Internet Engineering Task Force (IETF)
Request for Comments: 6773
Updates: 4340, 5762
Category: Standards Track
ISSN: 2070-1721

T. Phelan
Sonus
G. Fairhurst
University of Aberdeen
C. Perkins
University of Glasgow

November 2012

DCCP-UDP: A Datagram Congestion Control Protocol UDP Encapsulation for NAT Traversal

Abstract

This document specifies an alternative encapsulation of the Datagram Congestion Control Protocol (DCCP), referred to as DCCP-UDP. This encapsulation allows DCCP to be carried through the current generation of Network Address Translation (NAT) middleboxes without modification of those middleboxes. This document also updates the Session Description Protocol (SDP) information for DCCP defined in RFC 5762.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6773.

Copyright Notice

Table of Contents

1. Introduction

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

DCCP support has been specified for devices that use Network Address Translation (NAT) or Network Address and Port Translation (NAPT) [RFC5597]. However, there is a significant installed base of NAT/NAPT devices that do not support [RFC5597]. It is therefore useful to have an encapsulation for DCCP that is compatible with this installed base of NAT/NAPT devices that support [RFC4787] but do not support [RFC5597]. This document specifies that encapsulation, which is referred to as DCCP-UDP. For convenience, the standard encapsulation for DCCP [RFC4340] (including [RFC5596] as required) is referred to as DCCP-STD.

The encapsulation described in this document may also be used as a transition mechanism to enable support for DCCP in devices that support UDP but do not yet natively support DCCP. This also allows the DCCP transport to be implemented within an application using DCCP-UDP.

This document also updates the SDP specification for DCCP [RFC5762] to convey the encapsulation type. In this respect only, it updates the method in [RFC5762].

The DCCP-UDP encapsulation specified in this document supports all of the features contained in DCCP-STD, but with limited functionality for partial checksums.

Network optimisations for DCCP-STP and UDP may need to be updated to allow these optimisations to take advantage of DCCP-UDP. Encapsulation with an additional UDP protocol header can complicate or prevent inspection of DCCP header fields by equipment along the network path in the case where multiple DCCP connections share the same UDP 4-tuple, for example, routers that wish to identify DCCP ports to perform Equal-Cost Multi-Path (ECMP) routing, network devices that wish to inspect DCCP ports to inform algorithms for sharing the network load across multiple links, firewalls that wish to inspect DCCP ports and service codes to inform algorithms that implement access rules, media gateways that inspect SDP information to derive characteristics of the transport and session, etc.

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. DCCP-UDP

The basic approach is to insert a UDP [RFC0768] header between the IP header and the DCCP packet. Note that this is not a tunneling approach. The IP addresses of the communicating end systems are carried in the IP header. The method does not embed additional IP addresses.

The method is designed to support use when these addresses are modified by a device that implements NAT/NAPT. A NAT translates the IP addresses, which impacts the transport-layer checksum. A NAPT device may also translate the port values (usually the source port). In both cases, the outer transport header that includes these values would need to be updated by the NAT/NAPT.

A device offering or using DCCP services via DCCP-UDP encapsulation listens on a UDP port (default port, 6511) or may bind to a specified port utilising out-of-band signalling, such as the Session Description Protocol (SDP). The DCCP-UDP server accepts incoming packets over the UDP transport and passes the received packets to the DCCP protocol module, after removing the UDP encapsulation.

A DCCP implementation endpoint may simultaneously provide services over any or all combinations of DCCP-STD and/or DCCP-UDP encapsulations with IPv4 and/or IPv6.

The basic format of a DCCP-UDP packet is:

```
+-----------------------------------+
|     IP Header (IPv4 or IPv6)      |  Variable length
+-----------------------------------+
|            UDP Header             |  8 bytes
+-----------------------------------+
|       DCCP Generic Header         |  12 or 16 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)
+-----------------------------------+
```

Section 3.8 describes usage of UDP ports. This includes implementation of a DCCP-UDP encapsulation service as a daemon that listens on a well-known port, allowing multiplexing of different DCCP applications over the same port.

3.1. The UDP Header

The format of the UDP header is specified in [RFC0768]:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |           Dest Port           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Length            |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

For DCCP-UDP, the fields are interpreted as follows:

Source and Dest(ination) Ports: 16 bits each

These fields identify the UDP ports on which the source and destination (respectively) of the packet are listening for incoming DCCP-UDP packets. The UDP port values do not 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 DCCP-UDP, the payload is a DCCP-UDP datagram).

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 UDP packet that carries DCCP-UDP.

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).

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          Source Port          |           Dest Port           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Data Offset  | CCVal | CsCov |           Checksum            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     |       |X|               |                               .
    | Res | Type  |=|   Reserved    |  Sequence Number (high bits)  .
    |     |       |1|               |                               .
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                  Sequence Number (low bits)                   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Generic DCCP Header with Long Sequence Numbers [RFC4340]

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          Source Port          |           Dest Port           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Data Offset  | CCVal | CsCov |           Checksum            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     |       |X|                                               |
    | Res | Type  |=|   Sequence Number (low bits)                  |
    |     |       |0|                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Generic DCCP Header with Short Sequence Numbers [RFC4340]

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

Section 3.8 describes how a DCCP-UDP implementation treats UDP and DCCP ports.

3.3. DCCP-UDP Checksum Procedures

DCCP-UDP employs a checksum at the UDP level and eliminates the use of the DCCP checksum. This approach was chosen to enable use of current NAT/NATP traversal methods developed for UDP. Such methods will generally be unaware whether DCCP is being encapsulated and hence do not update the inner checksum in the DCCP header. Standard DCCP requires protection of the DCCP header fields; this justifies any processing overhead incurred from calculating the UDP checksum.

In addition, UDP NAT traversal does not support partial checksums. Although this is still permitted end-to-end in the encapsulated DCCP datagram, links along the path will treat these as UDP packets and can not enable special partial checksum processing.

DCCP-UDP does not update or modify the operation of UDP. The UDP transport protocol is used in the following way:

For DCCP-UDP, the function of the DCCP Checksum field is performed by the UDP Checksum field. On transmission, the DCCP Checksum field SHOULD be set to zero. On receipt, the DCCP Checksum field MUST be ignored.

The UDP checksum MUST NOT be zero for a UDP packet that is sent using DCCP-UDP. If the received UDP Checksum field is zero, the packet MUST be dropped.

If the UDP Length field of a received packet is less than 20 (the UDP header length and minimum DCCP-UDP header length), the packet MUST be dropped.

If the UDP Checksum field, computed using standard UDP methods, is invalid, the received packet MUST be dropped.

If the UDP Length field in a received packet is less than the length of the UDP header plus the entire DCCP-UDP header (including the generic header and type-specific fields and options, if present) or if the UDP Length field is greater than the length of the packet from the beginning of the UDP header to the end of the packet, the packet MUST be dropped.

3.3.1. Partial Checksums and the Minimum Checksum Coverage Feature

This document requires the UDP checksum to be enabled when using DCCP-UDP. This checksum provides coverage of the entire encapsulated DCCP datagram.

DCCP-UDP supports the syntax of partial checksums. It also supports negotiation of the Minimum Checksum Coverage feature and settings of the CsCov field. However, the UDP Checksum field in DCCP-UDP always covers the entire DCCP datagram, and the DCCP checksum is ignored on receipt. An application that enables the partial checksums feature in the DCCP module will therefore experience a service that is functionally identical to using full DCCP checksum coverage. This is also the service that the application would have received if it had used a network path that did not provide optimised processing for DCCP partial checksums.

3.4. Network-Layer Options

A DCCP-UDP implementation MAY transfer network-layer options intended for DCCP to the network-layer header of the encapsulating UDP packet.

A DCCP-UDP endpoint that receives IP-options for the encapsulating UDP packet MAY forward these to the DCCP protocol module. If the endpoint forwards a specific network-layer option to the DCCP module, it MUST also forward all subsequent packets with this option. Consistent forwarding is essential for correct operation of many end-to-end options.

3.5. Explicit Congestion Notification

A DCCP-UDP endpoint SHOULD follow the procedures of DCCP-STD in [RFC4340], Section 12 by setting the Explicit Congestion Notification (ECN) in the IP headers of outgoing packets and examining the values received in the ECN fields of incoming IP packets, relaying any packet markings to the DCCP module.

Implementations that do not support ECN MUST follow the procedures of DCCP-STD in [RFC4340], Section 12.1 with regard to implementations that are not ECN capable.

3.6. ICMP Handling for Messages Relating to DCCP-UDP

To allow ICMP messages to be demultiplexed by the receiving endpoint, part of the original packet that resulted in the message is included in the payload of the ICMP error message. The receiving endpoint can therefore use this information to associate the ICMP error with the transport protocol instance that resulted in the ICMP message. When DCCP-UDP is used, the error message and the payload of the ICMP error message relate to the UDP transport.

DCCP-UDP endpoints SHOULD forward ICMP messages relating to a UDP packet that carries a DCCP-UDP to the DCCP module. This may imply translation of the payload of the ICMP message into a form that is recognised by the DCCP stack. [RFC5927] describes precautions that are desirable before TCP acts on the receipt of an ICMP message. Similar precautions are desirable prior to forwarding by DCCP-UDP to the DCCP module.

The minimal length ICMP error message generated in response to processing a UDP datagram only identifies the UDP source port and UDP destination port. This ICMP message does not carry sufficient information to discover the encapsulated DCCP Port values. A DCCP-

UDP endpoint that supports multiple DCCP connections over the same pair of UDP ports (see Section 3.8) may not therefore be able to associate an ICMP message with a unique DCCP-UDP connection.

3.7. Path Maximum Transmission Unit Discovery

DCCP-UDP implementations MUST follow DCCP-STD [RFC4340], Section 14 with regard to determining the maximum packet size and the use of Path Maximum Transmission Unit Discovery (PMTUD). This requires the processing of ICMP Destination Unreachable messages with a code that indicates that an unfragmentable packet was too large to be forwarded (a "Datagram Too Big" message), as defined in RFC 4340.

An effect of encapsulation is to incur additional datagram overhead. This will reduce the Maximum Packet Size (MPS) at the DCCP level.

3.8. Usage of the UDP Port by DCCP-UDP

A DCCP-UDP server (that is, an initially passive endpoint that wishes to receive DCCP-Request packets [RFC4340] over DCCP-UDP) listens for connections on one or more UDP ports. UDP port number 6511 has been allocated as the default listening UDP port for a DCCP-UDP server. Some NAT/NAPT topologies may require using a non-default listening port.

The purpose of this IANA-assigned port is for the operating system or a framework to receive and process DCCP-UDP datagrams for delivery to the DCCP module (e.g., to support a system-wide DCCP-UDP daemon serving multiple DCCP applications or a DCCP-UDP server placed behind a firewall).

An application-specific implementation SHOULD use an ephemeral port and advertise this port using outside means, e.g., SDP. This method of implementation SHOULD NOT use the IANA-assigned port to listen for incoming DCCP-UDP packets.

A DCCP-UDP client provides UDP source and destination ports as well as DCCP source and destination ports at connection initiation time. A client SHOULD ensure that each DCCP connection maps to a single DCCP-UDP connection by setting the UDP source port. Choosing a distinct UDP source port for each distinct DCCP connection ensures that UDP-based flow identifiers differ whenever DCCP-based flow identifiers differ. Specifically, two connections with different <source IP address, source DCCP port, destination IP address, destination DCCP port> DCCP 4-tuples will have different <source IP address, source UDP port, destination IP address, destination UDP port> UDP 4-tuples.

A DCCP-UDP server SHOULD accept datagrams from any UDP source port. There is a risk that the same DCCP source port number could be used by two endpoints, each behind a NAPT. A DCCP-UDP server MUST therefore demultiplex a DCCP-UDP flow using both the UDP source and destination port numbers and the encapsulated DCCP ports. This ensures than an active DCCP connection is uniquely identified by the 6-tuple <source IP address, source UDP port, source DCCP port, destination IP address, destination UDP port, destination DCCP port>. (The active state of a DCCP connection is defined in Section 3.8: a DCCP connection becomes active following transmission of a DCCP-Request and becomes inactive after sending a DCCP-Close.)

This demultiplexing at a DCCP-UDP endpoint occurs in two stages:

1. In the first stage, DCCP-UDP packets are demultiplexed using the UDP 4-tuple: <source IP address, source UDP port, destination IP address, destination UDP port>.

2. In the second stage, a receiving endpoint MUST ensure that two independent DCCP connections that were multiplexed to the same UDP 4-tuple are not associated with the same connection in the DCCP module. The endpoint therefore needs to keep state for the set of active DCCP-UDP endpoints using each combination of a UDP 4-tuple: <source IP address, source UDP port, destination IP address, destination UDP port>. Two DCCP endpoint methods are specified. A DCCP-UDP implementation MUST implement exactly one of these:

   * The DCCP server may accept only one active 6-tuple at any one time for a given UDP 4-tuple. In this method, DCCP-UDP packets that do not match an active 6-tuple MUST NOT be passed to the DCCP module and the DCCP Server SHOULD send a DCCP-Reset with Reset Code 12, "Encapsulated Port Reuse". An endpoint that receives a DCCP-Reset with this reset code will clear its connection state but MAY immediately try again using a different 4-tuple. This provides protection should the same UDP 4-tuple be re-used by multiple DCCP connections, ensuring that only one DCCP connection is established at one time.

   * The DCCP server may support multiple DCCP connections over the same UDP 4-tuple. In this method, the endpoint MUST then associate each 6-tuple with a single DCCP connection. If an endpoint is unable to demultiplex the 6-tuple (e.g., due to internal resource limits), it MUST discard DCCP-UDP packets that do not match an active 6-tuple instead of forwarding them to the DCCP module. The DCCP endpoint MAY send a DCCP-Reset

with Reset Code 12, "Encapsulated Port Reuse", indicating the connection has been closed but may be retried using a different UDP 4-tuple.

3.9. Service Codes and the DCCP Port Registry

This section clarifies the usage of DCCP Service Codes and the registration of server ports by DCCP-UDP. The section is not intended to update the procedures for allocating Service Codes or server ports.

There is one Service Code registry and one DCCP port registration that apply to all combinations of encapsulation and IP version. A DCCP Service Code specifies an application using DCCP regardless of the combination of DCCP encapsulation and IP version. An application may choose not to support some combinations of encapsulation and IP version, but its Service Code will remain registered for those combinations, and the Service Code must not be used by other applications. An application should not register different Service Codes for different combinations of encapsulation and IP version. [RFC5595] provides additional information about DCCP Service Codes.

Similarly, a DCCP port registration is applicable to all combinations of encapsulation and IP version. Again, an application may choose not to support some combinations of encapsulation and IP version on its registered DCCP port, although the port will remain registered for those combinations. Applications should not register different DCCP ports just for the purpose of using different combinations of encapsulation.

4. DCCP-UDP and Higher-Layer Protocols

The encapsulation of a higher-layer protocol within DCCP MUST be the same for both DCCP-STD and DCCP-UDP. Encapsulation of Datagram Transport Layer Security (DTLS) over DCCP is defined in [RFC5238] and RTP over DCCP is defined in [RFC5762]. This document therefore does not update these encapsulations when using DCCP-UDP.

5. Signalling the Use of DCCP-UDP

Applications often signal transport connection parameters through outside means, such as SDP. Applications that define such methods for DCCP MUST define how the DCCP encapsulation is chosen and MUST allow either encapsulation to be signalled. Where DCCP-STD and DCCP-UDP are both supported, DCCP-STD SHOULD be preferred.

The Session Description Protocol (SDP) [RFC4566] and the offer/answer model [RFC3264] can be used to negotiate DCCP sessions, and [RFC5762] defines SDP extensions for signalling the use of an RTP session running over DCCP connections. However, since [RFC5762] predates this document, it does not define a mechanism for signalling that the DCCP-UDP encapsulation is to be used. This section updates [RFC5762] to describe how SDP can be used to signal RTP sessions running over the DCCP-UDP encapsulation.

The new SDP support specified in this section is expected to be useful when the offering party is on the public Internet or in the same private addressing realm as the answering party. In this case, the DCCP-UDP server has a public address. The client may either have a public address or be behind a NAT/NAPT. This scenario has the potential to be an important use case. Some other NAT/NAPT topologies may result in the advertised port being unreachable via the NAT/NAPT.

5.1. Protocol Identification

SDP uses a media ("m=") line to convey details of the media format and transport protocol used. The ABNF syntax [RFC5234] of a media line for DCCP is as follows (from [RFC4566]):

```
media-field =          %x6d "=" media SP port ["/" integer]
                       SP proto 1*(SP fmt) CRLF
```

The proto field denotes the transport protocol used for the media, while the port indicates the transport port to which the media is sent, following [RFC5762]. This document defines the following five values of the proto field to indicate media transported using DCCP-UDP encapsulation:

UDP/DCCP

UDP/DCCP/RTP/AVP

UDP/DCCP/RTP/SAVP

UDP/DCCP/RTP/AVPF

UDP/DCCP/RTP/SAVPF

The "UDP/DCCP" protocol identifier is similar to the "DCCP" protocol identifier defined in [RFC5762] and denotes the DCCP transport protocol encapsulated in UDP, but not its upper-layer protocol.

The "UDP/DCCP/RTP/AVP" protocol identifier refers to RTP using the RTP Profile for Audio and Video Conferences with Minimal Control [RFC3551] running over the DCCP-UDP encapsulation.

The "UDP/DCCP/RTP/SAVP" protocol identifier refers to RTP using the Secure Real-time Transport Protocol [RFC3711] running over the DCCP-UDP encapsulation.

The "UDP/DCCP/RTP/AVPF" protocol identifier refers to RTP using the Extended RTP Profile for RTCP-based Feedback [RFC4585] running over the DCCP-UDP encapsulation.

The "UDP/DCCP/RTP/SAVPF" protocol identifier refers to RTP using the Extended Secure RTP Profile for RTCP-based Feedback [RFC5124] running over the DCCP-UDP encapsulation.

The fmt value in the "m=" line is used as described in [RFC5762].

The port number specified in the "m=" line indicates the UDP port that is used for the DCCP-UDP encapsulation service. The DCCP port number MUST be sent using an associated "a=dccp-port:" attribute, as described in Section 5.2.

The use of ports with DCCP-UDP encapsulation is described further in Section 3.8.

5.2. Signalling Encapsulated DCCP Ports

When using DCCP-UDP, the UDP port used for the encapsulation is signalled using the SDP "m=" line. The DCCP ports MUST NOT be included in the "m=" line but are instead signalled using a new SDP attribute ("dccp-port") defined according to the following ABNF:

```
dccp-port-attr = %x61 "=dccp-port:" dccp-port

dccp-port = 1*DIGIT
```

where DIGIT is as defined in [RFC5234]. This is a media-level attribute that is not subject to the charset attribute. The "a=dccp-port:" attribute MUST be included when the protocol identifiers described in Section 5.1 are used.

The use of ports with DCCP-UDP encapsulation is described further in Section 3.8.

- o If the "a=rtcp:" attribute [RFC3605] is used, then the signalled port is the DCCP port used for RTCP.
- o If the "a=rtcp-mux" attribute [RFC5761] is negotiated, then RTP and RTCP are multiplexed onto a single DCCP port; otherwise, separate DCCP ports are used for RTP and RTCP [RFC5762].

NOTE: In each case, only a single UDP port is used for the DCCP-UDP encapsulation.

o If the "a=rtcp-mux" attribute is not present, then the second of the two demultiplexing methods described in Section 3.8 MUST be implemented; otherwise, the second DCCP connection for the RTCP flow will be rejected. For this reason, using "a=rtcp-mux" is RECOMMENDED when using RTP over DCCP-UDP.

5.3. Connection Management

The "a=setup:" attribute is used in a manner compatible with [RFC5762], Section 5.3 to indicate which of the DCCP-UDP endpoints should initiate the DCCP-UDP connection establishment.

5.4. Negotiating the DCCP-UDP Encapsulation versus Native DCCP

An endpoint that supports both native DCCP and the DCCP-UDP encapsulation may wish to signal support for both options in an SDP offer, allowing the answering party the option of using native DCCP where possible, while falling back to the DCCP-UDP encapsulation otherwise.

An approach to doing this might be to include candidates for the DCCP-UDP encapsulation and native DCCP into an Interactive Connectivity Establishment (ICE) [RFC5245] exchange. Since DCCP is connection-oriented, these candidates would need to be encoded into ICE in a manner analogous to TCP candidates defined in [RFC6544]. Both active and passive candidates could be supported for native DCCP and DCCP-UDP encapsulation, as may DCCP simultaneous-open candidates [RFC5596]. In choosing local preference values, it may make sense to prefer DCCP-UDP over native DCCP in cases where low connection setup time is important and to prioritise native DCCP in cases where low overhead is preferred (on the assumption that DCCP-UDP is more likely to work through legacy NAT but has higher overhead). The details of this encoding into ICE are left for future study.

While ICE is appropriate for selecting basic use of DCCP-UDP versus DCCP-STD, it may not be appropriate for negotiating different RTP profiles with each transport encapsulation. The SDP Capability Negotiation framework [RFC5939] may be more suitable. Section 3.7 of RFC 5939 specifies how to provide attributes and transport protocols as capabilities and negotiate them using the framework. The details of the use of SDP Capability Negotiation with DCCP are left for future study.

5.5. Example of SDP Use

The example below shows an SDP offer, where an application signals support for DCCP-UDP:

```
v=0
o=alice 1129377363 1 IN IP4 192.0.2.47
s=-
c=IN IP4 192.0.2.47
t=0 0
m=video 50234 UDP/DCCP/RTP/AVP 99
a=rtpmap:99 h261/90000
a=dccp-service-code:SC=x52545056
a=dccp-port:5004
a=rtcp:5005
a=setup:passive
a=connection:new
```

The answering party at 192.0.2.128 receives this offer and responds with the following answer:

```
v=0
o=bob 1129377364 1 IN IP4 192.0.2.128
s=-
c=IN IP4 192.0.2.128
t=0 0
m=video 40123 UDP/DCCP/RTP/AVP 99
a=rtpmap:99 h261/90000
a=dccp-service-code:SC:RTPV
a=dccp-port:9
a=setup:active
a=connection:new
```

Note that the "m=" line in the answer includes the UDP port number of the encapsulation service. The DCCP service code is set to "RTPV", signalled using the "a=dccp-service-code" attribute [RFC5762]. The "a=dccp-port:" attribute in the answer is set to 9 (the discard port) in the usual manner for an active connection-oriented endpoint.

The answering party will then attempt to establish a DCCP-UDP connection to the offering party. The connection request will use an ephemeral DCCP source port and DCCP destination port 5004. The UDP packet encapsulating that request will have UDP source port 40123 and UDP destination port 50234.

6. Security Considerations

DCCP-UDP provides all of the security risk-mitigation measures present in DCCP-STD and also all of the security risks. It does not maintain additional state at the encapsulation layer.

The tunnel encapsulation recommends processing of ICMP messages received for packets sent using DCCP-UDP and translation to allow use by DCCP. [RFC5927] describes precautions that are desirable before TCP acts on receipt of ICMP messages. Similar precautions are desirable for endpoints processing ICMP for DCCP-UDP. The purpose of DCCP-UDP is to allow DCCP to pass through NAT/NAPT devices; therefore, it exposes DCCP to the risks associated with passing through NAT devices. It does not create any new risks with regard to NAT/NAPT devices.

DCCP-UDP may also allow DCCP applications to pass through existing firewall devices using rules for UDP, if the administrators of the devices so choose. A simple use may either allow all DCCP applications or allow none.

A firewall that interprets this specification could inspect the encapsulated DCCP header to filter based on the inner DCCP header information. Full control of DCCP connections by applications will require enhancements to firewalls, as discussed in [RFC4340] and related RFCs (e.g., [RFC5595]).

Datagram Transport Layer Security (DTLS) provides mechanisms that can be used to provide security protection for the encapsulated DCCP packets. DTLS may be used in two ways:

- o Individual DCCP connections may be protected in the same way that DTLS is used with native DCCP [RFC5595]. This does not encrypt the UDP transport header added by DCCP-UDP.

- o This specification also permits the use of DTLS with the UDP transport that encapsulates DCCP packets. When DTLS is used at the encapsulation layer, this protects the DCCP headers. This prevents the headers from being inspected or updated by network middleboxes (such as firewalls and NAPT). It also eliminates the need for a separate DTLS handshake for each DCCP connection.

7. IANA Considerations

IANA has made the allocations described in the following sections.

7.1. UDP Port Allocation

IANA has allocated a UDP port (6511) for the DCCP-UDP service. This port is allocated for use by a transport service rather than an application. In this case, the name of the transport should explicitly appear in the registry. Use of this port is defined in Section 3.8

7.2. DCCP Reset

IANA has assigned a new DCCP reset code (12) in the DCCP Reset Codes Registry, with the short description "Encapsulated Port Reuse". This code applies to all DCCP congestion control IDs. Use of this reset code is defined in Section 3.8. Section 5.6 of [RFC4340] defines three "Data" bytes that are carried by a DCCP Reset. For this reset code, these are defined as follows:

- o Data byte 1: The DCCP Packet Type of the DCCP datagram that resulted in the error message.
- o Data bytes 2 & 3: The encapsulated UDP source port from the DCCP-UDP datagram that triggered the ICMP message, in network order.

7.3. SDP Attribute Allocation

IANA has allocated the following new SDP attribute ("att-field"):

Contact name: DCCP Working Group

Attribute name: dccp-port

Long-form attribute name in English: Encapsulated DCCP Port

Type of attribute: Media level only

Subject to charset attribute? No

Purpose of the attribute: See this document, Section 5.1

Allowed attribute values: See this document, Section 5.1

8. Acknowledgments

This document was produced by the DCCP WG. The following individuals contributed during the working group last call: Andrew Lentvorski, Lloyd Wood, Pasi Sarolahti, Gerrit Renker, Eddie Kohler, and Dan Wing.

9. References

9.1. Normative References

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

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

[RFC3605] Huitema, C., "Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP)", RFC 3605, October 2003.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5762] Perkins, C., "RTP and the Datagram Congestion Control Protocol (DCCP)", RFC 5762, April 2010.

9.2. Informative References

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[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, July 2006.

[RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.

[RFC5238] Phelan, T., "Datagram Transport Layer Security (DTLS) over the Datagram Congestion Control Protocol (DCCP)", RFC 5238, May 2008.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5595] Fairhurst, G., "The Datagram Congestion Control Protocol (DCCP) Service Codes", RFC 5595, September 2009.

[RFC5596] Fairhurst, G., "Datagram Congestion Control Protocol (DCCP) Simultaneous-Open Technique to Facilitate NAT/ Middlebox Traversal", RFC 5596, September 2009.

[RFC5597] Denis-Courmont, R., "Network Address Translation (NAT) Behavioral Requirements for the Datagram Congestion Control Protocol", BCP 150, RFC 5597, September 2009.

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.

[RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, July 2010.

[RFC5939] Andreasen, F., "Session Description Protocol (SDP) Capability Negotiation", RFC 5939, September 2010.

[RFC6544] Rosenberg, J., Keranen, A., Lowekamp, B., and A. B. Roach, "TCP Candidates with Interactive Connectivity Establishment (ICE)", RFC 6544, March 2012.

Authors' Addresses

Tom Phelan
Sonus Networks
7 Technology Dr.
Westford, MA 01886
US

Phone: +1 978 614 8456
EMail: tphelan@sonusnet.com

Godred Fairhurst
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen, Scotland AB24 3UE
UK

EMail: gorry@erg.abdn.ac.uk
URI: http://www.erg.abdn.ac.uk

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow, Scotland G12 8QQ
UK

EMail: csp@csperkins.org
URI: http://csperkins.org/