traffic: A multiaccess framework should provide support for transporting unreliable traffic, such as QUIC or UDP based flows. Therefore, unreliable transmission should be supported.
- o Support for flexible re-ordering: A multiaccess framework should support flexible re-ordering of user traffic, including no re-ordering at all. This requirement is important to support low latency traffic, where the re-creation of packet order may negatively impact delivery latency.
- o Support for flexible congestion control: A multiaccess framework should support flexible congestion control, including the disabling of the congestion control, if the inner traffic is known to be congestion controlled.
- o Support for flexible packet scheduling: A multiaccess framework should support different packet scheduling mechanisms, which should be configurable from the control plane. Examples are cheapest path first, or other more sophisticated schedulers.
- o Lightweight: A multiaccess framework should be lightweight in computational resources and limit the encapsulation overhead.

To use QUIC as part of a multiaccess framework, by for example providing multipath support for QUIC, it could be beneficial if unreliable transmission is supported as well as being able to influence or disable QUIC's congestion control. In addition, it would be beneficial if the encryption of QUIC can be disabled. This is because for ATSSS, it is foreseen that the underlying tunnel from the mobile over public WLANs is based on IPsec.

### 3. IP compatible multipath framework based on MP-DCCP

We propose a new multiaccess framework, which overcomes MP-TCP's restriction to TCP services and provides IP compatibility in Figure 1. The framework employs MP-DCCP [I-D.amend-tsvwg-multipath-dccp] in combination with an encapsulation scheme. For simplification, Figure 1 assumes a traffic direction from the left (sender) to the right (receiver) and requires application in each direction for bi-directional transmission. The framework in Figure 1 can replace or complement MP-TCP to reach IP compatibility.

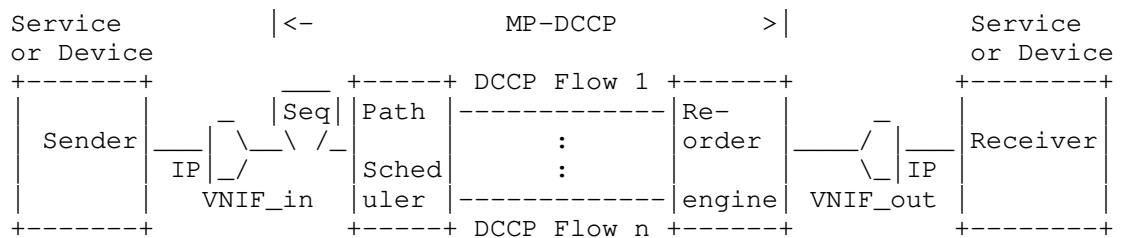


Figure 1: IP compatible multipath framework based on MP-DCCP

PDUs generated from the sender and travelling through the framework in Figure 1 pass the components in the following order:

1. Sender: Generates any PDU based on the IP protocol.
2. VNIF\_in: IP based Virtual Network Interface as entry point to the multipath framework. A simple routing logic in front (between (1) and (2)) can act as gatekeeper and decides upon redirecting traffic through the VNIF or bypassing it. The VNIF adds an extra IP header to reach the multi-connectivity termination point.
3. Seq: Sequencing of the PDUs passed through (2) depending on the incoming order. Adds an incrementing number, which is later added to the DCCP encapsulation in (4).
4. Path Scheduler: Decision logic for scheduling sequenced PDUs over the individual connected DCCP flows for multipath transmission. The path scheduler can use the information from the DCCP flows (see (5)) inherent congestion control information like CWND, packet loss, RTT, Jitter, etc.. After selection of a DCCP flow, the PDU is encapsulated into the individual flow. Further information, at least the sequencing, is added on top as DCCP option.
5. DCCP Flow(s): Responsible to transmit the encapsulated PDUs to the MP-DCCP exit point.
6. Reorder engine: Depending on the sequencing information of (3), a re-assembly of the PDU stream can be applied. Different re-order algorithms should be supported in a configurable way, including no re-ordering.
7. VNIF\_out: Releases PDUs that have passed the re-ordering engine and strips the DCCP specific overhead. Again, routing is responsible to deliver the PDUs to the receiver based on the destination information in the PDU.

8. Receiver: Receive the PDU as generated in (1).

The simple enclosing of the MP-DCCP with Virtual Network Interface (VNIF) provides the IP compatibility. However, a service or protocol classifier between sender and VNIF can reduce the scope to particular traffic, e.g. UDP, by simple routing decisions. The MP-DCCP takes over responsibility for the multi-path transfer of the traffic, which is directed through the VNIF\_in. For possible re-assembly operations, the IP packets may be stamped with a continuously incremented sequence number. This is not mandatory, but assumed required in most seamless handover and capacity aggregation use cases. The path scheduler decides for each IP packet, which DCCP flow it should use for encapsulation, based on a configurable decision logic and supported by the congestion control information of the DCCP flows available for transmission. A DCCP flow selection for a PDU leads to its encapsulation into the respective DCCP flow and adding extra information required for the multipath transmission, e.g. the sequence number. Encapsulation also means, that a DCCP and IP header is added to the original PDU to reach the multi-connectivity end-point. When the encapsulated PDUs arrive at the multi-path termination point, they are re-ordered depending on the carried sequence number and a configurable logic. The re-ordering engine may also include a logic in which packets are just forwarded (no re-ordering). Re-ordering needs to be considered carefully since any active intervention changes the latency responsiveness. The multi-path termination is finally completed when the DCCP overhead is stripped and the PDU leaves VNIF\_out. Further routing depends again on the IP layer of the original PDU.

#### 4. Application and placement

The framework of Figure 1 is very flexible in applying multipath support in different architectures and allows MP-DCCP to be applied at any place between sender and receiver. Figure 2 to Figure 5 provide several architectural options for the deployment of the framework.

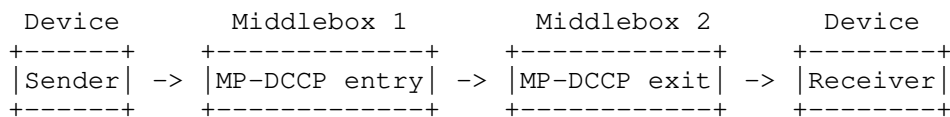


Figure 2: Sender and receiver independent MP-DCCP

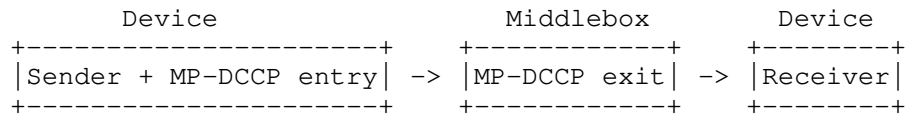


Figure 3: Sender integrated but receiver independent MP-DCCP

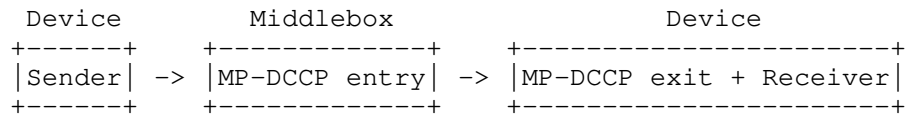


Figure 4: Sender independent and receiver integrated MP-DCCP

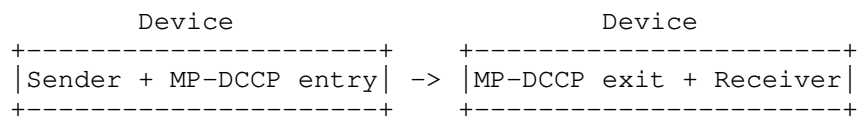


Figure 5: Sender and receiver integrated MP-DCCP

## 5. Conclusion

The specified IP compatible multipath framework based on MP-DCCP in this document comprises several benefits:

- o Pure routing
- o Inherent path estimation and measurement
- o Imposes no constraints on reliability or in-order delivery of application PDUs
- o Modular re-ordering
- o Modular scheduling
- o IP compatible
- o Based on the standardized DCCP.

Middle-box traversing, when the framework is applied in uncontrolled environments, is addressed in [RFC6733] and [I-D.amend-tsvwg-dccp-udp-header-conversion].

## 6. Security Considerations

[Tbd]

## 7. Acknowledgments

## 8. Informative References

- [I-D.amend-tsvwg-dccp-udp-header-conversion]  
Amend, M., Brunstrom, A., Kassler, A., and V. Rakocevic,  
"Lossless and overhead free DCCP - UDP header conversion  
(U-DCCP)", draft-amend-tsvwg-dccp-udp-header-conversion-01  
(work in progress), July 2019.
- [I-D.amend-tsvwg-multipath-dccp]  
Amend, M., Brunstrom, A., Kassler, A., and V. Rakocevic,  
"DCCP Extensions for Multipath Operation with Multiple  
Addresses", draft-amend-tsvwg-multipath-dccp-01 (work in  
progress), March 2019.
- [I-D.ietf-quic-http]  
Bishop, M., "Hypertext Transfer Protocol Version 3  
(HTTP/3)", draft-ietf-quic-http-18 (work in progress),  
January 2019.
- [I-D.lhwxyz-hybrid-access-network-architecture]  
Leymann, N., Heidemann, C., Wasserman, M., Xue, L., and M.  
Zhang, "Hybrid Access Network Architecture", draft-lhwxyz-  
hybrid-access-network-architecture-02 (work in progress),  
January 2015.
- [I-D.muley-network-based-bonding-hybrid-access]  
Muley, P., Henderickx, W., Geng, L., Liu, H., Cardullo,  
L., Newton, J., Seo, S., Draznin, S., and B. Patil,  
"Network based Bonding solution for Hybrid Access", draft-  
muley-network-based-bonding-hybrid-access-03 (work in  
progress), October 2018.
- [RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn,  
Ed., "Diameter Base Protocol", RFC 6733,  
DOI 10.17487/RFC6733, October 2012,  
<<https://www.rfc-editor.org/info/rfc6733>>.
- [RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure,  
"TCP Extensions for Multipath Operation with Multiple  
Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013,  
<<https://www.rfc-editor.org/info/rfc6824>>.



[TR23.793]

3GPP, "Study on access traffic steering, switch and splitting support in the 5G System (5GS) architecture", December 2018.

Authors' Addresses

Markus Amend  
Deutsche Telekom  
Deutsche-Telekom-Allee 7  
64295 Darmstadt  
Germany

Email: Markus.Amend@telekom.de

Eckard Bogenfeld  
Deutsche Telekom  
Deutsche-Telekom-Allee 7  
64295 Darmstadt  
Germany

Email: Eckard.Bogenfeld@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

Independent Stream  
Internet-Draft  
Intended status: Informational  
Expires: November 2, 2020

F. Dold  
Taler Systems SA  
C. Grothoff  
BFH  
May 01, 2020

The 'payto' URI scheme for payments  
draft-dold-payto-14

## Abstract

This document defines the 'payto' Uniform Resource Identifier (URI) scheme for designating targets for payments.

A unified URI scheme for all payment target types allows applications to offer user interactions with URIs that represent payment targets, simplifying the introduction of new payment systems and applications.

## 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 November 2, 2020.

## Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
1.1. Objective . . . . .	2
1.2. Requirements Language . . . . .	3
2. Syntax of a 'payto' URI . . . . .	3
3. Semantics . . . . .	3
4. Examples . . . . .	4
5. Generic Options . . . . .	4
6. Internationalization and Character Encoding . . . . .	5
7. Tracking Payment Target Types . . . . .	5
7.1. ACH Bank Account . . . . .	6
7.2. Business Identifier Code . . . . .	7
7.3. International Bank Account Number . . . . .	7
7.4. Unified Payments Interface . . . . .	8
7.5. Bitcoin Address . . . . .	8
7.6. Interledger Protocol Address . . . . .	8
7.7. Void Payment Target . . . . .	9
8. Security Considerations . . . . .	9
9. IANA Considerations . . . . .	9
9.1. URI Scheme Registration . . . . .	10
10. Payment Target Types . . . . .	10
11. References . . . . .	10
11.1. Normative References . . . . .	10
11.2. Informational References . . . . .	11
Authors' Addresses . . . . .	12

## 1. Introduction

This document defines the 'payto' Uniform Resource Identifier (URI) [RFC3986] scheme for designating transfer form data for payments.

### 1.1. Objective

A 'payto' URI always identifies the target of a payment. A 'payto' URI consists of a payment target type, a target identifier and optional parameters such as an amount or a payment reference.

The interpretation of the target identifier is defined by the payment target type, and typically represents either a bank account or an (unsettled) transaction.

A unified URI scheme for all payment target types allows applications to offer user interactions with URIs that represent payment targets, simplifying the introduction of new payment systems and applications.

## 1.2. Requirements Language

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.

## 2. Syntax of a 'payto' URI

This document uses the Augmented Backus-Naur Form (ABNF) of [RFC5234].

```
payto-URI = "payto://" authority path-abempty [ "?" opts ]
opts = opt *( "&" opt )
opt-name = generic-opt / authority-specific-opt
opt-value = *pchar
opt = opt-name "=" opt-value
generic-opt = "amount" / "receiver-name" / "sender-name" /
             "message" / "instruction"
authority-specific-opt = ALPHA *( ALPHA / DIGIT / "-" / "." )
authority = ALPHA *( ALPHA / DIGIT / "-" / "." )
```

'path-abempty' is defined in [RFC3986] in Section 3.3. 'pchar' is defined in [RFC3986], Appendix A.

## 3. Semantics

The authority component of a payment URI identifies the payment target type. The payment target types are defined in the "Payment Target Types" sub-registry, see Section 10. The path component of the URI identifies the target for a payment as interpreted by the respective payment target type. The query component of the URI can provide additional parameters for a payment. Every payment target type SHOULD accept the options defined in generic-opt. The default operation of applications that invoke a URI with the payto scheme MUST be to launch an application (if available) associated with the payment target type that can initiate a payment. If multiple handlers are registered for the same payment target type, the user SHOULD be able to choose which application to launch. This allows users with multiple bank accounts (each accessed the respective bank's banking application) to choose which account to pay with. An application SHOULD allow dereferencing a payto URI even if the payment target type of that URI is not registered in the "Payment Target Types" sub-registry. Details of the payment MUST be taken from the path and options given in the URI. The user SHOULD be allowed to modify these details before confirming a payment.

#### 4. Examples

```
payto://iban/DE75512108001245126199?amount=EUR:200.0&message=hello
```

```
INVALID (authority missing): payto:iban/12345
```

#### 5. Generic Options

Applications MUST accept URIs with options in any order. The "amount" option MUST NOT occur more than once. Other options MAY be allowed multiple times, with further restrictions depending on the payment target type. The following options SHOULD be understood by every payment target type.

amount: The amount to transfer. The format MUST be:

```
amount = currency ":" unit [ "." fraction ]
currency = 1*ALPHA
unit = 1*(DIGIT / ",")
fraction = 1*(DIGIT / ",")
```

If a 3-letter 'currency' is used, it MUST be an [ISO4217] alphabetic code. A payment target type MAY define semantics beyond ISO 4217 for currency codes that are not 3 characters. The 'unit' value MUST be smaller than  $2^{53}$ . If present, the 'fraction' MUST consist of no more than 8 decimal digits. The use of commas is optional for readability and they MUST be ignored.

receiver-name: Name of the entity that receives the payment (creditor). The value of this option MAY be subject to lossy conversion, modification and truncation (for example, due to line wrapping or character set conversion).

sender-name: Name of the entity that makes the payment (debtor). The value of this option MAY be subject to lossy conversion, modification and truncation (for example, due to line wrapping or character set conversion).

message: A short message to identify the purpose of the payment. The value of this option MAY be subject to lossy conversion, modification and truncation (for example, due to line wrapping or character set conversion).

instruction: A short message giving payment reconciliation instructions to the recipient. An instruction that follows the

character set and length limitation defined by the respective payment target type SHOULD NOT be subject to lossy conversion.

## 6. Internationalization and Character Encoding

Various payment systems use restricted character sets. An application that processes 'payto' URIs MUST convert characters that are not allowed by the respective payment systems into allowable character using either an encoding or a replacement table. This conversion process MAY be lossy, except for the instruction field. If the value of the instruction field would be subject to lossy conversion, modification or truncation, the application SHOULD refuse further processing of the payment until a different value for the instruction is provided.

To avoid special encoding rules for the payment target identifier, the userinfo component [RFC3986] is disallowed in payto URIs. Instead, the payment target identifier is given as an option, where encoding rules are uniform for all options.

Defining a generic way of tagging the language of option fields containing natural language text (such as "receiver-name", "sender-name" and "message") is out of the scope of this document, as internationalization must accomodate the restrictions and requirements of the underlying banking system of the payment target type. The internationalization concerns SHOULD be individually defined by each payment target type.

## 7. Tracking Payment Target Types

A registry of Payment Target Types is described in Section 10. The registration policy for this registry is "First Come First Served", as described in [RFC8126]. When requesting new entries, careful consideration of the following criteria is strongly advised:

1. The description clearly defines the syntax and semantics of the payment target and optional parameters if applicable.
2. Relevant references are provided if they are available.
3. The chosen name is appropriate for the payment target type, does not conflict with well-known payment systems, and avoids potential to confuse users.
4. The payment system underlying the payment target type is not fundamentally incompatible with the general options (such as positive decimal amounts) in this specification.

5. The payment target type is not a vendor-specific version of a payment target type that could be described more generally by a vendor-neutral payment target type.
6. The specification of the new payment target type remains within the scope of payment transfer form data. In particular specifying complete invoices is not in scope. Neither are processing instructions to the payment processor or bank beyond a simple payment.
7. The payment target and the options do not contain the payment sender's account details.

Documents that support requests for new registry entries should provide the following information for each entry:

- o Name: The name of the payment target type (case insensitive ASCII string, restricted to alphanumeric characters, dots and dashes)
- o Description: A description of the payment target type, including the semantics of the path in the URI if applicable.
- o Example: At least one example URI to illustrate the payment target type.
- o Contact: The contact information of a person to contact for further information
- o References: Optionally, references describing the payment target type (such as an RFC) and target-specific options, or references describing the payment system underlying the payment target type.

This document populates the registry with six entries as follows (see also Section 10).

#### 7.1. ACH Bank Account

- o Name: ach
- o Description: Automated Clearing House. The path consist of two components, the routing number and the account number. Limitations on the length and character set of option values are defined by the implementation of the handler. Language tagging and internationalization of options is not supported.
- o Example: payto://ach/122000661/1234
- o Contact: N/A



- o References: [NACHA], [this.I-D]

## 7.2. Business Identifier Code

- o Name: bic
- o Description: Business Identifier Code. The path consist of just a BIC. This is used for wire transfers between banks. The registry for BICs is provided by SWIFT. The path does not allow specifying a bank account number. Limitations on the length and character set of option values are defined by the implementation of the handler. Language tagging and internationalization of options is not supported.
- o Example: `payto://bic/SOGEDEFFXXX`
- o Contact: N/A
- o References: [BIC], [this.I-D]

## 7.3. International Bank Account Number

- o Name: iban
- o Description: International Bank Account Number (IBAN). Generally the IBAN allows to unambiguously derive the the associated Business Identifier Code (BIC). However, some legacy applications process payments to the same IBAN differently based on the specified BIC. Thus the path can either consist of a single component (the IBAN) or two components (BIC followed by IBAN). The "message" option of the payto URI corresponds to the "unstructured remittance information" of SEPA credit transfers and is thus limited to 140 characters with character set limitations that differ according to the countries of banks and payment processors involved in the payment. The "instruction" option of the payto URI corresponds to the "end to end identifier" of SEPA credit transfers and is thus limited to at most 35 characters that can be alphanumeric or from the allowed set of special characters "+?/:-:().,'". Language tagging and internationalization of options is not supported.
- o Example: `payto://iban/DE75512108001245126199`  
`payto://iban/SOGEDEFFXXX/DE75512108001245126199`
- o Contact: N/A
- o References: [ISO20022], [this.I-D]

#### 7.4. Unified Payments Interface

- o Name: upi
- o Description: Unified Payment Interface. The path is an account alias. The amount and receiver-name options are mandatory for this payment target. Limitations on the length and character set of option values are defined by the implementation of the handler. Language tags and internationalization of options are not supported.
- o Example: `payto://upi/alice@example.com?receiver-name=Alice&amount=INR:200`
- o Contact: N/A
- o References: [UPILinking], [this.I-D]

#### 7.5. Bitcoin Address

- o Name: bitcoin
- o Description: Bitcoin protocol. The path is a "bitcoinaddress" as per [BIP0021]. Limitations on the length and character set of option values are defined by the implementation of the handler. Language tags and internationalization of options are not supported.
- o Example: `payto://bitcoin/12A1MyfXbW6RhdRAZEqofac5jCQQjwEPBu`
- o Contact: N/A
- o References: [BIP0021], [this.I-D]

#### 7.6. Interledger Protocol Address

- o Name: ilp
- o Description: Interledger protocol. The path is an ILP address as per [ILP-ADDR]. Limitations on the length and character set of option values are defined by the implementation of the handler. Language tagging and internationalization of options is not supported.
- o Example: `payto://ilp/g.acme.bob`
- o Contact: N/A

- o References: [ILP-ADDR], [this.I-D]

#### 7.7. Void Payment Target

- o Name: void
- o Description: The "void" payment target type allows specifying the parameters of an out-of-band payment (such as cash or other types of in-person transactions). The path is optional and interpreted as a comment. Limitations on the length and character set of option values are defined by the implementation of the handler. Language tags and internationalization of options are not supported.
- o Example: `payto://void/?amount=EUR:10.5`
- o Contact: N/A
- o References: [this.I-D]

#### 8. Security Considerations

Interactive applications handling the payto URI scheme MUST NOT initiate any financial transactions without prior review and confirmation from the user, and MUST take measures to prevent clickjacking [HMMW12].

Unless a payto URI is received over a trusted, authenticated channel, a user might not be able to identify the target of a payment. In particular due to homographs [unicode-tr36], a payment target type SHOULD NOT use human-readable names in combination with unicode in the target account specification, as it could give the user the illusion of being able to identify the target account from the URI.

The authentication/authorization mechanisms and transport security services used to process a payment encoded in a payto URI are handled by the application and are not in scope of this document.

To avoid unnecessary data collection, payment target types SHOULD NOT include personally identifying information about the sender of a payment that is not essential for an application to conduct a payment.

#### 9. IANA Considerations

IANA maintains a registry called the "Uniform Resource Identifier (URI) Schemes" registry.

### 9.1. URI Scheme Registration

IANA maintains the "Uniform Resource Identifier (URI) Schemes" registry that contains an entry for the 'payto' URI scheme. IANA is requested to update that entry to reference this document when published as an RFC.

## 10. Payment Target Types

This document specifies a list of Payment Target Types. It is possible that future work will need to specify additional payment target types. The GNUnet Assigned Numbers Authority (GANA) [GANA] operates the "payto-payment-target-types" registry to track the following information for each payment target type:

- o Name: The name of the payment target type (case insensitive ASCII string, restricted to alphanumeric characters, dots and dashes)
- o Contact: The contact information of a person to contact for further information
- o References: Optionally, references describing the payment target type (such as an RFC) and target-specific options, or references describing the payment system underlying the payment target type.

The entries that have been made for the "payto-payment-target-types" defined in this document are as follows:

Name	Contact	Reference
ach	N/A	[This.I-D]
bic	N/A	[This.I-D]
iban	N/A	[This.I-D]
upi	N/A	[This.I-D]
bitcoin	N/A	[This.I-D]
ilp	N/A	[This.I-D]
void	N/A	[This.I-D]

## 11. References

### 11.1. Normative References

[ISO20022]

International Organization for Standardization, "ISO 20022 Financial Services - Universal financial industry message scheme", May 2013.

- [ISO4217] International Organization for Standardization, "ISO 4217 Currency Codes", August 2018.
- [NACHA] NACHA, "NACHA Operating Rules & Guidelines", January 2017.
- [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>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [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>>.
- [unicode-tr36] Davis, M., Ed. and M. Suignard, "Unicode Technical Report #36: Unicode Security Considerations", September 2014.

## 11.2. Informational References

- [BIC] International Organization for Standardization, "ISO 9362:2014 Business Identifier Code (BIC)", March 2019, <<https://www.iso.org/standard/60390.html>>.
- [BIP0021] Schneider, N. and M. Corallo, "Bitcoin Improvement Proposal 21", January 2012, <[https://en.bitcoin.it/wiki/BIP\\_0021](https://en.bitcoin.it/wiki/BIP_0021)>.
- [GANA] GNUnet e.V., "GNUnet Assigned Numbers Authority (GANA)", April 2020, <<https://gana.gnunet.org/>>.

- [HMW12] Huang, L., Moshchuk, A., Wang, H., Schecter, S., and C. Jackson, "Clickjacking: Attacks and Defenses", January 2012, <<https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final39.pdf>>.
- [ILP-ADDR] Interledger Team, "ILP Addresses - v2.0.0", September 2018, <<https://interledger.org/rfcs/0015-ilp-addresses/>>.
- [UPILinking] National Payment Corporation of India, "Unified Payment Interface - Common URL Specifications For Deep Linking And Proximity Integration", November 2017, <[https://www.npci.org.in/sites/default/files/UPI%20Linking%20Specs\\_ver%201.6.pdf](https://www.npci.org.in/sites/default/files/UPI%20Linking%20Specs_ver%201.6.pdf)>.

## Authors' Addresses

Florian Dold  
Taler Systems SA  
7, rue de Mondorf  
Erpeldange L-5421  
LU

Email: [dold@taler.net](mailto:dold@taler.net)

Christian Grothoff  
BFH  
Hoeheweg 80  
Biel/Bienne CH-2501  
CH

Email: [christian.grothoff@bfh.ch](mailto:christian.grothoff@bfh.ch)

TSVWG  
Internet-Draft  
Intended status: Informational  
Expires: September 11, 2019

R. Even  
Huawei  
March 10, 2019

Fast Congestion Response  
draft-even-fast-congestion-response-00

Abstract

The high link speed (100Gb/s) in Data Centers (DC) are making network transfers complete faster and in fewer RTTs. The short data bursts requires low latency while longer data transfer require high throughput. This document describes the current state of flow control and congestion handling in the DC using RoCEv2 and suggests new directions for faster congestion control.

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 September 11, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Conventions . . . . .	3
3. Problem statement . . . . .	3
4. Security Considerations . . . . .	4
5. IANA Considerations . . . . .	4
6. References . . . . .	4
6.1. Normative References . . . . .	4
6.2. Informative References . . . . .	4
Author's Address . . . . .	5

## 1. Introduction

The high link speed (100Gb/s) in Data Centers (DC) are making network transfers complete faster and in fewer RTTs. Network traffic in a data center is often a mix of short and long flows, where the short flows require low latencies and the long flows require high throughputs. [RFC8257] titled Data Center TCP (DCTCP): TCP Congestion Control for Data Centers is an Informational RFC that extends the Explicit Congestion Notification (ECN) [RFC3168] processing to estimate the fraction of bytes that encounter congestion, DCTCP then scales the TCP congestion window based on this estimate. DCTCP does not change the ECN reporting in TCP. Other ECN notification mechanisms are specified for RTP in [RFC6679] and for QUIC [I-D.ietf-quic-transport]. The ECN notification are reported from the end receiver to the sender and the notification includes only the occurrence of ECN in the TCP case and the number of ECN marked packet for RTP and QUIC. What is common for TCP, RTP and QUIC is that the switches in the middle just monitor and report while the analysis and the rate control are done by the data sender.

In Data Centers the InfiniBand Architecture (IBA) offers a rich set of I/O services based on an RDMA access method and message passing semantics. RDMA over Converged Ethernet (RoCEv2) [RoCEv2] is using UDP as the transport for RDMA. RoCEv2 Congestion Management (RCM) provides the capability to avoid congestion hot spots and optimize the throughput of the fabric. RCM relies on the Link-Layer Flow-Control IEEE 802.1Qbb(PFC) to provide a lossless network. RoCEv2 Congestion Management(RCM) use ECN [RFC3168] to signal the congestion to the destination. The ECN notification is sent back from the receiver to the data sender using RoCEv2 Congestion Notification Packet (CNP) that notifies the sender about ECN marked packets. The rate reduction by the sender as well as the increase in data injection is left to the implementation.



## 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] [RFC8174]

## 3. Problem statement

The congestion control using ECN in the DC is done between the receiver and the sender. The network measures the traffic and informs the receiver about problems by the ECN bit. The Receiver will send to the Sender in the RoCEv2 case, a CNP message and the sender adapts by reducing the rate. The sender reduces the rate based on pre-defined policy. The sender has also a policy about when to start sending at a higher rate and by how much to increase the traffic. In the DC network when latency and high transfer rate is important there is a need to define a congestion response mechanism that will be optimized for the DC network. The behavior of the sender on congestion is not specified by RoCEv2.

This type of congestion management is re-active. The high link speed in the DC (100Gb/s) are making network transfers complete faster and in fewer RTTs; allocating flows their proper rates as quickly as possible becomes a priority. The convergence time must become a primary metric for congestion control in high speed networks.

A pro-active direction will provide more information to the sender about the congestion that can be used to optimize the congestion response allowing the network to adapt faster to the changes in the traffic conditions. This information should be available to the sender to allow fast response (RTT or lower).

The entity that measures the congestion is the switch in the network. Currently it just notifies about congestion to the receiver (ECN), may drop packets (the receiver may use IEEE 802.1Qbb to provide a lossless network). The receiver NIC informs the sender about the ECN; the sender will analyze, control and execute an action to address the congestion based on some predefined policy.

The requirement is to allow the network to control the traffic instead of the end points. The proposal is to allow the network to analyze the congestion and inform the sender (QPSource in terms of ROCEv2) how to handle the congestion when in the transport layer (directly to the data sender). In the case of RoCEv2 as the transport protocol can be a new Congestion Notification Message. This requires a new message from the network to the sender (backward notification). The proposed solution for the DC should only be

deployed in an intra-data-center environment where both endpoints and the switching fabric are under a single administrative domain.

#### 4. Security Considerations

TBD

#### 5. IANA Considerations

No IANA action

#### 6. References

##### 6.1. Normative References

- [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>>.
- [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>>.
- [RoCEv2] "Infiniband Trade Association. Supplement to InfiniBand architecture specification volume 1 release 1.2.2 annex A17: RoCEv2 (IP routable RoCE).", <<https://cw.infinibandta.org/document/dl/7781>>.

##### 6.2. Informative References

- [I-D.ietf-quic-transport] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-18 (work in progress), January 2019.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.

- [RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, DOI 10.17487/RFC6679, August 2012, <<https://www.rfc-editor.org/info/rfc6679>>.
- [RFC8257] Bensley, S., Thaler, D., Balasubramanian, P., Eggert, L., and G. Judd, "Data Center TCP (DCTCP): TCP Congestion Control for Data Centers", RFC 8257, DOI 10.17487/RFC8257, October 2017, <<https://www.rfc-editor.org/info/rfc8257>>.

## Author's Address

Roni Even  
Huawei

Email: [roni.even@huawei.com](mailto:roni.even@huawei.com)

TSVWG  
Internet-Draft  
Intended status: Informational  
Expires: January 14, 2021

Y. Li  
X. Zhou  
Huawei  
M. Boucadair  
Orange  
J. Wang  
China Telecom  
F. Qin  
China Mobile  
July 13, 2020

LOOPS (Localized Optimizations on Path Segments) Problem Statement and  
Opportunities for Network-Assisted Performance Enhancement  
draft-li-tsvwg-loops-problem-opportunities-06

Abstract

In various network deployments, end to end forwarding paths are partitioned into multiple segments. For example, in some cloud-based WAN communications, stitching multiple overlay tunnels are used for traffic policy enforcement matters such as to optimize traffic distribution or to select paths exposing a lower latency. Likewise, in satellite communications, the communication path is decomposed into two terrestrial segments and a satellite segment. Such long-haul paths are naturally composed of multiple network segments with various encapsulation schemes. Packet loss may show different characteristics on different segments.

Traditional transport protocols (e.g., TCP) respond to packet loss slowly especially in long-haul networks: they either wait for some signal from the receiver to indicate a loss and then retransmit from the sender or rely on sender's timeout which is often quite long. With the increase of end-to-end transport encryption (e.g., QUIC), traditional PEP (performance enhancing proxy) techniques such as TCP splitting are no longer applicable.

LOOPS (Local Optimizations on Path Segments) is a network-assisted performance enhancement over path segment and it aims to provide local in-network recovery to achieve better data delivery by making packet loss recovery faster. In an overlay network scenario, LOOPS can be performed over a variety of the existing, or purposely created, tunnel-based path segments.

## 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 January 14, 2021.

## Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
1.1. The Problem and Opportunity Overview . . . . .	3
1.2. Sketching a Work Direction: Rationale & Goals . . . . .	4
2. Terminology . . . . .	5
3. Usage Scenarios . . . . .	6
3.1. Cloud-Internet Overlay Network . . . . .	6
3.2. Satellite Communication . . . . .	8
3.3. Branch Office WAN Connection . . . . .	9
4. Impact of Packet loss . . . . .	10
4.1. Tail Loss or Loss in Short Flows . . . . .	10
4.2. Packet Loss in Real Time Media Streams . . . . .	11
5. Features to be Considered for LOOPS . . . . .	11
5.1. Local Recovery . . . . .	11
5.2. Congestion Control Interaction . . . . .	12

5.3. Overlay Protocol Extensions . . . . .	12
6. Local in-network Recovery and End-to-end Retransmission . . .	13
7. Summary . . . . .	14
8. Security Considerations . . . . .	15
9. IANA Considerations . . . . .	15
10. Acknowledgements . . . . .	15
11. Informative References . . . . .	15
Authors' Addresses . . . . .	18

## 1. Introduction

### 1.1. The Problem and Opportunity Overview

Packet loss is ubiquitous in Internet. A reliable transport layer normally employs some end-to-end retransmission mechanisms which also address congestion control [RFC0793] [RFC5681]. The sender either waits for the receiver to send some signals on a packet loss or sets some form of timeout for retransmission. For unreliable transport protocols such as RTP [RFC3550], optional and limited usage of end-to-end retransmission is employed to recover from packet loss [RFC4585] [RFC4588]. End-to-end retransmission to recover lost packets is slow especially when the network is long-haul. For short-lived flows and transactional flows, latency suffers a lot from tail loss.

Tunnels are widely deployed within many networks to achieve various engineering goals, including long-haul WAN interconnection or enterprise wireless access networks. A connection between two endpoints can be decomposed into many connection legs. As such, the corresponding forwarding path can be partitioned into multiple path segments that some of them are using network overlays by means of tunnels. This design serves a number of purposes such as steering the traffic, optimizing egress/ingress link utilization, optimizing traffic performance metrics (such as delay, delay variation, or loss), optimizing resource utilization by invoking resource bonding, provide high-availability, etc.

When a path is partitioned into multiple path segments that are realized typically as overlay tunnels, LOOPS (Local Optimizations on Path Segments) aims to provide in-network recovery over segments to achieve better data delivery by making packet loss recovery faster. In an overlay network scenario, LOOPS can be performed over the existing, or purposely created, overlay tunnel based path segments. Figure 1 show an overall usage scenarios of LOOPS.

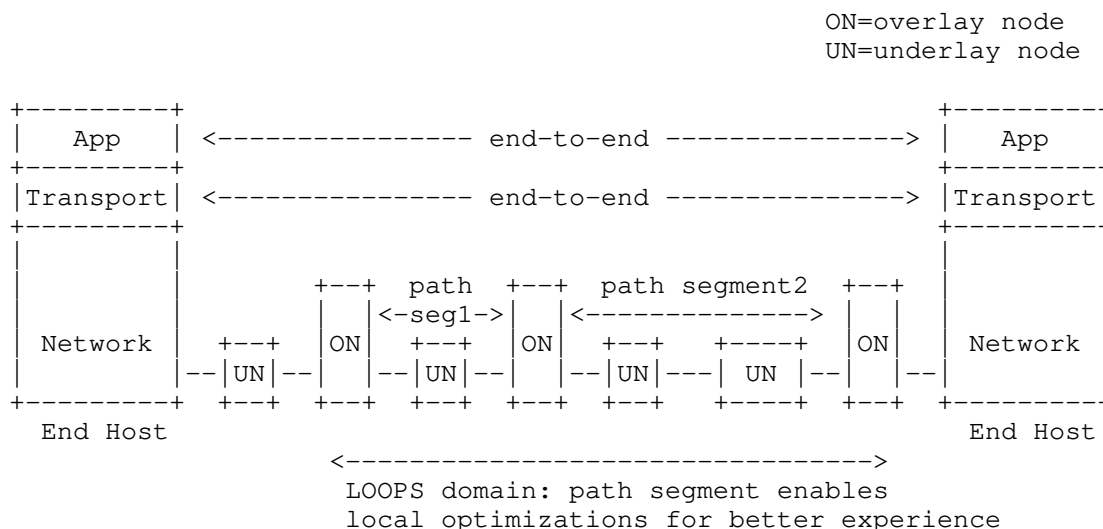


Figure 1: LOOPS Usage Scenario

## 1.2. Sketching a Work Direction: Rationale & Goals

This document sketches a proposal that is meant to experimentally investigate to what extent a network-assisted approach can contribute to increase the overall perceived quality of experience in specific situations (e.g., Sections 3.5 and 3.6 of [RFC8517]) without requiring access to internal transport primitives. The rationale beneath this approach is that some information (loss detection and segment characteristics, etc.) can be used to trigger local in-network recovery actions which have a faster effect while not impacting the end-to-end congestion control loop.

To that aim, the work is structured into two (2) phased stages:

- o Stage 1: Network-assisted optimization. This one assumes that optimizations can be implemented at the network without requiring defining new interaction with the endpoint. Existing tools such as ECN will be used. Loss signal would be converted to CE (congestion experienced) signal to interact with the end-to-end control loop.
- o Stage 2: Collaborative networking optimization. This one requires more interaction between the network and an endpoint to implement coordinated and more surgical network-assisted optimizations based on information/instructions shared by an endpoint or sharing locally-visible information with endpoint for better and faster recovery.

The document focuses on the first stage. Effort related to the second stage is out of scope of the initial planned work.

The proposed mechanism is not meant to be applied to all traffic, but only to a subset which is particularly benefits from, and has been selected for the network-assisted optimization service.

Which traffic is selected is deployment-specific and policy-based. For example, techniques for dynamic information about optimization function (e.g., SFC) may be leveraged to unambiguously identify the aggregate of traffic that is eligible to the service. Such identification may be triggered by subscription actions made by customers or be provided by a network provider (e.g., specific applications, during specific events such as during severe DDoS attack or flash crowds events).

Likewise, whether the optimization function is permanently instantiated or on-demand is deployment-specific.

This document does not intend to provide a comprehensive list of target deployment cases. Sample scenarios are described to illustrate some LOOPS potentials. Similar issues and optimizations may be helpful in other deployments such as enhancing the reliability of data transfer when a fleet of drones are used for specific missions (e.g., site inspection, live streaming, and emergency service). Captured data should be reliably transmitted via paths involving radio connections.

It is not required that all segments are LOOPS-aware to benefit from LOOPS advantages.

Section 3 presents the issues and opportunities found in some multiple path segments scenarios. Section 3 describes the impact of packet loss for different traffic. Section 5 describes the LOOPS desired features and their impact on existing network technologies. Section 6 shows the analysis on local retransmission and end-to-end retransmission. Section 7 summarizes LOOPS key elements.

## 2. Terminology

This document makes use of the following terms:

**LOOPS:** Local Optimizations on Path Segments. LOOPS includes the local in-network (i.e., non end-to-end) recovery functions and other supporting features such as local measurement, loss detection, and congestion feedback.

**LOOPS Node:** A node supporting LOOPS functions.



Overlay Node (ON): A node having overlay functions (e.g., overlay protocol encapsulation/decapsulation, header modification, TLV inspection) and LOOPS functions in the LOOPS overlay network usage scenario.

Overlay Tunnel: A tunnel with designated ingress and egress nodes using some network overlay protocol as encapsulation, optionally with a specific traffic type.

Path segment: A LOOPS enabled tunnel-based network subpath. It is used interchangeably with overlay segment in this document when the context wants to emphasize on its overlay encapsulated nature. It is also called segment for simplicity in this document.

Overlay segment: Refers to path segment.

Underlay Node (UN): A node not participating in the overlay network.

### 3. Usage Scenarios

#### 3.1. Cloud-Internet Overlay Network

CSPs (Cloud Service Providers) are connecting their data centers using the Internet or via self-constructed networks/links. This expands the traditional Internet's infrastructure and, together with the original ISP's infrastructure, forms the Internet underlay.

Automation techniques and NFV (Network Function Virtualization) make it easier to dynamically provision a new virtual node/function as a workload in a cloud for CPU/storage intensive functions. Virtual nodes can be in form of virtual machines or containers hosting the workloads sharing a physical node's infrastructure. With the aid of various mechanisms such as kernel bypassing and Virtual IO, forwarding based on virtual nodes is becoming more and more effective. The interconnection among the purposely positioned virtual nodes and/or the existing nodes with virtualization functions potentially form an overlay infrastructure. It is called the Cloud-Internet Overlay Network (CION) in this document for short.

This architecture scenario makes use of overlay technologies to direct the traffic going through the specific overlay path in order to achieve better service delivery. It purposely creates or selects overlay nodes (ON) from providers. By continuously measuring the delay of path segments and use them as metrics for path selection, when the number of overlay nodes is sufficiently large, there is a high chance that a better path could be found [DOI\_10.1109\_ICDCS.2016.49] [DOI\_10.1145\_3038912.3052560]. [DOI\_10.1145\_3038912.3052560] further shows all cloud providers

experience random loss episodes and random loss accounts for more than 35% of total loss.

Figure 2 shows an example of an overlay path over large geographic distances. An overlay node (ON) is usually a virtual node, though it does not have to be. Three path segments, i.e., ON1-ON2, ON2-ON3, ON3-ON4 are shown. Each segment transmits packets using some form of network overlay protocol encapsulation. ON has the computing and memory resources that can be used for some functions like packet loss detection, network measurement and feedback, packet retransmission and FEC (Forward Error Correction) computation. ONs here are managed by a single administrator though they can be workloads created from different CSPs.

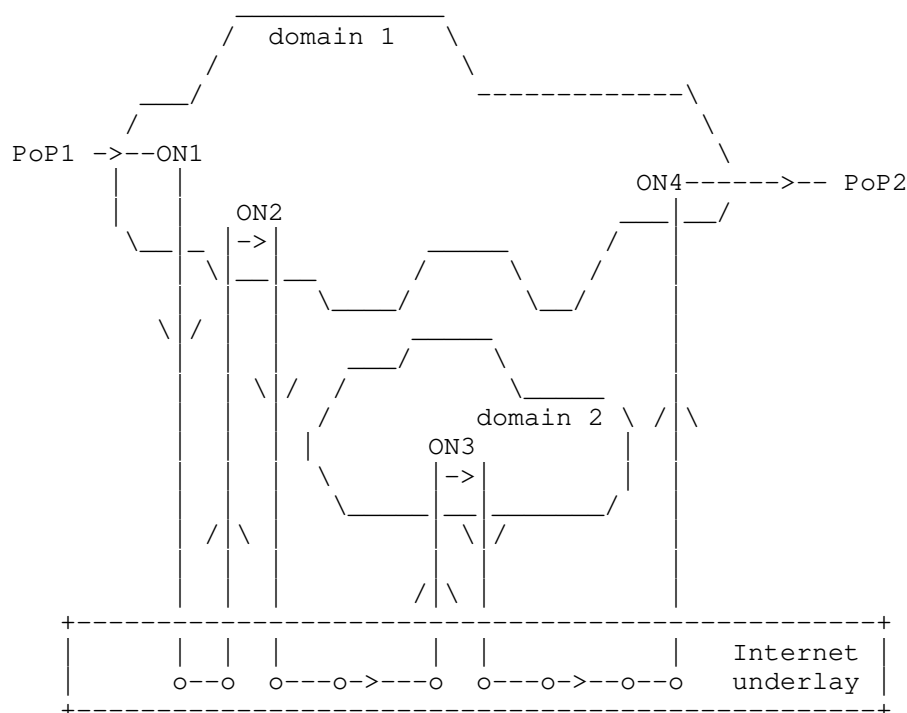


Figure 2: Cloud-Internet Overlay Network (CION)

We tested based on 37 overlay nodes from multiple cloud providers globally. Each pair of the overlay nodes are used as sender and receiver. When the traffic is not intentionally directed to go through any intermediate virtual nodes, we call the path followed by the traffic in the test the default path. When any of the virtual nodes is intentionally used as an intermediate node to forward the

traffic, the path that the traffic takes is called an overlay path. The preliminary experiments showed that the delay of a specifically selected overlay path has lower latency than the one of the default path in 69% of cases at 99% percentile and improvement is 17.5% at 99% percentile when we probe Ping packets every second for a week. The average number of hops for an overlay path is 3.02. More experimental information can be found in [DOI\_10.1109\_INFCOMW.2019.8845208].

Lower average delay does not necessarily mean less or no packet loss. Different path segments have different packet loss rates. Loss rate is another major factor impacting the user experience, especially for the short-lived or transactional flows. From some customer requirements, the target loss rate is set in the test to be less than 1% at 99% percentile and 99.9% percentile, respectively. The loss was measured between any two overlay nodes, i.e., any potential path segment. Two thousand Ping packets were sent every 20 seconds between two overlay nodes for 55 hours. This preliminary experiment showed that the packet loss rate satisfaction are only 44.27% and 29.51% at the 99% and 99.9% percentiles, respectively.

As CION naturally consists of multiple overlay segments, LOOPS can leverage this to perform local optimizations on a single hop between two overlay nodes. ("Local" here is a concept relative to end-to-end, it does not mean such optimization is limited to LAN networks.)

### 3.2. Satellite Communication

Traditionally, satellite communications deploy PEP (performance enhancing proxy [RFC3135]) nodes around the satellite link to enhance end-to-end performance. TCP splitting is a common approach employed by such PEPs, where the TCP connection is split into three: the segment before the satellite hop, the satellite section (uplink, downlink), and the segment behind the satellite hop. This requires heavy interactions with the end-to-end transport protocols, usually without the explicit consent of the end hosts. Unfortunately, this is indistinguishable from a man-in-the-middle attack on TCP. With end-to-end encryption moving under the transport (QUIC), this approach is no longer useful.

Geosynchronous Earth Orbit (GEO) satellites have a one-way delay (up to the satellite and back) on the order of 250 milliseconds. This does not include queueing, coding and other delays in the satellite ground equipment. The Round Trip Time for a TCP or QUIC connection going over a satellite hop in both directions, in the best case, will be on the order of 600 milliseconds. And, it may be considerably longer. RTTs on this order of magnitude have significant performance implications.

Packet loss recovery is an area where splitting the TCP connection into different parts helps. Packets lost on the terrestrial links can be recovered at terrestrial latencies. Packet loss on the satellite link can be recovered more quickly by an optimized satellite protocol between the PEPs and/or link layer FEC than they could be end to end. Again, encryption makes TCP splitting no longer applicable. Enhanced error recovery at the satellite link layer helps for the loss on the satellite link but doesn't help for the terrestrial links. Even when the terrestrial segments are short, any loss must be recovered across the satellite link delay. And, there are cases when a satellite ground station connects to the general Internet with a potentially larger terrestrial segment (e.g., to a correspondent host in another country). Faster recovery over such long terrestrial segments is desirable.

There are two high level classes of solutions for making encrypted transport traffic like QUIC work well over satellite:

- o Hooks in the transport protocol which can adapt to large BDPS where both the bandwidth and the latency are large. This would require end to end enhancement.
- o Capabilities (such as LOOPS) under the transport protocol to improve performance over specific segments of the path. In particular, separating the terrestrial from the satellite losses. Fixing the terrestrial loss quickly.

This document focuses on the latter.

### 3.3. Branch Office WAN Connection

Enterprises usually require network connections between the branch offices, or between branch office and cloud data center over geographic distances. With the increasing deployment of vCPE (virtual CPE), services hosted on the CPE are moved to the provider network from the customer site. Such vCPE approach enables some value added service to be provided such as WAN optimization and traffic steering.

Figure 3 shows a branch office access to public cloud via a selected PoP (point of presence) for service access or reaching another branch office via vPC (Virtual Private Cloud) interconnect. vCPE connects to the PoP which can be hundreds of kilometers away via Internet. From vCPE1 to vCPE2, it can consist of three segments, vCPE1-PoP1, PoP1-PoP2 and PoP2-vCPE2. Packet loss can happen on any of them. Segment based in-network recovery can be employed here to improve the WAN connection quality.

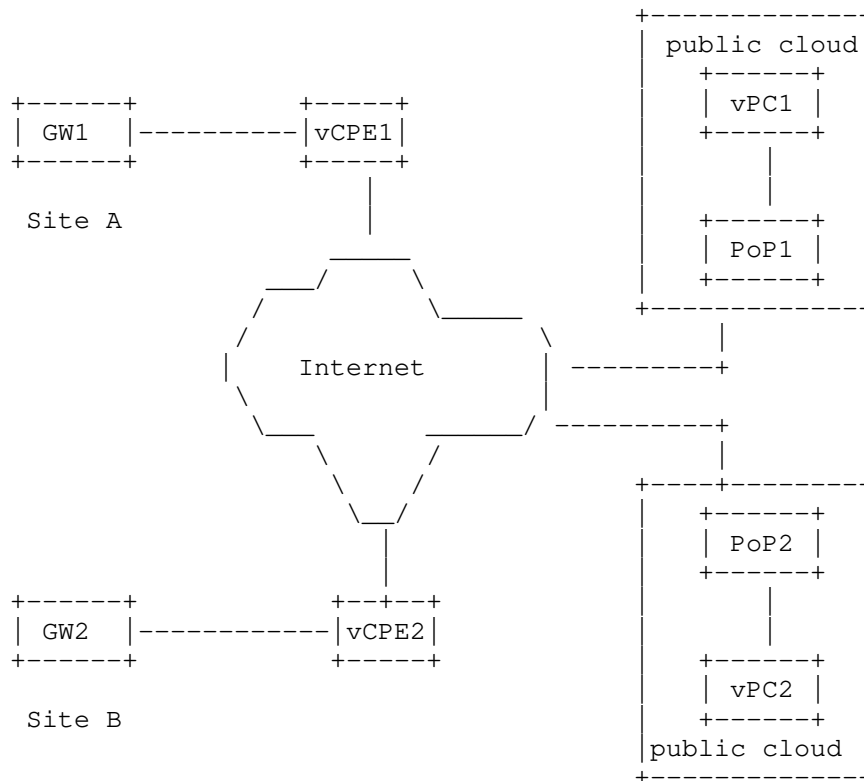


Figure 3: Enterprise Cloud Access

#### 4. Impact of Packet loss

##### 4.1. Tail Loss or Loss in Short Flows

When the lost segments are at the end of a transaction, TCP's fast retransmit algorithm does not work as there are no ACKs to trigger it. When a sender does not receive an ACK for a given segment within a certain amount of time called retransmission timeout (RTO), it re-sends the segment [RFC6298]. RTO can be as long as several seconds. Hence the recovery of lost segments triggered by RTO is lengthy. [I-D.dukkipati-tcpm-tcp-loss-probe] indicates that large RTOs make a significant contribution to the long tail on the latency statistics of short flows such as loading web pages.

The short-lived flows often complete in one or two RTTs. Even when the lost packet is not an exact tail, it can possibly add another RTT

because there may not be enough packets in flight to trigger the fast retransmit). In long-haul networks, it can result in extra time of tens or hundreds of milliseconds. For ant short lived or transactional flows, it affects the latency greatly.

An overlay segment transmits the aggregated flows from ON to ON. As short-lived flows are aggregated, the probability of tail loss over this specific overlay segment decreases compared to an individual flow. The overlay segment is much shorter than the end-to-end path, hence loss recovery over an overlay segment helps to obtain low latency.

#### 4.2. Packet Loss in Real Time Media Streams

The Real-time transport protocol (RTP) is widely used in interactive audio and video. Packet loss degrades the quality of the received media. When the latency tolerance of the application is sufficiently large, the RTP sender may use RTCP NACK feedback from the receiver [RFC4585] to trigger the retransmission of the lost packets before the playout time is reached at the receiver.

The end-to-end path over WAN can be hundreds of milliseconds, so the end-to-end feedback based retransmission may be not be very useful when applications can not tolerate one more RTT. Loss recovery over an overlay segment can then be used for the scenarios in which a shorter delayed retransmission can catch up with the playout time.

### 5. Features to be Considered for LOOPS

This section provides an overview of the LOOPS features. This section is not meant to document a detailed specification, but it is meant to highlight some design choices that may be followed during the solution design phase.

#### 5.1. Local Recovery

LOOPS (Local Optimizations on Path Segments) aims to provide in-network recovery over segments to achieve better data delivery by making packet loss recovery faster. This is viable because LOOPS nodes will be instantiated to partition the path into segments. At the same time, LOOPS does not replace the end-to-end loss recovery (if any). With the advent of automation and technologies like NFV and virtual IO, it is possible to dynamically instantiate functions to nodes. The enabling of LOOPS is expected to be dynamic. When to enable this function is out of scope. The operator or administrator can make the decision based on their historical experience or real-time monitoring.

There are two ways to recover packet, retransmission and Forward Error Correction (FEC). A document to specify the generic elements for loss detection, sequence number space, acknowledgment generation and state transition is available in [I-D.welzl-loops-gen-info].

## 5.2. Congestion Control Interaction

When a TCP-like transport layer protocol is used, local recovery in LOOPS has to interact with the upper layer transport congestion control. Classic TCP adjusts the congestion window when a loss is detected and then fast retransmit is invoked. LOOPS performs in-network recovery which may cause a loss event not being observed by the TCP sender. Then TCP sender may overshoot then.

To solve this issue, LOOPS needs to report the loss to end-to-end congestion control LOOPS. LOOPS can CE (Congestion Experienced) marks its recovered packets as the loss signal to end-to-end. Converting a packet loss signal to CE marking signal brings the benefits of reducing Head-of-Line blocking and probability of RTO expiry [RFC8087] without affecting TCP sender's loss based congestion control behaviour while enjoying the faster local recovery. ECN based indication is equivalent to a loss event at the TCP sender [RFC3168]. In this way, a requirement is set for applying LOOPS. Only ECT (ECN-Capable Transport) flows should be directed to an LOOPS enabled path segment.

## 5.3. Overlay Protocol Extensions

Some tunnel protocols such as VXLAN [RFC7348], GENEVE [I-D.ietf-nvo3-geneve], LISP [RFC6830] or CAPWAP [RFC5415] are employed in overlay network. They are used in various ways. A path can have single overlay tunnel as a sub-path or stitch multiple segments together, like VXLAN [RFC7348] or GENEVE [I-D.ietf-nvo3-geneve], or specify a sequence of intermediate nodes, as in SRv6 [RFC8754].

LOOPS does not look into the inner packet. LOOPS information is required to be embedded in the overlay protocol header. An example shown in Figure 4. The current protocol focus is GENEVE [I-D.ietf-nvo3-geneve]. The specific information is to be defined in separate documents.

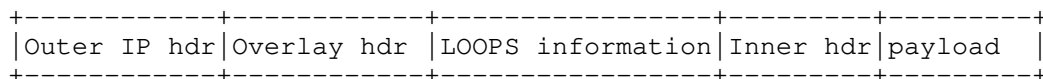


Figure 4: LOOPS Extension Header Example

## 6. Local in-network Recovery and End-to-end Retransmission

Most transport layer protocols have their own end-to-end retransmission to recover the lost packet. When LOOPS is in use, its local recovery can affect the end-to-end one. This section talks about such impacts.

There are two ways to perform local recovery, retransmission and FEC (Forward Error Correction). They are possibly used together in some cases. Such approaches between two overlay nodes recover the lost packet in relatively shorter distance and thus shorter latency. Therefore the local recovery is generally faster compared to end-to-end.

End-to-end retransmission is normally triggered by a NACK as in RTCP, multiple duplicate ACKs as in traditional TCP or time based detection as in RACK [I-D.ietf-tcpm-rack].

When FEC is used for local recovery, it may come with a buffer to make sure the recovered packets delivered are in order subsequently. Therefore the receiver side is unlikely to see the out-of-order packets and then send a NACK or multiple duplicate ACKs. The side effect to unnecessarily trigger end-to-end retransmit is minimum. When FEC is used in this way, if redundancy and block size are determined, extra latency required to recover lost packets is also bounded. Then RTT variation caused by it is predictable. In some extreme case like a large number of packet loss caused by persistent burst, FEC may not be able to recover it. Then end-to-end retransmit will work as a last resort. In summary, when FEC is used as local recovery, the impact on end-to-end retransmission is limited.

When local retransmission is used, it has the following impacts on the end-to-end retransmission.

For packet loss in RTP streaming, local retransmission can recover those packets which would not be retransmitted end-to-end otherwise due to long RTT. Therefore when the segment(s) being retransmitted on is a small portion of the whole end to end path, the retransmission will have a significant effect of improving the quality at receiver.

When the sender also re-transmits the packet based on a NACK received, the receiver may receive the duplicated retransmitted packets.

For packet loss in TCP flows, TCP RENO and CUBIC use duplicate ACKs as a loss signal to trigger the fast retransmit. Though we are not



standardize the buffering feature of a LOOPS egress, an introductory analysis is given as follows.

- o The egress overlay node can buffer the out-of-order packets for a while, giving a limited time for a packet being retransmitted somewhere in the overlay path to reach it. The retransmitted packet and the buffered packets caused by it may increase the RTT variation at the sender. When the retransmitted latency is a small portion of RTT or the loss is rare, such RTT variation will be smoothed without much impact.

The buffer management is nontrivial in this case. It has to be determined how many out-of-order packets can be buffered at the egress overlay node before it gives up waiting for a successful local retransmission. In some extreme case the lost packet is not recovered successfully locally, the sender may invoke end-to-end fast retransmit slower than it would be in classic TCP.

- o If LOOPS network does not buffer the out-of-order packets caused by packet loss, TCP sender which uses a time based loss detection like RACK [I-D.ietf-tcpm-rack] will perform well here. It uses the notion of time to replace the conventional DUPACK threshold approach to detect losses. Hence it prevents the TCP sender from invoking fast retransmit too early. Local retransmission will not interfere the sender's retransmission generally in this case. If time based loss detection is not supported at the sender, end to end retransmission may be invoked as usual. It consumes extra bandwidth Because the lost packets (i.e. recovered packet) is normally a very small percentage of the total packets. Then extra bandwidth cost is not significant.

## 7. Summary

LOOPS will extend the existing overlay protocols in data plane, potential starting from GENEVE [I-D.ietf-nvo3-geneve] which has good extensibility. Path or segment selection can be feature provided by the overlay protocols via SDN techniques [RFC7149] or other approaches and is not a part of LOOPS. LOOPS is a set of functions to be implemented on Overlay Nodes as a tunnel transport with best effort reliability. LOOPS targets the following features.

1. Local recovery: Local recovery: Retransmission, FEC, or combination thereof can be used as local recovery method. Such recovery mechanism is in-network. It is performed by two network nodes with computing and memory resources.

2. Local measurement: Ingress/Egress overlay nodes measure the local segment RTT, loss and/or throughput to immediately get the overlay segment status for loss detection.
3. Interact with end-to-end congestion control: Convert a packet loss signal to an ECN-marking signal to notify the end host sender.

## 8. Security Considerations

LOOPS does not require access to the traffic payload in clear, so encrypted payload does not affect functionality of LOOPS.

The use of LOOPS introduces some issues which impact security. ON with LOOPS function represents a point in the network where the traffic can be potentially manipulated and intercepted by malicious nodes. Means to ensure that only legitimate nodes are involved should be considered.

Denial of service attack can be launched from an ON. A rogue ON might be able to spoof packets as if it come from a legitimate ON. It may also modify the ECN CE marking in packets to influence the sender's rate. In order to protected from such attacks, the overlay protocol itself should have some built-in security protection which is used by LOOPS. The operator should use some authentication mechanism to make sure ONs are valid and non-compromised.

## 9. IANA Considerations

No IANA action is required.

## 10. Acknowledgements

Thanks to etosat mailing list about the discussion about the SatCom and LOOPS use case.

## 11. Informative References

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC3135] Border, J., Kojo, M., Griner, J., Montenegro, G., and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations", RFC 3135, DOI 10.17487/RFC3135, June 2001, <<https://www.rfc-editor.org/info/rfc3135>>.

- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <<https://www.rfc-editor.org/info/rfc4585>>.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, DOI 10.17487/RFC4588, July 2006, <<https://www.rfc-editor.org/info/rfc4588>>.
- [RFC5415] Calhoun, P., Ed., Montemurro, M., Ed., and D. Stanley, Ed., "Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification", RFC 5415, DOI 10.17487/RFC5415, March 2009, <<https://www.rfc-editor.org/info/rfc5415>>.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <<https://www.rfc-editor.org/info/rfc5681>>.
- [RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <<https://www.rfc-editor.org/info/rfc6298>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <<https://www.rfc-editor.org/info/rfc7149>>.

- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.
- [RFC8087] Fairhurst, G. and M. Welzl, "The Benefits of Using Explicit Congestion Notification (ECN)", RFC 8087, DOI 10.17487/RFC8087, March 2017, <<https://www.rfc-editor.org/info/rfc8087>>.
- [RFC8517] Dolson, D., Ed., Snellman, J., Boucadair, M., Ed., and C. Jacquenet, "An Inventory of Transport-Centric Functions Provided by Middleboxes: An Operator Perspective", RFC 8517, DOI 10.17487/RFC8517, February 2019, <<https://www.rfc-editor.org/info/rfc8517>>.
- [RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/info/rfc8754>>.
- [I-D.dukkipati-tcpm-tcp-loss-probe]  
Dukkipati, N., Cardwell, N., Cheng, Y., and M. Mathis, "Tail Loss Probe (TLP): An Algorithm for Fast Recovery of Tail Losses", draft-dukkipati-tcpm-tcp-loss-probe-01 (work in progress), February 2013.
- [I-D.ietf-nvo3-geneve]  
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-16 (work in progress), March 2020.
- [I-D.ietf-tcpm-rack]  
Cheng, Y., Cardwell, N., Dukkipati, N., and P. Jha, "RACK: a time-based fast loss detection algorithm for TCP", draft-ietf-tcpm-rack-08 (work in progress), March 2020.
- [I-D.welzl-loops-gen-info]  
Welzl, M. and C. Bormann, "LOOPS Generic Information Set", draft-welzl-loops-gen-info-03 (work in progress), March 2020.

[DOI\_10.1109\_ICDCS.2016.49]

Cai, C., Le, F., Sun, X., Xie, G., Jamjoom, H., and R. Campbell, "CRONets: Cloud-Routed Overlay Networks", 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), DOI 10.1109/icdcs.2016.49, June 2016.

[DOI\_10.1145\_3038912.3052560]

Haq, O., Raja, M., and F. Dogar, "Measuring and Improving the Reliability of Wide-Area Cloud Paths", Proceedings of the 26th International Conference on World Wide Web, DOI 10.1145/3038912.3052560, April 2017.

[DOI\_10.1109\_INFCOMW.2019.8845208]

Xu, Z., Ju, R., Gu, L., Wang, W., Li, J., Li, F., and L. Han, "Using Overlay Cloud Network to Accelerate Global Communications", IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), DOI 10.1109/infcomw.2019.8845208, April 2019.

#### Authors' Addresses

Yizhou Li  
Huawei Technologies

Email: liyizhou@huawei.com

Xingwang Zhou  
Huawei Technologies

Email: zhouxingwang@huawei.com

Mohamed Boucadair  
Orange

Email: mohamed.boucadair@orange.com

Jianglong Wang  
China Telecom

Email: wangjll.bri@chinatelecom.cn

Fengwei Qin  
China Mobile

Email: qinfengwei@chinamobile.com

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: March 11, 2020

A. Rundgren  
Independent  
September 8, 2019

Signed HTTP Requests (SHREQ)  
draft-rundgren-signed-http-requests-01

Abstract

The SHREQ specification describes how the JSON Web Signature (JWS) specification combined with the JSON Canonicalization Scheme (JCS), can be utilized to support HTTP based applications needing digitally signed requests. SHREQ is specifically tailored for Web applications using JSON as data interchange format. There is also a SHREQ scheme for HTTP requests that do not have a body ("payload") like GET. SHREQ was designed to be agnostic with respect to REST concepts versus traditional GET/POST schemes.

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 March 11, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	4
3. HTTP Processing . . . . .	4
3.1. Determining Request Type . . . . .	4
3.2. Return Codes . . . . .	5
4. Processing of JSON Based Requests . . . . .	5
4.1. Request Creation . . . . .	6
4.2. Request Validation . . . . .	8
5. Processing of URI Based Requests . . . . .	9
5.1. Request Creation . . . . .	10
5.2. Request Validation . . . . .	11
6. Common Operations . . . . .	13
6.1. Create Signable JSON Data . . . . .	13
6.2. Create Signable URI Data . . . . .	13
6.3. Create HTTP Header Object . . . . .	14
6.4. Create JWS Protected Header . . . . .	15
6.5. Create JWS String . . . . .	15
6.6. Decode JWS String . . . . .	16
6.7. Normalize Target URI . . . . .	16
6.8. Normalize Header Data . . . . .	17
6.9. Validate HTTP Header Object . . . . .	18
6.10. Validate JWS Signature . . . . .	19
6.11. Time Stamps . . . . .	20
6.12. Hash Algorithms . . . . .	20
7. Local Naming Conventions . . . . .	21
8. Attachments . . . . .	21
9. IANA Considerations . . . . .	21
10. Security Considerations . . . . .	22
11. Acknowledgements . . . . .	22
12. References . . . . .	22
12.1. Normative References . . . . .	22
12.2. Informal References . . . . .	24
Appendix A. Test Vectors . . . . .	24
A.1. Type=URI, Method=GET, Algorithm=HS256 . . . . .	24
A.2. Type=JSON, Method=POST, Algorithm=ES256 . . . . .	25
A.3. Type=JSON, Method=PUT, Algorithm=ES256 . . . . .	25
A.4. Type=URI, Method=DELETE, Algorithm=RS256 . . . . .	26
Appendix B. Other Signed HTTP Request Solutions . . . . .	27
B.1. Amazon Web Services . . . . .	27
B.2. HTTP Signatures . . . . .	27
B.3. Open Banking (UK) . . . . .	28
B.4. Financial API . . . . .	28



Appendix C. Development Portal . . . . .	28
Author's Address . . . . .	29

## 1. Introduction

Currently there is no standard for digitally signing HTTP [RFC7230] [RFC7231] requests. This has led to the development of a multitude of more or less proprietary solutions (see Appendix B), typically building on using HTTP header data for holding security constructs, while JSON request data is provided in clear in the HTTP body.

SHREQ is intended to provide a standardized alternative, including supporting the REST [REST] concept.

SHREQ builds on a common security model where all elements of an HTTP request are signed:

- o HTTP URI.
- o HTTP method.
- o HTTP body (if applicable).
- o Optional: Additional HTTP headers as defined by applications implementing this specification.

In addition there is a mandatory time stamp.

One of the design goals was turning signed requests into self-contained objects. To achieve this for HTTP requests having a JSON [RFC8259] body (see Section 4), the request data also carries the signature. This arrangement has certain implications:

- o Signed requests may be stored in databases or be embedded in other JSON objects. The latter includes supporting counter signatures. The canonicalization offered by JCS [JCS] enables validating the integrity of request data at any time.
- o For general interoperability concerns as well as due to the reliance on JCS, JSON request data is limited to the I-JSON [RFC7493] subset.

For HTTP requests that do not have a JSON body (see Section 5), the signature and additional request data is added to the original URI [RFC3986], making signed URI-only requests self-contained and serializable as well. For simplicity such requests are (in this specification NB), referred to as URI based requests.

Both variants utilize JWS [RFC7515] for holding the signature data.

For supporting signed HTTP responses any solution may be used. For maximum "symmetry" and code reuse, the [JWSJCS] scheme should be a suitable candidate since it builds on the same building blocks as SHREQ.

The intended audiences of this document are Web tool vendors, as well as designers of secure Web applications.

## 2. Terminology

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. HTTP Processing

The following subsections describe HTTP specifics associated with this specification.

### 3.1. Determining Request Type

In this specification the distinction between HTTP requests having a JSON body or not is based on the presence of a "Content-Length" header. Requests without a body object are in this specification referred to as URI based requests.

This also implies that not all header combinations permitted by HTTP can be used with this specification:

#### "Content-Length"

MUST NOT be used with URI based requests. MUST be present for requests having a body and have an argument holding the length of the body in bytes.

#### "Content-Type"

MUST NOT be used with URI based requests. MUST be present for requests having a body and have the argument "application/json".

#### "Content-Encoding"

MUST NOT be used with any requests targeting this specification.

#### "Transfer-Encoding"

MUST NOT be used with any requests targeting this specification.

### 3.2. Return Codes

This specification utilizes a single HTTP return code 400 (Bad request) for indicating syntax or security errors. Since the number of possible error conditions is significant, it is RECOMMENDED to accompany the error code with a short explanation in "text/plain" format in the HTTP Body like:

- Missing ".secinf" element
- Invalid "alg": es256
- Unknown "kid": example.com:rsa:2018.1
- com.example.jose.Core.validate(2653): Signature validation error
- Missing header variable "x-testing"

Communities using this specification MAY customize error codes if needed. However, in practice, it usually turns out to be of little value compared to a text message and a generic "hard error" code since neither users nor machines can do very much on their own to fix errors that are outside of normal processing.

Application level errors are dealt with in an application specific manner. As an example a bank application which finds out that the customer do not have enough funds to perform a transaction would presumably not return an HTTP error code but rather a specifically crafted error message to be displayed to the user.

Return codes for successful operation are application specific but are typically 200 (OK) or 201 (Created).

### 4. Processing of JSON Based Requests

Assume there is an unsigned HTTP request like the following:

```
POST /foo HTTP/1.1
Host: example.com
Content-Type: application/json
Content-Length: 1234

{
  "something": "data",

  Additional application specific properties
}
```

Adding a signature to the request above would require the following enhancements to the JSON payload:

```
{
  "something": "data",

  Additional application specific properties

  ".secinf": {
    "uri": "https://example.com/foo",
    "mtd": "POST",
    "iat": 1551709923,
    "jws": "eyJhbGciOiJIUzI1NiJ9..VHVItCBCb849imarDtjw4"
  }
}
```

Notes:

- o This specification presumes that request data featured in an HTTP body is expressed as a JSON Object.
- o The "uri" property holds a normalized target URI.
- o The "mtd" property holds the expected HTTP method. In the example it is actually redundant since the absence of a "mtd" property defaults to "POST" for JSON based requests in this specification.
- o The "iat" property holds a time stamp in UNIX "epoch" format.
- o The argument to "jws" (a detached compact JWS) was truncated for brevity.

The following subsections detail the operation for requests having an HTTP body.

#### 4.1. Request Creation

Precondition: the application data to be submitted with the request already exists in a format serializable as a JSON Object, from now on referred to as "message".

In order to create a valid signed JSON request, the following steps (and ordering) MUST be adhered to:

1. Store the target HTTP method in a variable "targetMethod".
2. Store the target URI in a variable "targetURI" after having normalized it as described in Section 6.7.
3. Add a JSON Object called ".secinf" to "message".

4. Add a property "uri" to ".secinf" where the argument is a JSON String holding a copy of "targetURI".
5. Add a property "mtd" to ".secinf" where the argument is a JSON String holding a copy of "targetMethod".

Note: for the HTTP method "POST", this step is optional because "POST" is the default for JSON based requests.

6. If there is a need to include additional HTTP headers in the signed request data, perform the following steps:
  - \* Derive (or define) the hash algorithm to use as described in Section 6.12. Save the algorithm in a variable "hashAlgorithm".
  - \* Create a header object as described in Section 6.3.

7. Add a time stamp property (see Section 6.11) to ".secinf".

Note: if the application data already contains a suitable time stamp property, this step MAY be excluded.

8. Create a "JWS Payload" [RFC7515] as described in Section 6.1.
9. Create a "JWS Protected Header" [RFC7515] as described in Section 6.4.
10. Create a JWS string object as described in Section 6.5.
11. Add a property "jws" to ".secinf" where the argument is a JSON String holding the result of the preceding step.
12. Serialize "message" into an UTF-8 [UNICODE] encoded byte array called "requestData".
13. Submit an HTTP compliant request to the "targetURI" including the following information:
  - \* HTTP method set to "targetMethod".
  - \* HTTP Header "Content-Length" with the argument set to the length of "requestData".
  - \* HTTP Header "Content-Type" with the argument set to "application/json".
  - \* All HTTP headers (if any) specified in step 6.

- \* Other HTTP headers (if any) needed by the application.

- \* An HTTP Body containing a copy of "requestData".

#### 4.2. Request Validation

In order to validate a request the following steps MUST be performed:

1. Store the HTTP method of the request in a variable "targetMethod".
2. Store the from the request recreated target URI in a variable "targetURI" after having normalized it as described in Section 6.7.
3. If the HTTP header "Content-Type" is missing or differs from "application/json" the service MUST reject the request (see Section 3.2).
4. If the HTTP header "Content-Length" is missing or malformed the service MUST reject the request (see Section 3.2). Save the length data.
5. Read HTTP body data into a byte array with the length retrieved in the preceding step.
6. Parse the byte array created in the preceding step with a JSON parser and return the result in an object from now on referred to as "message". If there are parsing errors or if "message" is not a JSON Object the service MUST reject the request (see Section 3.2).
7. Using "message", retrieve a JSON Object called ".secinf". If ".secinf" is missing or is not a JSON Object the service MUST reject the request (see Section 3.2).
8. Using ".secinf", the property "jws" is read. If "jws" is missing or is not a JSON String the service MUST reject the request (see Section 3.2). Decode the read string as described in Section 6.6.
9. Using ".secinf", the property "uri" is read. If "uri" is missing or does not match "targetURI" or is not a JSON String the service MUST reject the request (see Section 3.2).

Note: in some proxy arrangements it may be difficult retrieving the proper value of "targetURI". In such cases the comparison with "uri" MAY be disabled.

10. Using ".secinf", the property "mtd" is read. if "mtd" is missing the HTTP method is assumed to be "POST" else it is assumed to be the read value. If the derived method does not match "targetMethod" or is not a JSON String the service MUST reject the request (see Section 3.2).
11. If the optional "hdr" property is present in ".secinf" perform the following steps:
  - \* Derive the hash algorithm to use as described in Section 6.12. Save the algorithm in a variable "hashAlgorithm".
  - \* Process the argument of "hdr" as described in Section 6.9.
12. Using ".secinf", the property "iat" (see Section 6.11) is read. if "iat" is missing or is not a JSON Number the service MUST reject the request (see Section 3.2).

Note: if the application data already contains a suitable time stamp property, this step MAY be excluded.

13. Remove the "jws" property from ".secinf".
14. Create a "JWS Payload" [RFC7515] as described in Section 6.1.
15. Perform signature validation as described in Section 6.10.

Note: validation of application specific data can be performed anytime after step 6. The action(s) to perform after a possible failure is out of scope for this specification (see Section 3.2).

## 5. Processing of URI Based Requests

Assume there is an unsigned HTTP request like the following:

```
GET /users?id=435 HTTP/1.1
Host: example.com
```

The full URI would be as follows:

```
https://example.com/users?id=435
```

Adding a signature to this request according to this specification would return the following URI:

```
https://example.com/users?id=435&.jws=eyJhhiJ.eyJ7fgw.VHVIt
```

## Notes:

- o The revised URI represents a complete serializable signed request object.
- o The argument to ".jws" (a standard compact JWS) was truncated for brevity.

The middle component of the JWS string ("JWS Payload"), contains Base64Url encoded signed data related to the request. It should (after Base64Url decoding) yield a JSON Object like the following:

```
{
  "htu": "WUjqfXPztLzzXRCs6EcWCw-GC9hSL7hwCR1nG2FSvQ8",
  "mtd": "GET",
  "iat": 1551863696
}
```

## Notes:

- o The property "htu" holds a Base64Url encoded value of the normalized target URI after it has been hashed by the hash algorithm associated with the JWS signature.
- o The "mtd" property holds the expected HTTP method. In the example it is actually redundant since the absence of a "mtd" property defaults to "GET" for URI based requests in this specification.
- o The "iat" property holds a time stamp in UNIX "epoch" format.

The following subsections detail the operation for requests using an HTTP query string component for holding a signature.

### 5.1. Request Creation

In order to create a valid signed URI request, the following steps (and ordering) MUST be adhered to:

1. Store the target HTTP method in a variable "targetMethod".
2. Store the target URI in a variable "targetURI" after having normalized it as described in Section 6.7.
3. Create an empty JSON Object from now on referred to as ".secinf".
4. Derive (or define) the hash algorithm to use as described in Section 6.12. Save the algorithm in a variable "hashAlgorithm".



5. Add a property "htu" (Hashed Target URI) to ".secinf" where the argument is a JSON String holding the outcome of the process described in Section 6.2.

6. Add a property "mtd" to ".secinf" where the argument is a JSON String holding a copy of "targetMethod".

Note: for the HTTP method "GET", this step is optional because "GET" is the default for URI based requests.

7. If there is a need to include additional HTTP headers in the signed request data, create a header object as described in Section 6.3.
8. Add a time stamp property (see Section 6.11) to ".secinf".
9. Serialize the ".secinf" JSON Object into a UTF-8 [UNICODE] byte array representing a "JWS Payload" [RFC7515].
10. Create a "JWS Protected Header" [RFC7515] as described in Section 6.4.
11. Create a JWS string object as described in Section 6.5.
12. Create a query string component by concatenating ".jws=" with the JWS string created in the preceding step. This component is appended to the original unsigned request URI prepended by & or ? depending on if it is the only query component or not.
13. Submit an HTTP compliant request to the target URI including the following information:
  - \* HTTP method set to "targetMethod".
  - \* All HTTP headers (if any) specified in step 7.
  - \* Other HTTP headers (if any) needed by the application.

## 5.2. Request Validation

In addition to normal validation of received data (which may be carried out before or after the steps outlined here), the following steps MUST be performed in order to validate a URI based HTTP request:

1. Store the HTTP method of the request in a variable "targetMethod".

2. Store the from the request recreated target URI in a variable "targetURI" after having normalized it as described in Section 6.7.
3. Extract the JWS string from the ".jws" element which MUST reside in the query string of "targetURI". If the JWS string is missing the service MUST reject the request (see Section 3.2).
4. Decode the argument of the preceding step as described in Section 6.6.
5. Remove the ".jws" query string component from "targetURI". Note that if the ".jws" query component is the last part of "targetURI", the delimiter immediately preceding the ".jws" component is removed, else the succeeding delimiter is removed.
6. Parse "JWS Payload" (created in step 4) with a JSON parser and from now on refer to the result as ".secinf". If ".secinf" is not a JSON Object the service MUST reject the request (see Section 3.2).
7. Derive the hash algorithm to use as described in Section 6.12. Save the algorithm in a variable "hashAlgorithm".
8. Using ".secinf" the property "htu" is read. If "htu" is missing or is not a JSON String the service MUST reject the request (see Section 3.2).
9. Perform the operation described in Section 6.2 and compare the outcome with the argument to "htu". If these value do not match the service MUST reject the request (see Section 3.2).

Note: in some proxy arrangements it may be difficult retrieving the proper value of "targetURI". In such cases the comparison with "htu" MAY be disabled.

10. Using ".secinf", the property "mtd" is read. if "mtd" is missing the HTTP method is assumed to be "GET" else it is assumed to be the read value. If the derived method does not match "targetMethod" or is not a JSON String the service MUST reject the request (see Section 3.2).
11. If the optional "hdr" property is present in ".secinf", process the argument of "hdr" as described in Section 6.9.
12. Using ".secinf", the property "iat" (see Section 6.11) is read. if "iat" is missing or is not a JSON Number the service MUST reject the request (see Section 3.2).

13. Perform signature validation as described in Section 6.10.

Note: validation of application specific data can be performed anytime. The action(s) to perform after a possible failure is out of scope for this specification (see Section 3.2).

## 6. Common Operations

This specification builds on a modular scheme using common procedures described in the following subsections.

### 6.1. Create Signable JSON Data

Unsigned request data is now supposed to reside in "message". To facilitate resilience against (legitimate) variances in JSON processing between different platforms and systems, "message" needs to be canonicalized and serialized into a UTF-8 [UNICODE] encoded byte array. If the used JSON tools offer intrinsic support for JCS [JCS], this is typically a single operation, else the followings steps are performed:

1. Serialize "message" using standard JSON tools for the platform.
2. Create a canonical and UTF-8 encoded form of the data created in the preceding step, through an external software solution supporting JCS.

The output from JCS represents a "JWS Payload" [RFC7515].

### 6.2. Create Signable URI Data

For URI based requests, the steps to create signable URI data are as follows:

1. Convert "targetURI" into a UTF-8 [UNICODE] encoded byte array.
2. Create a digest of the result of the preceding step using the previously defined "hashAlgorithm". The result is a byte array.
3. The result of the preceding step is subsequently Base64Url encoded.

The test vectors in Appendix A provide a few examples showing authentic values of the "htu" (Hashed Target URI) property.

### 6.3. Create HTTP Header Object

To create a digest of headers to be included in a signed request, perform the following operations:

1. Create an empty string "headerBlob".
2. Create an empty string "headerList".
3. Create a collection of headers to be sent as described in Section 6.8.
4. Enumerate the "headerCollection" and perform the following steps for each entry:
  - \* Append header field name to "headerList".
  - \* Append header field name to "headerBlob".
  - \* Append a semicolon (':') to "headerBlob".
  - \* Append header field value to "headerBlob".
  - \* For all but the last entry, append a newline (U+000A) to "headerBlob".
  - \* For all but the last entry, append a comma (',') to "headerList".
5. Create a two element JSON Array object "headerData".
6. Run the previously defined "hashAlgorithm" (see Section 6.12) over the UTF-8 [UNICODE] representation of "headerBlob".
7. Base64Url-encode the result of the preceding operation and assign the result to the first entry in "headerData" in the form of a JSON String.
8. Assign a copy of "headerList" to the second entry in "headerData" in the form of a JSON String.
9. Add a property "hdr" to ".secinf" using a copy of "headerData" as argument.

Below is an example of header input data:

```
x-debug: full
Cache-Control: max-age=60, must-revalidate
```

Applying the process described in this subsection and using the SHA-256 [SHS] hash algorithm should generate the following ".secinf" data:

```
"hdr": ["Ljzuq8C9PScbvLpBxG8GNOs-WQUd7gl7R64izahhe-0",  
        "x-debug,cache-control"]
```

#### 6.4. Create JWS Protected Header

Create a "JWS Protected Header" [RFC7515] JSON Object with algorithm and key data adapted for the application. Below is a minimal example:

```
{  
  "alg": "ES256"  
}
```

#### 6.5. Create JWS String

To create a compact JWS object (a string), perform the following steps:

1. Serialize the previously defined "JWS Protected Header" object into a UTF-8 [UNICODE] encoded byte array.
2. Base64Url-encode the output from the preceding step into a local variable "jwsProtectedHeaderB64U".
3. Base64Url-encode the previously defined variable "JWS Payload" into a local variable "jwsPayloadB64U".
4. Set a local variable "signedData" to the UTF-8 encoded representation of the concatenation of:
  - \* The previously defined variable "jwsProtectedHeaderB64U".
  - \* A point character (".").
  - \* The previously defined variable "jwsPayloadB64U".
5. Use the designated signature key, signature algorithm and "signedData" to create a "JWS Signature" object (byte array).
6. Return the string consisting of the concatenation of:
  - \* The previously defined variable "jwsProtectedHeaderB64U".
  - \* A period character ('.').

- \* For URI based requests only: "jwsPayloadB64U". That is, JSON based requests use the detached JWS format described in Appendix F of [RFC7515].
- \* A period character ('.').
- \* The previously defined variable "JWS Signature", here encoded in Base64Url [RFC4648].

#### 6.6. Decode JWS String

The following processing steps presume that there is an input string holding a JWS compact object, here called "jwsString":

1. Verify that "jwsString" has the syntax

"header.payload.signature"

where the length of the "payload" element is zero for JSON based requests and non-zero for URI based requests. That is, JSON based requests use the detached JWS format described in Appendix F of [RFC7515].

2. Assign the "header" portion of "jwsString" to a variable "jwsProtectedHeaderB64U".
3. Base64Url-decode "jwsProtectedHeaderB64U" into a byte array.
4. Parse the output from the preceding step with a JSON parser and assume that the result (which MUST be a JSON Object) represents a "JWS Protected Header" [RFC7515].
5. For URI based requests only:  
base64Url-decode the "payload" portion of "jwsString" into a byte array representing a "JWS Payload" [RFC7515].
6. Base64Url-decode the "signature" portion of "jwsString" into a byte array representing a "JWS Signature" [RFC7515].

If any of the steps above fail, the service MUST reject the request (see Section 3.2).

#### 6.7. Normalize Target URI

To facilitate comparison between actual (received) URIs and signed URIs, URIs MUST be normalized according to the following:

The schema default ports 443 and 80 MUST be removed from HTTPS and HTTP URIs respectively.

URI characters that have been escaped that are in the non-reserved set [ALPHA DIGIT '-' '.' '\_' '~'] MUST be restored in their natural form.

Escape sequences MUST be transformed into uppercase.

Non-ASCII characters MUST be escaped to their UTF-8 [UNICODE] counterpart.

Host names MUST be lowercased.

The following URI shows a non-normalized URI:

```
https://EXAMPLE.COM:443/%63EURO%2f
```

Note: EURO denotes a single Euro character (Unicode: U+20AC), which not being ASCII, is currently not displayable in RFCs.

The same URI after normalization:

```
https://example.com/c%E2%82%AC%2F
```

[[ This section is still incomplete ]]

## 6.8. Normalize Header Data

Headers to be included in signed requests MUST be normalized. This subsection shows a common procedure for senders and receivers based on Section 3.2.4 of [RFC7230].

Collect received headers or headers to be submitted in a list of header field name and header field value pairs according to the following:

- o Header field names MUST be lowercased.
- o Leading and trailing optional whitespace (OWS) in the header field value MUST be omitted. If there are multiple instances of the same header field name, all header field values associated with the header field name MUST be concatenated, separated by an ASCII comma and an ASCII space (', '), and used in the order in which they are intended to appear in an HTTP message. Any other modification to the header field value MUST NOT be made.

This list is referred to as "headerCollection" in other places in this specification.

Below is an example of header input data:

```
x-debug: full
Cache-control: max-age=60
Cache-Control: must-revalidate
```

Applying the process described in this subsection should generate the following collection:

=====	
Header Field Name	Header Field Value
=====	
x-debug	full
-----	
cache-control	max-age=60, must-revalidate
-----	

For interoperability reasons it is RECOMMENDED to not use duplicate header names for headers that are to be signed. Apparently proxy servers do not always honor original header ordering.

#### 6.9. Validate HTTP Header Object

To validate a digest of headers in a signed request, perform the following operations:

1. Create a collection of received headers as described in Section 6.8.
2. Create an empty string "headerBlob".
3. Read the "hdr" property of ".secinf". This MUST be a JSON Array holding exactly two JSON String elements.
4. Perform the following actions on the data obtained in the preceding step:
  - \* Base64Url-decode the first string into a byte array "digest".
  - \* Split the second string into ordered array of strings called "headerList". Note that the format MUST be a list of header field names in lowercase, separated by comma (',') characters. There MUST NOT be any whitespace or terminating comma in this string.



- \* Verify that the received "headerList" contains the header field names as defined by an application specific policy.
5. Enumerate the "headerList" and perform the following steps for each entry:
    - \* Append header field name to "headerBlob".
    - \* Append a semicolon (':') to "headerBlob".
    - \* Retrieve the matching header field value from "headerCollection".
    - \* Append the result of the preceding step to "headerBlob".
    - \* For all but the last entry, append a newline (U+000A) to "headerBlob".
  6. Run the previously defined "hashAlgorithm" (see Section 6.12) over the UTF-8 [UNICODE] representation of "headerBlob".
  7. Verify that the result of the preceding step is identical to "digest".

If any of the steps above fail, the service MUST reject the request (see Section 3.2).

Note that this specification does not enforce any particular ordering of signed header elements.

#### 6.10. Validate JWS Signature

Validation of the JWS [RFC7515] object, using the previously extracted and decoded objects requires the following steps:

1. Verify that the received "JWS Protected Header" contains a JWS algorithm ("alg") and key identifiers that matches the needs of the application.
2. Retrieve the signature validation key. This part is application specific since the key may be implicit, specified by a key ID ("kid") or be supplied in a certificate path ("x5c").
3. Set a local variable "signedData" to the UTF-8 [UNICODE] encoded representation of the string created by concatenating the following elements:
  - \* The previously collected variable "jwsProtectedHeaderB64U".

- \* A period character ('.').
  - \* The previously collected variable "JWS Payload", but here encoded in Base64Url [RFC4648].
4. Validate the signature using the algorithm retrieved in step 1, the signature validation key from step 2, "signedData" from step 3 and the previously collected "JWS Signature" object (byte array).

If any of the steps above fail, the service MUST reject the request (see Section 3.2).

#### 6.11. Time Stamps

Time stamps have the same name ("iat"), format and function as described in JWT [RFC7519], Section 4.1.6. However, in this specification time stamps are REQUIRED, and stored in the ".secinf" JSON Object.

Although JWT permits non-integer values, implementers of this specification SHOULD limit generated time stamp granularity to seconds and use integer representation.

The policy with respect to the difference between the current time and received time stamps is out of scope for this specification. However, for security reasons it is generally a good idea limiting deviations to a few minutes as well as using network based clock synchronization in both ends.

#### 6.12. Hash Algorithms

Inclusion of HTTP header elements as well as the "htu" property of URI based requests depends on digests produced by a hash algorithm. The default is using the hash algorithm associated with the JWS signature algorithm ("alg") featured in the "JWS Protected Header". That is, the JWA [RFC7518] algorithms "ES256" and "HS384" imply the hash algorithms SHA-256 and SHA-384 respectively.

In case this is not desired, this specification permits overriding the default by including a "hao" (Hash Algorithm Override) property in the ".secinf" JSON Object. The currently recognized arguments to "hao" are:

"S256" for SHA-256 [SHS]

"S384" for SHA-384 [SHS]

"S512" for SHA-512 [SHS]

## 7. Local Naming Conventions

Although using the ".secinf" JSON property name and ".jws" query component name is RECOMMENDED, this specification permits (=being considered as compatible), the use of local naming conventions as long as the specified procedures and formats are adhered to.

Local naming conventions MUST be properly communicated in the community using them.

## 8. Attachments

[[

It may be possible to extending the JSON based request to also support attachments using MIME multipart schemes. This is though currently out of scope for this specification.

An alternative to attachments is featuring such data in Base64Url encoded fields.

Recently, "cloud" based schemes using (preferably time-limited) URLs with hard-to-guess nonce values have become a viable method for supporting related additional data. By combining hash values with such URLs, integrity of the additional data can be verified.

]]

## 9. IANA Considerations

This document currently has no IANA actions but the reserved names below could be candidates for IANA registration:

.secinf

JSON Object holding the security related data of this specification.

.jws

HTTP query component holding the security related data of this specification.

The hash algorithms defined in Section 6.12 could also benefit from IANA registration.

## 10. Security Considerations

The purpose of this specification is adding an integrity and authorization layer to HTTP requests. This part is subject to the same security considerations as the underpinning JCS and JWS schemes.

For most applications HTTPS [RFC7231] would be the logical choice, not only for protecting application data from snooping, but also to not unnecessarily reveal data about signature keys.

In a cloud scenario with Web servers open for access by any party new security challenges are introduced. Cryptographic solutions protect data but may also add vulnerabilities to denial-of-service attacks since they often need substantial processing.

Protecting against replay attacks is important because replay may actually be a legitimate facility for systems repeating a request due to a communication failure. This cannot be entirely solved by time stamps and cryptography; you usually need unique transactions IDs and data base support as well. For reliable operation there must be common rules within a community using such features. The REST [REST] paradigm also requires such measures due to the idempotent operation specified for "PUT", "GET" and "DELETE".

## 11. Acknowledgements

Parts of this specification were derived from the HTTP signature [HTTPSIG] draft.

## 12. References

### 12.1. Normative References

- [JCS] A. Rundgren, B. Jordan, S. Erdtman, "JSON Canonicalization Scheme - Work in progress", <<https://tools.ietf.org/html/draft-rundgren-json-canonicalization-scheme-05>>.
- [JWSJCS] A. Rundgren, "Combined JWS and JCS Signature Scheme - Work in progress", <<https://github.com/cyberphone/jws-jcs>>.
- [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>>.

- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.
- [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <<https://www.rfc-editor.org/info/rfc7231>>.
- [RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <<https://www.rfc-editor.org/info/rfc7493>>.
- [RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <<https://www.rfc-editor.org/info/rfc7515>>.
- [RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <<https://www.rfc-editor.org/info/rfc7518>>.
- [RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <<https://www.rfc-editor.org/info/rfc7519>>.
- [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>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [SHS] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, August 2015, <<https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>>.

[UNICODE] The Unicode Consortium, "The Unicode Standard, Version 10.0.0",  
<<https://www.unicode.org/versions/Unicode10.0.0/>>.

## 12.2. Informal References

[AWS] Amazon.com, "Signing AWS API Requests",  
<[https://docs.aws.amazon.com/general/latest/gr/signing\\_aws\\_api\\_requests.html](https://docs.aws.amazon.com/general/latest/gr/signing_aws_api_requests.html)>.

[FAPI] Open ID, "Financial-grade API",  
<<https://openid.net/wg/fapi/>>.

[HTTPSIG] M. Cavage, M. Sporny, "Signing HTTP Messages",  
<<https://tools.ietf.org/html/draft-cavage-http-signatures-10>>.

[OBIE] Open Banking UK, "Open Banking API",  
<<https://www.openbanking.org.uk/>>.

[REST] Roy Fielding, "Architectural Styles and the Design of Network-based Software Architectures",  
<<http://roy.gbiv.com/pubs/dissertation/top.htm>>.

[STET] STET, "PSD2 API V1.4.1", <<https://www.stet.eu/en/psd2/>>.

## 12.3. URIs

[1] <https://github.com/cyberphone/ietf-signed-http-requests>

## Appendix A. Test Vectors

The following test vectors "activate" all parts of the specification. After removing the line breaks needed for publishing, the test vectors are supposed to be fully validatable.

### A.1. Type=URI, Method=GET, Algorithm=HS256

Target URI:

<https://example.com/users/456>

Signed URI:

<https://example.com/users/456?.jws=eyJhbGciOiJIUzI1NiJ9.eyJodHUiOiJmaVZpNGpZaER0N1ZDdVFJS1VJZFdTbkVXZm9oX05YSGZMVFpORWVITYXZZiwiYWw0IjoxNTUxOTUxOTAwfQ.Wll5cFEE9sidHs0lsADus8kbHhNAC5DCzyytYoAtT2g>

Decoded JWS Payload:

```
{
  "htu": "fiVi4jYhDt7VCuQIKUIdWINEWfoh_NXHfLTZNeeSavY",
  "iat": 1551951900
}
```

Symmetric signature validation key, here in hexadecimal notation:

7fdd851a3b9d2dafc5f0d00030e22b9343900cd42ede4948568a4a2ee655291a

A.2. Type=JSON, Method=POST, Algorithm=ES256

Target URI:

https://example.com/users

JSON Body:

```
{
  "name": "John Doe",
  "profession": "Unknown",
  ".secinf": {
    "uri": "https://example.com/users",
    "iat": 1551951900,
    "jws": "eyJhbGciOiJIJFZlIiwiaXNjaWkiOiJES256IiwiaWF0IjoxNTUxOTUxOTAwfQ.eyJhbGciOiJIJFZlIiwiaXNjaWkiOiJES256IiwiaWF0IjoxNTUxOTUxOTAwfQ.eyJhbGciOiJIJFZlIiwiaXNjaWkiOiJES256IiwiaWF0IjoxNTUxOTUxOTAwfQ"
  }
}
```

Public signature validation key, here in PEM format:

```
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEcensDzcMEkgiePz6DXB7cDuwFems
hAFR90UNVQFCg8TGryvN7p7AbT55VxIXvYnvuAqIPQgefOnAdpTu3qdV5g==
-----END PUBLIC KEY-----
```

A.3. Type=JSON, Method=PUT, Algorithm=ES256

Target URI:

https://example.com/users/456

JSON Body:

```
{
  "name": "Jane Smith",
  "profession": "Hacker",
  ".secinf": {
    "uri": "https://example.com/users/456",
    "mtd": "PUT",
    "iat": 1551951900,
    "jws": "eyJhbGciOiJFUzI1NiJ9..VWtXYcgr6OTCcJg6XZzPkHsLU-jUT
TlHoQ92bihMIDlXR7xNfmxlHWSUc9cyFCxzsBy9yq33eFn3fApIH42SA"
  }
}
```

Public signature validation key, here in PEM format:

```
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEcensDzcMEkgiePz6DXB7cDuwFems
hAFR90UNVQFCg8TGryvN7p7AbT55VxIXvYnvuAqIPQgefOnAdpTu3qdV5g==
-----END PUBLIC KEY-----
```

#### A.4. Type=URI, Method=DELETE, Algorithm=RS256

Target URI:

<https://example.com/users/456>

Signed URI:

[https://example.com/users/456?.jws=eyJhbGciOiJFUzI1NiJ9.eyJodHUiOiI5R3FtRDBSRWRqSDFZNklvSXR3UjdKRURuU0pjVzNuSnhoM085eHQ3Zk1RQ1cyN3FtOEQyWUNtN1h4RzRwU1hwOGJFM3lTT3RzWlhIR0VJSWw1M05jQSIsIm10ZCI6IkRFTEVURSI6Im1hdCI6MTU1MTk1MTkwMCwiaGFvIjo1UzUxMiIsImhkeiI6WyI6ZXBrQno4RUJwMUxYX01EdFd1WnFWZjFLYjJya1FNZzE5RjVvT2FhbG91SVFpS1Z1SHBrSG5wdWFLMlZzZbVZ2bEpOSGxey2NEeHFxVGQxNFU5VXg5USIsIngtZGVidWciXX0.YRTEbCrOy11c10HcPSDX\\_Dct156S5qmcYWFcuG6wqsgg7vnCr22vCDhE1xJLxeM2hp\\_-gSmdxykJ-Q060xetax-nMmXUhrDtcRoeCf012-xDTymZTjXtb11SX6Pnmt9CnM4ZOVrJVro7eLW8hCc4p5As7zDS2arNM\\_-IsWiNJ1T25EDb8ZS7kLLSA6Im1o31o8815kC0oHNI0q-1ZeaOX3DhnL1tMJKZQzrItXvmZ0oqJ3kL8bxF6ag1OFC0zOYUU2kciIf55jVcfBgwupecFw-rN56QEg8PzA8YA-nGPWHBpxJUWWseY4qXZudRcQQZtms7Yc1yK7z3fNhht6Oh1A](https://example.com/users/456?.jws=eyJhbGciOiJFUzI1NiJ9.eyJodHUiOiI5R3FtRDBSRWRqSDFZNklvSXR3UjdKRURuU0pjVzNuSnhoM085eHQ3Zk1RQ1cyN3FtOEQyWUNtN1h4RzRwU1hwOGJFM3lTT3RzWlhIR0VJSWw1M05jQSIsIm10ZCI6IkRFTEVURSI6Im1hdCI6MTU1MTk1MTkwMCwiaGFvIjo1UzUxMiIsImhkeiI6WyI6ZXBrQno4RUJwMUxYX01EdFd1WnFWZjFLYjJya1FNZzE5RjVvT2FhbG91SVFpS1Z1SHBrSG5wdWFLMlZzZbVZ2bEpOSGxey2NEeHFxVGQxNFU5VXg5USIsIngtZGVidWciXX0.YRTEbCrOy11c10HcPSDX_Dct156S5qmcYWFcuG6wqsgg7vnCr22vCDhE1xJLxeM2hp_-gSmdxykJ-Q060xetax-nMmXUhrDtcRoeCf012-xDTymZTjXtb11SX6Pnmt9CnM4ZOVrJVro7eLW8hCc4p5As7zDS2arNM_-IsWiNJ1T25EDb8ZS7kLLSA6Im1o31o8815kC0oHNI0q-1ZeaOX3DhnL1tMJKZQzrItXvmZ0oqJ3kL8bxF6ag1OFC0zOYUU2kciIf55jVcfBgwupecFw-rN56QEg8PzA8YA-nGPWHBpxJUWWseY4qXZudRcQQZtms7Yc1yK7z3fNhht6Oh1A)

Decoded JWS Payload:



```
{
  "htu": "9GqmD0REdjH1Y6IoItwR7JEDnSJcW3nJxh3O9xt7fMQCW27qm8D2YC
m7XxG4pSXp8bE3ySOtsZXHGEEI153NcA",
  "mtd": "DELETE",
  "iat": 1551951900,
  "hao": "S512",
  "hdr": ["3epkBz8EBp1LX_MDtWuZqVf1Kb2rjQMg19F5oOaanOuIQiKVuHpkH
nVuaK2VYmVvlJNHLdCcDxqqTd14U9Ux9Q", "x-debug"]
}
```

Note the overridden hash algorithm.

Required HTTP Headers:

x-debug: full

Public signature validation key, here in PEM format:

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAhFWEXArvaZEpSP5qNX7x
4C4Hl28GJQTNvnDwkfqiwS63kXbdyPeS06bz6GnY3tfQ/093nGauWsimqKBmGAGM
PtsV83Qxw10IeO4ujbIIb9pema0qtVqs0MWlHxklZGFkYfAmbuEUFxYDeLDHe0bk
kXbSlB7/t8pCSvc8HLgHjEQjYOlFRwjR0D+uLo+xgsCbpmCtYkB51cT/zFgpRgY4
zJNLsv7GZiz2S4Fc5ArGjd34lL47+L8bozuYjqNOv9sqX0Zgl15XaJlndvr7UqZu
1xQFgm38reoM3IarBP/SkEFbt/v9iak602VO3k28fQhMaocP7JWR2YLT3kZM0+WT
FwIDAQAB
-----END PUBLIC KEY-----
```

## Appendix B. Other Signed HTTP Request Solutions

This appendix briefly outlines a few other solutions addressing Signed HTTP Requests.

### B.1. Amazon Web Services

AWS provides a system for their clients using HTTP headers holding security constructs while the digested HTTP body may hold any valid media type. For more information see the [AWS]. Signatures may also be added to query strings in a similar fashion to Section 5.

### B.2. HTTP Signatures

HTTP Signatures is a system using HTTP headers holding security constructs while the (optional) digested HTTP body may hold any valid media type. This scheme has been adopted by the French Open Banking API [STET]. For more information see the Internet draft [HTTPSIG]. HTTP Signatures also supports signed response data.

### B.3. Open Banking (UK)

The current (3.1) version of the Open Banking API [OBIE] use a scheme where a dedicated HTTP header holds a detached JWS signature covering a clear text JSON message in the HTTP body:

```
POST /foo HTTP/1.1
Host: example.com
Content-Type: application/json
x-jws-signature: eyJhbGciOiJSUzI1NiI..SD7xMbpL-2QgwUsAlMGzw
Content-Length: 2765

{
  "something": "data",

  Additional application specific properties
}
```

Notes:

- o The HTTP URI, method and headers are unsigned.
- o The signature argument (a JWS) was truncated for brevity.

### B.4. Financial API

The current version (Draft 06) of the financial API [FAPI] use a scheme where the payload is signed using JWS in Base64Url mode:

```
POST /foo HTTP/1.1
Host: example.com
Content-Type: application/jws
Content-Length: 1288

eyJhbGciOiJSUzI1NiI..ew0KICJfZds56gty5ypc3MiOiA.2QgwUsA565656lMGzw
```

Notes:

- o The HTTP URI, method and headers are unsigned.
- o The JWS signature was truncated for brevity.

## Appendix C. Development Portal

The SHREQ specification is currently developed at:  
<https://github.com/cyberphone/ietf-signed-http-requests> [1].

Author's Address

Anders Rundgren  
Independent  
Montpellier  
France

Email: anders.rundgren.net@gmail.com

URI: <https://www.linkedin.com/in/andersrundgren/>