An Update for DCCP Connection Establishment to Assist NAT & Firewall Traversal
draft-fairhurst-dccp-behave-update-01

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on May 17, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2007).
Abstract

This document proposes an update to the Datagram Congestion Control Protocol (DCCP), a connection-oriented and datagram-based transport protocol. The update assists DCCP applications that need to traverse middleboxes (e.g., Network Address Translators or firewalls), where peering endpoints need to nearly simultaneously initiate the connection to establish middlebox state required for communication.

Table of Contents

1. Introduction ................................................. 3
   1.1. Scope of this Document ............................... 3
   1.2. Scope of the Problem to be Tackled .................. 4
   1.3. Discussion of Traversal Techniques for Traditional NAT Devices .................................. 4
       1.3.1. Near Simultaneous-Open of Connections .......... 5
       1.3.2. Role Reversal .................................... 6
2. Solution for DCCP NAT Traversal .............................. 7
   2.1. Conventions and Terminology ............................ 7
   2.2. DCCP-Listen Packet Format ............................. 7
   2.3. Protocol Method ....................................... 8
       2.3.1. Protocol Method for DCCP-Server Endpoints .......... 8
       2.3.2. Protocol Method for DCCP-Client Endpoints .......... 9
       2.3.3. Processing by Routers and Middleboxes .......... 9
   2.4. Examples of Use ...................................... 9
   2.5. Backwards Compatibility with RFC 4340 ................ 10
3. Discussion of Design Decisions .............................. 12
   3.1. Generation of Listen Packets .......................... 12
   3.2. Repetition of Listen Packets .......................... 12
4. IANA Considerations ....................................... 13
5. Security Considerations .................................... 14
6. References ................................................ 15
   6.1. Normative References .................................. 15
   6.2. Informative References ................................ 15
Appendix A. Change Log ...................................... 17
Authors’ Addresses ........................................... 18
Intellectual Property and Copyright Statements .................. 19
1. Introduction

UDP Network Address Translator (NAT) traversal is well understood and widely implemented. NAT traversal for connection-oriented protocols (e.g. TCP) uses similar principles, but in some cases requires more complex and expensive solutions, such as dedicated relay servers.

DCCP [RFC4340] is both datagram-based and connection-oriented; and thus NAT traversal of DCCP faces the same problems as TCP NAT traversal, without being able to reap the benefits of datagram-based NAT traversal as in UDP. In addition, DCCP has the disadvantage of not being able to perform a simultaneous-open, a TCP-inherent characteristic which greatly simplifies TCP NAT traversal.

After discussing the problem space for DCCP NAT traversal, this document proposes a solution to natively support NAT traversal in DCCP. Among the discussed solutions, it requires the least changes and is based on an indicator message. A new type of DCCP packet is introduced, which is sent by the server and addressed to the expected client. Since the use of this packet type is tied to an optional condition, it facilitates robust and native support for NAT traversal, while at the same time leaving the standard operational procedure of DCCP untouched.

1.1. Scope of this Document

The solution proposed by this document assists in connection establishment when one or both peers are located behind a middlebox. While the solution is specifically targeted at NAT traversal, due to the similarity of involved principles, it may also be of similar use to the traversal of other types of middlebox, such as firewalls.

For the scope of this document we consider traditional (outbound) types of NAT as defined in [RFC2663] and further discussed in [RFC3022]. We consider NAT traversal to involve one or more NAT devices of this type in the path (i.e. hierarchies of nested NAT devices are possible). It is assumed that all NATs in the path between endpoints are BEHAVE-compliant [NAT-APP].

We consider NAT devices which provide a minimum of DCCP protocol support, in that layer-4 checksums can be updated to account for changes in the pseudo-header. Since this document tackles an inherent problem of DCCP NAT traversal, we do not specify any further requirements for DCCP support in NAT devices. These may be described by a separate document.

The method described by this document is relevant to both client/server and peer-to-peer applications, such as VoIP, file sharing, or
online gaming. This update assists connections that utilise prior
out-of-band signaling between the client and server (e.g. a well-
known rendezvous server, [RFC3261], [H.323]) to notify both endpoints
of the connection parameters ([RFC3235], [NAT-APP]).

1.2. Scope of the Problem to be Tackled

We refer to DCCP hosts located behind one or more NAT devices as
having "private" addresses, and to DCCP hosts located in the global
address realm as having "public" addresses.

We consider DCCP NAT traversal for the following scenarios:

1. Private client connects to public server.
2. Public server connects to private client.
3. Private client connects to private server.

A defining characteristic of traditional NAT devices [RFC3022] is
that private hosts can connect to external hosts, but not vice versa.
Hence the case (1) is always possible, whereas cases (2) and (3)
require NAT traversal techniques. In this document we do not
consider use of pre-configured static NAT address maps, which would
also allow outside hosts to connect to the private network in cases
(2) and (3).

A DCCP implementation conforming to [RFC4340] can perform NAT
traversal with the help of a public relay server. The update
described by this document allows simple NAT traversal, without
indirection via relay servers.

1.3. Discussion of Traversal Techniques for Traditional NAT Devices

This section is a brief review of some existing techniques to
establish connectivity across NAT devices, the basic idea being to
make peer-to-peer sessions look like "outbound" sessions on each NAT
device.

Often a rendezvous server, located in the public address realm, is
used to enable clients to discover their NAT topology and the
addresses of peers.

The term ‘hole punching’ was coined in [FSK05] and refers to creating
soft state in a traditional NAT device by initiating an outbound
connection. A well-behaved NAT can subsequently exploit this to
allow a reverse connection back to the host in the private address
realm.
The adaptation of the basic hole punching principle to TCP NAT traversal was introduced in section 4 of [FSK05] and relies on the simultaneous-open feature of TCP [RFC0793]. UDP and TCP hole punching use nearly the same protocol. The main difference lies in the way the clients perform connectivity checks after obtaining the address pairs from the server; whereas in UDP a single socket is sufficient, TCP clients require several sockets for the same address/port tuple:

- a passive socket to listen for connectivity tests from peers and
- multiple active connections from the same address to test connectivity of other peers.

The SYN sent out by client A to its peer B creates soft state in A’s NAT. At the same time, B tries to connect to A:

- if the SYN from B has left B’s NAT before the arrival of A’s SYN, both endpoints perform simultaneous-open (4-way handshake of SYN/SYNACK);
- otherwise A’s SYN may not enter B’s NAT, which leads to B performing a normal open (SYN_SENT => ESTABLISHED) and A performing a simultaneous-open (SYN_SENT => SYN_RCVD => ESTABLISHED).

In the latter case it is necessary that the NAT does not interfere with a RST segment (REQ-4 in [GBF+07]). The simultaneous-open solution is convenient due to its simplicity, and is thus a preferred mode of operation in the TCP extension for ICE (section 2 of [Ros07]).

We note that a simultaneous-open is not the only existing solution for TCP NAT traversal [GTF04], [GF05]; other approaches are reviewed in the next subsection.

The considerations in this section and the following subsections motivate a discussion as to whether and how the DCCP state machine can be enhanced to natively facilitate easier middlebox traversal.

1.3.1. Near Simultaneous-Open of Connections

Among the various TCP NAT traversal approaches, simultaneous-open suggests itself due to its simplicity [GF05], [NAT-APP]. A characteristic of simultaneous-open is that the clear distinction between client and server is erased: both sides enter through active (SYN_SENT) as well as passive (SYN_RCVD) states. This characteristic is in conflict with several ideas underlying DCCP, as a clear
separation between client and server has been one of the initial design decisions ([RFC4340], 4.6). Furthermore, several mechanisms implicitly rely on clearly-defined client/server roles:

- **Feature Negotiation:** with few exceptions, almost all of DCCP’s negotiable features use the "server-priority" reconciliation rule ([RFC4340], 6.3.1), whereby peers exchange their preference lists of feature values, and the server decides the outcome.

- **Closing States:** only servers may generate CloseReq packets (asking the peer to hold timewait state), while clients are only permitted to send Close or Reset packets to terminate a connection ([RFC4340], 8.3).

- **Service Codes:** servers may be associated with multiple service codes, while clients must be associated with exactly one ([RFC4340], 8.1.2).

- **Init Cookies:** may only be used by the server and on DCCP-Response packets ([RFC4340], 8.1.4).

The latter two points are not obstacles per se, but hinder the transition from a passive to an active socket. The assumption that "all DCCP hosts are clients", on the other hand, must be dismissed since it limits application programming. As a consequence, retrofitting simultaneous-open into DCCP does not seem a very sensible idea.

### 1.3.2. Role Reversal

After the simultaneous-open, one of the simplest TCP NAT traversal schemes involves role traversal ([Epp05] and [GTF04]), where a peer first opens an active connection for the single purpose of punching a hole in the firewall, and then reverts to a listening socket to accept incoming connections arriving via the new path.

This solution has several disadvantages for DCCP. First, a DCCP client would be required to change its role from initially ‘client’ to ‘server’. This makes common server processing difficult: it is not clear how to assign multiple service codes (this would have to be done after the transition); a similar issue arises with Init Cookies.

Further, the client must not yet have started feature negotiation, since its choice of initial options may rely on its role (i.e. if an endpoint knows it is the server, it can make a priori assumptions about the preference lists of features it is negotiating with the client, thereby enforcing a particular policy). We therefore do not recommend this approach for DCCP.
2. Solution for DCCP NAT Traversal

This section revises packet processing for a DCCP-client and Server, to assist in connection establishment when one or both peers are located behind a middlebox. The method does not employ role reversal; both endpoints start out with their designated roles, as specified in [RFC4340].

2.1. Conventions and Terminology

The document uses the terms and definitions provided in [RFC4340]. Familiarity with this specification is assumed. In particular, the following convention from ([RFC4340], 3.2) is used:

"Each DCCP connection runs between two hosts, which we often name DCCP A and DCCP B. Each connection is actively initiated by one of the hosts, which we call the client; the other, initially passive host is called the server."

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].

2.2. DCCP-Listen Packet Format

The document updates DCCP by adding a new packet type: DCCP-Listen, whose format is shown below

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |           Dest Port           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data Offset  | CCVal | CsCov |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Res | Type  |X|             Sequence Number                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

DCCP-Listen Packet Format

The Reserved (Res) field is specified by [RFC4340], and its successors. This document does not modify that definition.

The Type field has the value XX-IANA-assigned-XX, which indicates that this is a DCCP-Listen packet.

For a DCCP-Listen type of packet the following protocol fields MUST be set to zero:
Data Offset (the connection has not been established),
CCVal (the connection has not been established),
CsCov (there is no payload),
Sequence Number (no initial sequence number has been negotiated).

DCCP-Listen packets MUST set X to 1 (short sequence numbers are not permitted), and endpoints MUST ignore any such packets with X set to zero.

A DCCP-Listen Packet MUST NOT include any DCCP options (since this packet does not modify the receiver protocol state) and MUST NOT include application data.

2.3. Protocol Method

We use the term "session" as defined in ([RFC2663], 2.3): DCCP sessions are uniquely identified by the tuple of <source IP-address, source port, target IP-address, target port>.

DCCP, in addition, introduces service codes ([RFC4340], 8.1.2) which can be used to identify different services that may be connected using the same port.

We call the five-tuple <source IP-address, source port, service code, target IP-address, target port> a fully specified DCCP connection, and refer to endpoints that have been assigned all five parameters as a "fully specified endpoint". DCCP-Listen packets are only sent for the specific case of fully specified DCCP-server endpoints.

2.3.1. Protocol Method for DCCP-Server Endpoints

This document updates [RFC4340] for the case of fully specified DCCP-server endpoints.

Prior to connection setup, it is common for DCCP-server endpoints to not be fully specified: before the connection is established, a server usually sets the target address / target port to wildcard numbers (i.e. leaves these unspecified); the endpoint only becomes fully specified after performing the handshake with an incoming, new connection. For such cases, this document does not update the server behaviour described in [RFC4340], i.e. no DCCP-Listen packet is sent.

A fully specified DCCP-server endpoint permits only one specific client (matching the target IP-address / target port) to set up the connection. When such a server is in the LISTEN state, it SHOULD use
the method specified in this document, and send a single DCCP-Listen packet to the remote endpoint.

The server SHOULD treat ICMP error messages received in response to a DCCP-Listen packet as "soft errors", that do not abort a connection.

2.3.2.  Protocol Method for DCCP-Client Endpoints

A DCCP-client that implements [RFC4340] and receives a DCCP-Listen would issue a DCCP-Reset (as it would for all unknown types of packet).

This document updates [RFC4340], so that a DCCP-client in any state that receives a DCCP-Listen packet, MUST silently discard the packet.

The packet indicates only a willingness to accept a connection; if the client has already established a connection, this has no meaning. If the client is awaiting the response to a DCCP-Request, the client does not need to take specific action, and should continue to use the protocol method defined in [RFC4340].

Receipt of a DCCP-Listen packet by a server in the LISTEN state must necessarily lead to a Reset (Reset Code 7, "Connection Refused" is suggested). This is the expected behaviour for [RFC4340].

2.3.3.  Processing by Routers and Middleboxes

Routers and middleboxes both need to forward DCCP packets. This document does not specify the rules for forwarding these packets, but notes that DCCP-Listen packets do not require special treatment and should be forwarded end-to-end across the internet path. Middleboxes may utilise the connection information (address, port, Service Code) to establish local forwarding state.

2.4.  Examples of Use

In the examples below, DCCP A is the client and DCCP B is the server. NAT/Firewall device NA is placed before DCCP A, and NAT/Firewall device NB is placed before DCCP B.

Both NA and NB use a policy that permits DCCP packets to traverse the device for outgoing links, but only permit incoming DCCP packets when a previous packet has been sent for the same connection.

DCCP A and DCCP B decide to communicate using some out-of-band mechanism, whereupon the client and server are started. DCCP A initiates a connection by sending a DCCP-Request. DCCP B actively indicates its state by sending a Listen message. This fulfills the
requirement of punching a hole in NB so that DCCP A can retransmit the DCCP-Request and connect through it.

DCCP A                                       DCCP B
------               NA      NB              ------
+------------------+  +-+    +-+  +-----------------+
| (1) Initiation    |  | |    | |  |                 |
DCCP-Request -->  +--+-+---X| |  |                 |
DCCP-Request -->  +------------->
DCCP-Ack -->      +------------->
|                  |  | |    | |  |                 |
| DCCP-Data -->     +--+-+----+-+->|                 |
|                  |  | |    | |  |                 |
| (2) Data transfer |  | |    | |  |                 |
+------------------+  +-+    +-+  +-----------------+

Sequence of events when a client is started before the server

The diagram below reverses this sequencing:

DCCP A                                       DCCP B
------               NA      NB              ------
+------------------+  +-+    +-+  +-----------------+
| (1) Initiation    |  | |    | |  |                 |
DCCP-Request -->  +--+-+----+-+->|                 |
DCCP-Request -->  +------------->
DCCP-Ack -->      +------------->
|                  |  | |    | |  |                 |
| DCCP-Data -->     +--+-+----+-+> |                 |
|                  |  | |    | |  |                 |
| (2) Data transfer |  | |    | |  |                 |
+------------------+  +-+    +-+  +-----------------+

Sequence of events when a server is started before the client

2.5. Backwards Compatibility with RFC 4340

This proposal does not modify communication between a server conforming to [RFC4340] and a client that implements the update described in this document.

This proposal also does not modify communication for any server that implements a passive-open without fully binding the addresses, ports and service codes to be used.

A change in the protocol exchange does occur when a server implements
this update and binds to a client that implements [RFC4340].

If DCCP A and B were not behind NAT devices, the receipt of a DCCP-Listen packet at A by a server that implements [RFC4340] would lead to a DCCP-Reset (presumably Code 3, "No Connection", since Code 7, "Connection Refused" applies to DCCP-Requests only). This would abort the connection.

Since the outgoing DCCP-Listen packet is expected to open a hole in the firewall, the same conclusion is expected when using NATs; as in the examples presented in the previous section.

The authors do not however expect these issues to introduce practical deployment problems.
3. Discussion of Design Decisions

This section identifies a number of design decisions that were made in defining the current approach.

3.1. Generation of Listen Packets

Since DCCP-Listen packets solve a particular problem (NAT and/or firewall traversal), the generation of DCCP-Listen packets on passive sockets has been tied to a condition (i.e. the server is aware of the expected client connection request), so as to not interfere with the general case of "normal" DCCP connections.

In the TCP world, the analogue is a transition from LISTEN to SYN_SENT by virtue of sending data: "A fully specified passive call can be made active by the subsequent execution of a SEND" ([RFC0793], 3.8). Unlike TCP, this proposal does not perform a role-change from passive to active.

Like TCP, we require that DCCP-Listen packets are only sent by a DCCP-server when the endpoint is fully specified (Section 2.3). If this condition were considered too weak, it could be coupled with a socket option to trigger generation of DCCP-Listen packets on fully specified passive sockets.

3.2. Repetition of Listen Packets

Section 4.3 of [RFC4340] defines the LISTEN state as:

"Represents server sockets in the passive listening state. LISTEN and CLOSED are not associated with any particular DCCP connection."

This document revises this definition with regard to the protocol method described in Section 2.3. In the current revision of this document, this is represented as an additional transition (to the LISTEN state), but the authors also acknowledge the possibility of introducing a new state.

This state would allow a DCCP-Listen packet to be re-issued periodically, to refresh firewall state. This approach would add robustness to the proposed mechanism, but has the disadvantage that the DCCP-Listen packets could then contribute unwanted load in a mis-configured network and procedures may be needed to limit the rate and number of packets sent.
4. IANA Considerations

This document requires IANA action by allocation of a new Packet Type from the IANA Packet Types Registry. The Registry entry is to reference this document. This allocation requires IESG review and approval and standards-track IETF RFC publication.
5. Security Considerations

The method specified in this document exposes the state of a DCCP server that has been explicitly configured to accept a connection from a client. This requires prior out-of-band signaling between the client and server (e.g. via SIP) to establish this state. The method generates a packet addressed to the expected client.

This increases the vulnerability of the DCCP host by revealing which ports are in a passive LISTEN state to the expected client (this information is not encrypted, and therefore could be seen on the path to the client through the network). We do not believe this change significantly increases the complexity or vulnerability of a DCCP implementation that conforms to [RFC4340].

Reception of a DCCP-Listen packet would trigger a DCCP-Reset in [RFC4340], but is ignored by the method described in this protocol. This does not introduce new security concerns.
6. References

6.1. Normative References


6.2. Informative References


Appendix A. Change Log

This is the first public draft.
Authors’ Addresses

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

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

Gerrit Renker
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen AB24 3UE
Scotland

Email: gerrit@erg.abdn.ac.uk
URI: http://www.erg.abdn.ac.uk