Network Working Group
Request for Comments: 5596
Updates: 4340
Category: Standards Track

G. Fairhurst
University of Aberdeen

September 2009

Datagram Congestion Control Protocol (DCCP) Simultaneous-Open Technique to Facilitate NAT/Middlebox Traversal

Abstract

This document specifies an update to the Datagram Congestion Control Protocol (DCCP), a connection-oriented and datagram-based transport protocol. The update adds support for the DCCP-Listen packet. This assists DCCP applications to communicate through middleboxes (e.g., a Network Address Port Translator or a DCCP server behind a firewall), where peering endpoints need to initiate communication in a near-simultaneous manner to establish necessary middlebox state.

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Table of Contents

1. Introduction

The Datagram Congestion Control Protocol (DCCP) [RFC4340] is both datagram-based and connection-oriented. According to RFC 4340, DCCP servers establish connections by passively listening for incoming connection requests that are actively transmitted by DCCP clients. These asymmetric roles can cause problems when the server is 'inside' a middlebox, such as a Network Address Port Translation (NAPT), that only allows connection requests to be initiated from inside (e.g., due to port overloading) [RFC5597]. Host-based and network firewalls can also implement policies that lead to similar problems. This behaviour is currently the default for many firewalls.

UDP can support middlebox traversal using various techniques [RFC4787] that leverage the connectionless nature of UDP and are therefore not suitable for DCCP. TCP supports middlebox traversal through the use of its simultaneous-open procedure [RFC5382]. The concepts of the TCP solution are applicable to DCCP, but DCCP cannot simply reuse the same methods (see Appendix A).

After discussing the problem space for DCCP, this document specifies an update to the DCCP state machine to offer native support that allows a DCCP client to establish a connection to a DCCP server that is inside one or more middleboxes. This reduces dependence on external aids such as data relay servers [STUN] by explicitly supporting a widely used principle known as 'hole punching'.

The method requires only a minor change to the standard DCCP operational procedure. The use of a dedicated DCCP packet type ties usage to a specific condition, ensuring the method is inter-operable with hosts that do not implement this update or that choose to disable it (see Section 4).

1.1. Scope of This Document

This method is useful in scenarios when a DCCP server is located inside the perimeter controlled by a middlebox. It is relevant to both client/server and peer-to-peer applications, such as Voice over IP (VoIP), file sharing, or online gaming, and assists connections that utilise prior out-of-band signaling (e.g., via a well-known rendezvous server ([RFC3261], [H.323])) to notify both endpoints of the connection parameters ([RFC3235], [NAT-APP]).

1.2. DCCP NAT Traversal

The behavioural requirements for NAT devices supporting DCCP are described in [RFC5597]. A "traditional NAT" [RFC3022] that directly maps an IP address to a different IP address does not require the simultaneous-open technique described in this document.

The method is required when the DCCP server is positioned behind one or more NAPT devices in the path (hierarchies of nested NAPT devices are possible). This document refers to DCCP hosts located inside the perimeter controlled by one or more NAPT devices as having "private" addresses, and to DCCP hosts located in the global address realm as having "public" addresses.

DCCP NAT traversal is considered for the following scenarios:

1. Private client connects to public server.

2. Public client connects to private server.

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, case (1) is possible using the protocol defined in [RFC4340]. A pre-configured, static NAT address map would allow outside hosts to establish connections in cases (2) and (3).

A DCCP implementation conforming to [RFC4340] and a NAT device conforming to [RFC5597] would require a DCCP relay server to perform NAT traversal for cases (2) and (3).

This document describes a method to support cases (2) and (3) without the aid of a DCCP relay server. This method updates RFC 4340 and requires the DCCP server to discover the IP address and the DCCP port that correspond to the DCCP client. Such signaling may be performed out-of-band (e.g., using the Session Description Protocol (SDP) [RFC4566]).

1.3. Structure of This Document

For background information on existing NAT traversal techniques, please consult Appendix A.

The normative specification of the update is presented in Section 2. An informative discussion of underlying design decisions then follows in Section 3. Security considerations are provided in Section 4 and IANA considerations are provided in the concluding Section 5.

2. Procedure for Near-Simultaneous-Open

This section is normative and specifies the simultaneous-open technique for DCCP. It updates the connection-establishment procedures of [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 Section 3.2 of [RFC4340] 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. Protocol Method

The term "session" is used as defined in ([RFC2663], Section 2.3): DCCP sessions are uniquely identified by the 4-tuple of <source IP-address, source port, target IP-address, target port>.

DCCP, in addition, introduces Service Codes, which can be used to identify different services available via the same port [RFC5595].

2.2.1. DCCP-Listen Packet Format

This document adds a new DCCP packet type, DCCP-Listen, whose format is shown below.

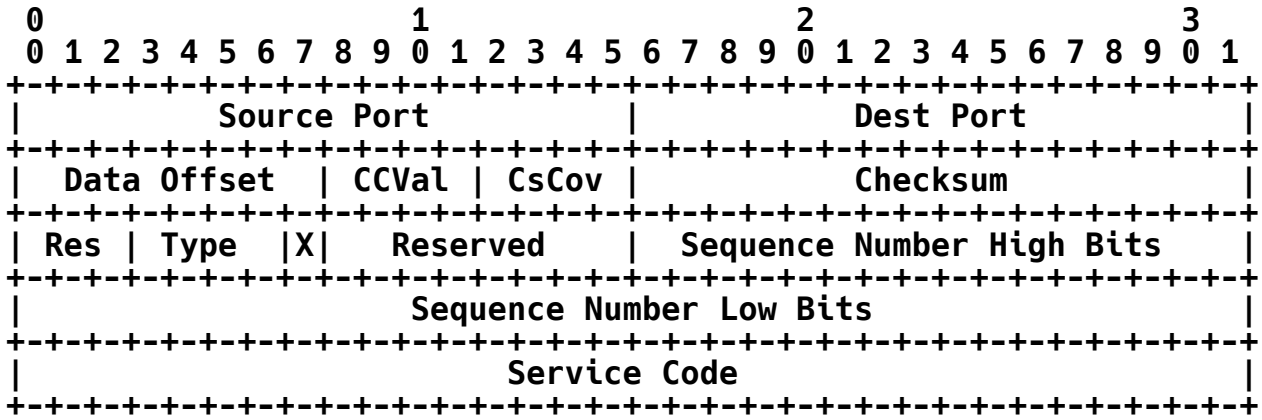


Figure 1: Format of a DCCP-Listen Packet

- The Source Port field indicates the port on which the DCCP server is listening for a connection from the IP address that appears as the destination IP address in the packet.

- The Destination Port field indicates the port selected by a DCCP client to identify the connection. In this technique, this value must be communicated out-of-band to the server.

- The value of X MUST be set to 1. A DCCP-Listen packet is sent before a connection is established; therefore, there is no way to negotiate use of short sequence numbers ([RFC4340], Section 5.1).

- The value of the Sequence Number field in a DCCP-Listen packet is not related to the DCCP sequence number used in normal DCCP messages (see Section 3 for a description of the use of the DCCP sequence number). Thus, for DCCP-Listen packets:

  * A DCCP server SHOULD set the high and low bits of the Sequence Number field to 0.

  * A DCCP client MUST ignore the value of the Sequence Number field.

  * Middleboxes MUST NOT interpret sequence numbers in DCCP-Listen packets.

- The Service Code field contains the Service Code value for which the server is listening for a connection (Section 8.1.2 of [RFC4340] and [RFC5595]). This value MUST correspond to a Service Code that the server is actually offering for a connection identified by the same source IP address and the same source port as that of the DCCP-Listen packet. Since the server may use multiple Service Codes, the specific value of the Service Code field needs to be communicated out-of-band, from client to server, prior to sending the DCCP-Listen packet, e.g., described using the Session Description Protocol (SDP) [RFC4566].

- At the time of writing, there are no known uses of header options ([RFC4340], Section 5.8) with a DCCP-Listen packet. Clients MUST ignore all options in received DCCP-Listen packets. Therefore, feature values cannot be negotiated using a DCCP-Listen packet.

- At the time of writing, there are no known uses of payload data with a DCCP-Listen packet. Since DCCP-Listen packets are issued before an actual connection is established, endpoints MUST ignore any payload data encountered in DCCP-Listen packets.

- The following protocol fields are required to have specific values:

  * Data Offset MUST have a value of five or more (i.e., at least 20 bytes).

  * CCVal MUST be zero (a connection has not been established).

  * CsCov MUST be zero (use of the CsCov feature cannot be negotiated).

  * Type has the value 10, assigned by IANA to denote a DCCP-Listen packet.

  * X MUST be 1 (the generic header must be used).

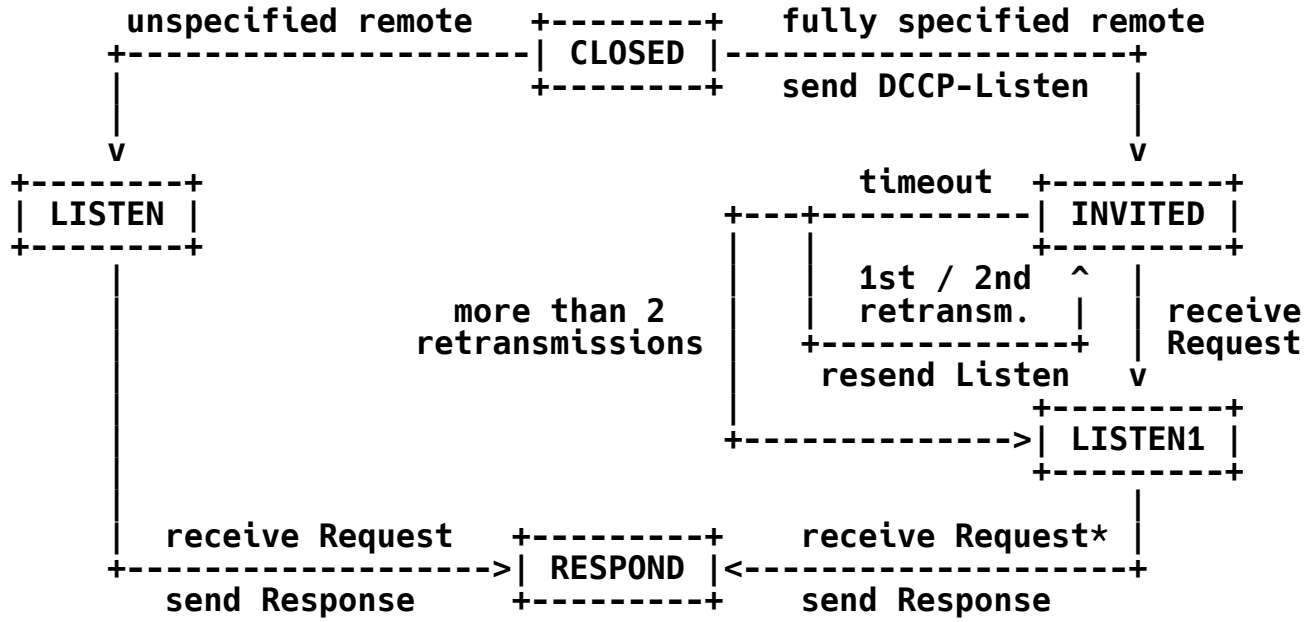
The remaining fields, including the "Res" and "Reserved" fields are specified by [RFC4340] and its successors. The interpretation of these fields is not modified by this document.

2.2.2. Protocol Method for DCCP Server Endpoints

This document updates Section 8.1 of [RFC4340] for the case of a fully specified DCCP server endpoint. The update modifies the way the server performs a passive-open.

Prior to connection setup, it is common for a DCCP server endpoint to not be fully specified: before the connection is established, a server usually specifies only the destination port and Service Code. (Sometimes the destination address is also specified.) This leaves the source address and source port unspecified. The endpoint only becomes fully specified after performing the handshake for an incoming connection. For such cases, this document does not update Section 8.4 of [RFC4340], i.e., the server adheres to the existing state transitions in the left half of Figure 2 (CLOSED => LISTEN => RESPOND).

A fully specified DCCP server endpoint permits exactly one client, identified by source IP-address:port, destination IP-address:port, plus a single Service Code, to set up the connection. Such a server SHOULD perform the actions and state transitions shown in the right half of Figure 2 and specified below.



* Note: The case of a server that responds to a DCCP-Request in the INVITED state, transitions to the LISTEN1 state, and then immediately transitions to the RESPOND state does not require reception of an additional DCCP-Request packet.

Figure 2: Updated State Transition Diagram for DCCP-Listen

This diagram introduces two additional DCCP server states in addition to those listed in Section 4.3 of [RFC4340]:

INVITED
: The INVITED state is associated with a specific DCCP connection and represents a fully specified server socket in the listening state that is generating DCCP-Listen packets towards the client.

LISTEN1
: The LISTEN1 state is associated with a specific DCCP connection and represents a fully specified server socket in the passive listening state that will not generate further DCCP-Listen packets towards the client.

A fully specified server endpoint performs a passive-open from the CLOSED state by inviting the remote client to connect. This is performed by sending a single DCCP-Listen packet to the specified remote IP-address:port, using the format specified in Section 2.2.1. The figure below provides pseudocode describing the packet processing in the INVITED state. This processing step follows Step 2 in Section 8.5 of [RFC4340]).

The INVITED state is, like LISTEN, a passive state, characterised by waiting in the absence of an established connection. If the server endpoint in the INVITED state receives a DCCP-Request that matches the set of bound ports and addresses, it transitions to the LISTEN1 state and then immediately transitions to the RESPOND state, where further processing resumes as specified in [RFC4340].

The server SHOULD repeat sending a DCCP-Listen packet while in the INVITED state, at a 200-millisecond interval with up to at most 2 repetitions (Section 3 discusses this choice of time interval). If the server is still in the INVITED state after a further period of 200ms following transmission of the third DCCP-Listen packet, it SHOULD progress to the LISTEN1 state.

Fully specified server endpoints SHOULD treat ICMP error messages received in response to a DCCP-Listen packet as "soft errors" that do not cause a state transition. Reception of an ICMP error message as a result of sending a DCCP-Listen packet does not necessarily indicate a failure of the following connection request, and therefore should not result in a server state change. This reaction to soft errors exploits the valuable feature of the Internet that, for many network failures, the network can be dynamically reconstructed without any disruption of the endpoints.

Server endpoints SHOULD ignore any incoming DCCP-Listen packets. A DCCP server in the LISTEN, INVITED, or LISTEN1 states MAY generate a

DCCP-Reset packet (Code 7, "Connection Refused") in response to a received DCCP-Listen packet. This DCCP-Reset packet is an indication that two servers are simultaneously awaiting connections on the same port.

Further details on the design rationale are discussed in Section 3.

The figure below provides pseudocode describing the packet processing in the INVITED state. This processing step follows Step 2 in Section 8.5 of RFC 4340 [RFC4340].

```
 Step 2a: Process INVITED state
   If S.state == INVITED,
       /* State only entered for fully specified server endpoints */
       /* on entry S will have been set to a socket */
       If P.type == Request
          /* Exit INVITED state and continue to process the packet */
          S.state = LISTEN1
          Continue with S.state := LISTEN1
       Otherwise,
          If P.type == Listen
             /* The following line is optional */
             Generate Reset(Connection Refused)
             /* Otherwise, drop packet and return */
          Otherwise,
             Generate Reset(No Connection) unless P.type == Reset

 Step 2b: Process LISTEN1 state
   If S.state == LISTEN1,
       /* State only entered for fully specified server endpoints */
       /* Follows receipt of a Response packet */
       /* or sending third Listen packet (after timer expiry) */
       If P.type == Request,
          S.state = RESPOND
          Choose S.ISS (initial seqno) or set from Init Cookies
          Initialize S.GAR := S.ISS
          Set S.ISR, S.GSR, S.SWL, S.SWH from packet or Init Cookies
          Continue with S.state == RESPOND
          /* A Response packet will be generated in Step 11 */
       Otherwise,
          If P.type == Listen
             /* The following line is optional */
             Generate Reset(Connection Refused)
             /* Otherwise, drop packet and return */
          Otherwise,
             Generate Reset(No Connection) unless P.type == Reset
```

Figure 3: Updated DCCP Pseudocode for INVITED and LISTEN1 States

2.2.3. Protocol Method for DCCP Client Endpoints

This document updates Section 8.1.1 of [RFC4340] by adding the following rule for the reception of DCCP-Listen packets by clients:

Endpoints are required to ignore any header options or payload data encountered in DCCP-Listen packets (Section 2.2.1) and hence do not provide meaningful communication to a client. A client in any state MUST silently discard any received DCCP-Listen packet, unless it implements the optional procedure defined in the following section.

2.2.3.1. Optional Generation of Triggered Requests

This section describes an optional optimisation at the client that can allow the client to avoid having to wait for a timeout following a dropped DCCP-Request. The operation requires clients to respond to reception of DCCP-Listen packets when received in the REQUEST state. DCCP-Listen packets MUST be silently discarded in all other states.

A client implementing this optimisation MAY immediately perform a retransmission of a DCCP-Request following the reception of the first DCCP-Listen packet. The retransmission is performed in the same manner as a timeout in the REQUEST state [RFC4340]. A triggered retransmission SHOULD result in the client increasing the exponential-backoff timer interval.

Note that a path delay greater than 200ms will result in multiple DCCP-Listen packets arriving at the client before a DCCP-Response is received. Clients MUST therefore perform only one such retransmission for each DCCP connection. This requires maintaining local state (e.g., one flag per connection).

Implementors and users of this optional method need to be aware that host timing or path reordering can result in a server receiving two DCCP-Requests (i.e., the server sending one unnecessary packet). This would, in turn, trigger a client to send a second corresponding DCCP-Response (also unnecessary). These additional packets are not expected to modify or delay the DCCP open procedure [RFC4340].

Section 2.3.2 provides examples of the use of triggered retransmission.

2.2.4. Processing by Routers and Middleboxes

DCCP-Listen packets do not require special treatment and should thus be forwarded end-to-end across Internet paths, by routers and middleboxes alike.

Middleboxes may utilise the connection information (address, port, Service Code) to establish local forwarding state. The DCCP-Listen packet carries the necessary information to uniquely identify a DCCP session in combination with the source and destination addresses (found in the enclosing IP header), including the DCCP Service Code value [RFC5595]. The processing of the DCCP-Listen packet by NAT devices is specified in [RFC5597].

2.3. Examples of Use

In the examples below, DCCP A is the client and DCCP B is the server. A middlebox device (NAT/Firewall), NA, is placed before DCCP A, and another middlebox, 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 permits incoming DCCP packets when a previous packet has been sent out for the same connection.

In the figure below, DCCP A and DCCP B decide to communicate using an out-of-band mechanism (in this case, labelled SDP), whereupon the client and server are started. DCCP B actively indicates its listening state by sending a DCCP-Listen message. This fulfills the requirement of punching a hole in NB (also creating the binding to the external address and port). This message is dropped by NA since no hole exists there yet. DCCP A initiates a connection by entering the REQUEST state and sending a DCCP-Request. (It is assumed that if NA were a NAT device, then this would also result in a binding that maps the pinhole to the external address and port.) The DCCP-Request is received by DCCP B, via the binding at NB. DCCP B transmits the DCCP-Response and connects through the bindings now in place at NA and NB.

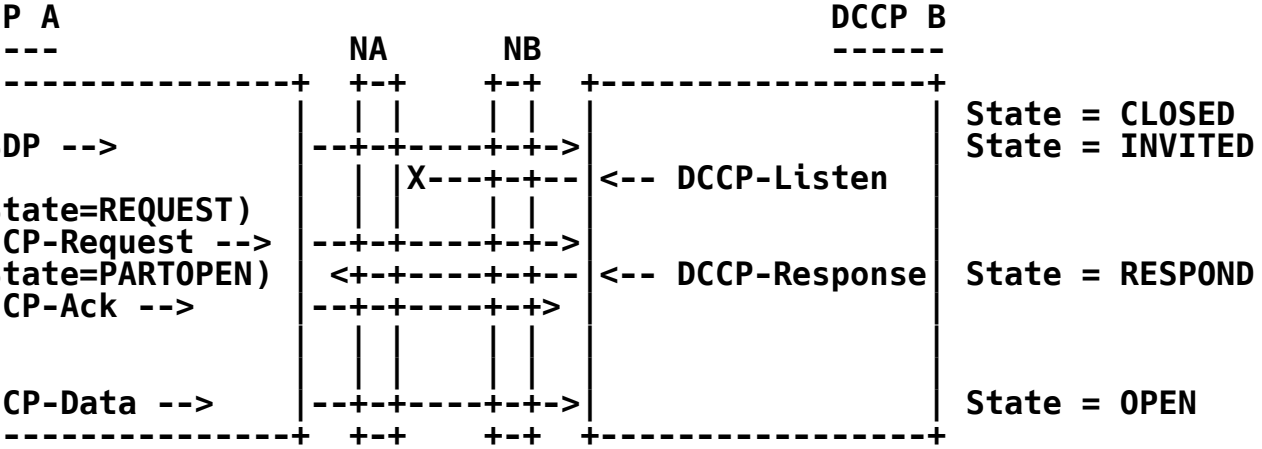


Figure 4: Event Sequence When the Server Is Started Before the Client

2.3.1. Repetition of DCCP-Listen

This section examines the effect of not receiving the DCCP-Request.

The figure below shows the sequence of packets where the DCCP server enters the INVITED state after reception of out-of-band signaling (e.g., SDP). The key timer operations at the client and server are respectively shown on the left and right of the diagram. It considers the case when the server does not receive a DCCP-Request within the first 600ms (often the request would be received within this interval).

The repetition of DCCP-Listen packets may be implemented using a timer. The timer is restarted with an interval of 200ms when sending each DCCP-Listen packet. It is cancelled when the server leaves the INVITED state. If the timer expires after the first and second transmission, it triggers a transmission of another DCCP-Listen packet. If it expires after sending the third DCCP-Listen packet, the server leaves the INVITED state to enter the LISTEN1 state (where it passively waits for a DCCP-Request).

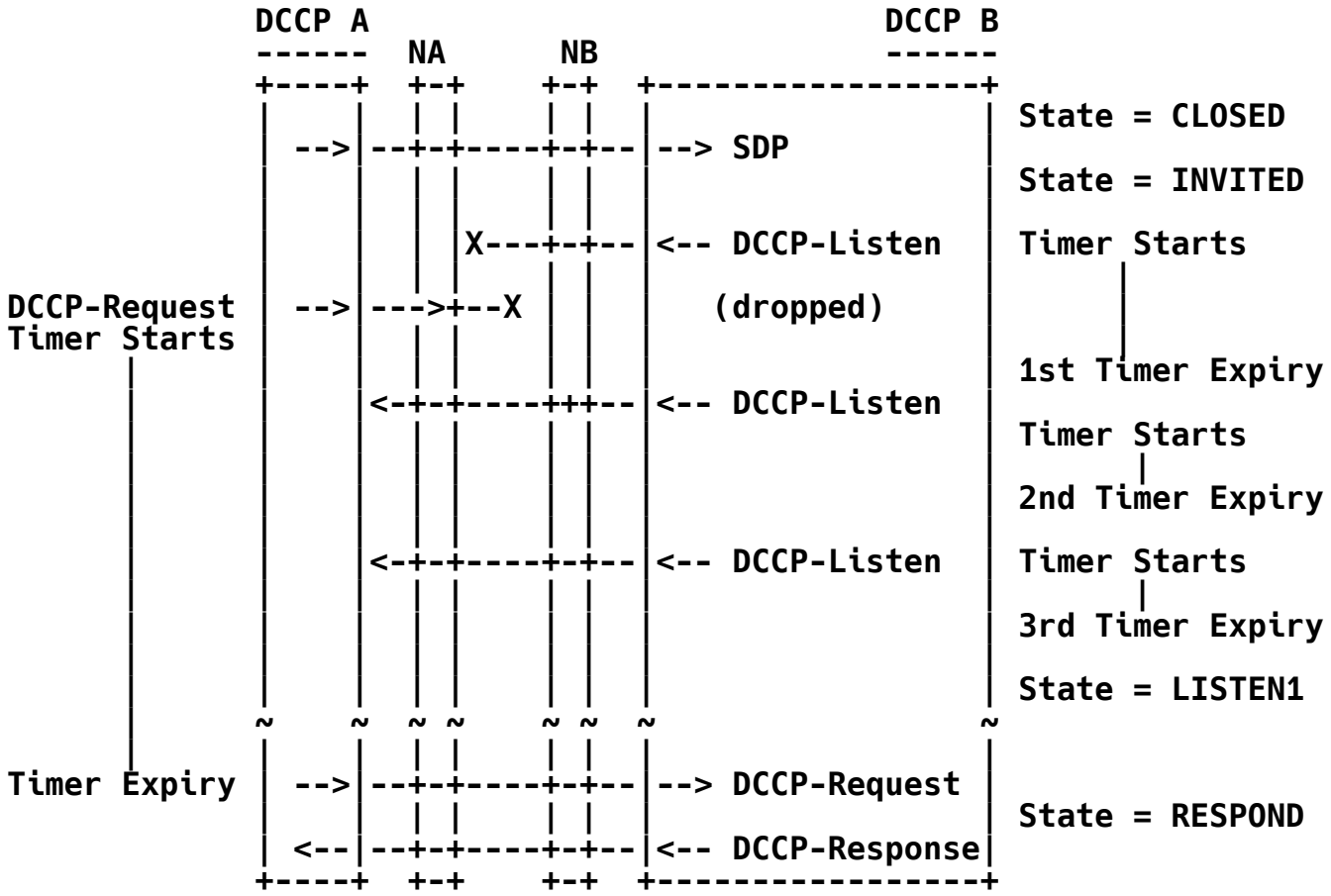


Figure 5: Repetition of the DCCP-Listen Packet

2.3.2. Optional Triggered Retransmission of DCCP-Request

The following figure illustrates a triggered retransmission. In this figure, the first DCCP-Listen is assumed to be lost in the network (e.g., does not open a pinhole at NB). A later DCCP-Request is also not received (perhaps as a side effect of the first loss). After 200ms, the DCCP-Listen packet is retransmitted and correctly received. This triggers the retransmission of the DCCP-Request, which, when received, results in a corresponding DCCP-Response.

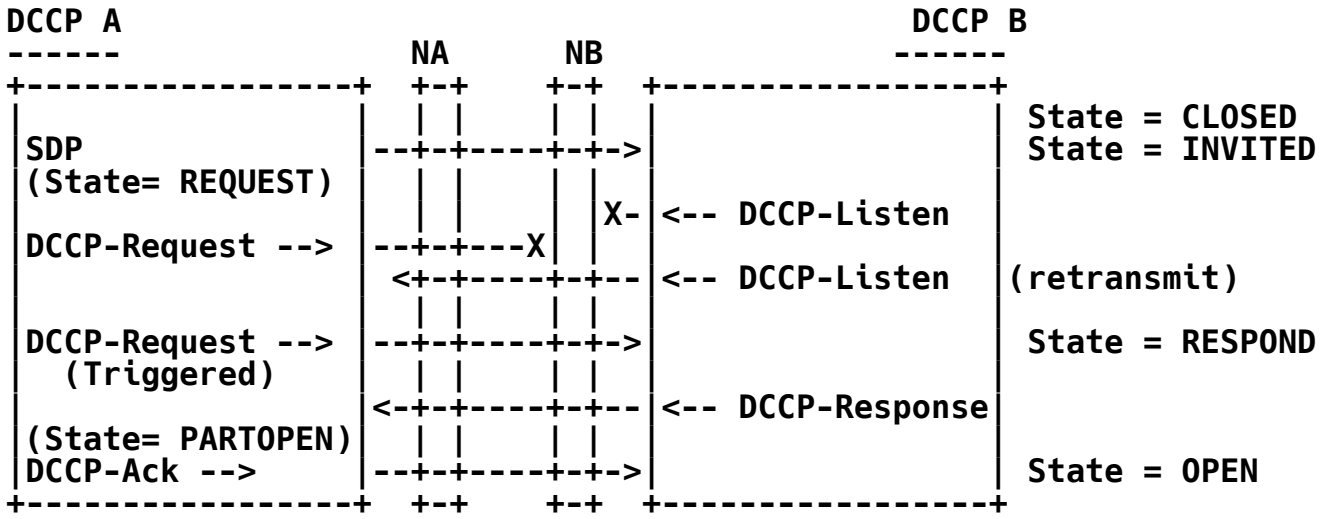


Figure 6: Example Showing a Triggered DCCP-Request

The figure below illustrates the sequence of packets exchanged when a DCCP-Listen and DCCP-Request are processed out of order. Reception of the DCCP-Listen packet by the client triggers retransmission of the DCCP-Request. The server responds to the first DCCP-Request and enters the RESPOND state. The server subsequently responds to the second DCCP-Request with another DCCP-Response, which is ignored by the client (already in the PARTOPEN state).

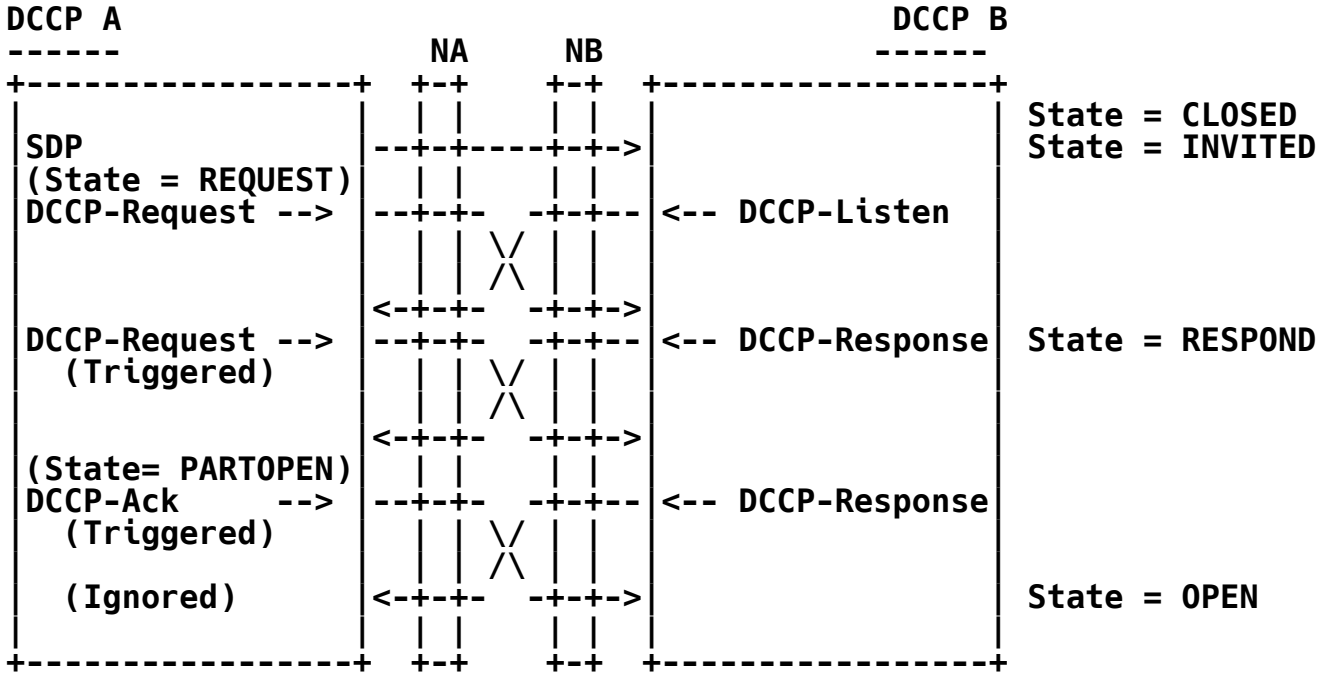


Figure 7: Example Showing an Unnecessary Triggered DCCP-Request

2.4. Backwards Compatibility with RFC 4340

No changes are required if a DCCP client conforming to this document communicates with a DCCP server conforming to [RFC4340].

If a client implements only [RFC4340], an incoming DCCP-Listen packet would be ignored due to step 1 in Section 8.1 of [RFC4340], which at the same time also conforms to the behaviour specified by this document.

This document further does not modify communication for any DCCP server that implements a passive-open without fully binding the addresses, ports, and Service Codes to be used. The authors therefore do not expect practical deployment problems with existing, conformant DCCP implementations.

3. Discussion of Design Decisions

This is an informative section that reviews the rationale for the design of this method.

3.1. Rationale for a New Packet Type

The DCCP-Listen packet specified in Section 2.2.1 has the same format as the DCCP-Request packet ([RFC4340], Section 5.1), the only difference is in the value of the Type field. The usage, however, differs. The DCCP-Listen packet serves as an advisory message, not as part of the actual connection setup: sequence numbers have no meaning, and no payload can be communicated.

A DCCP-Request packet could, in theory, also have been used for the same purpose. The following arguments were against this:

The first problem was that of semantic overloading: the DCCP-Request defined in [RFC4340] serves a well-defined purpose, being the initial packet of the 3-way handshake. Additional use in the manner of a DCCP-Listen packet would have required DCCP processors to have two different processing paths: one where a DCCP-Request was interpreted as part of the initial handshake, and another where the same packet was interpreted as an indication of an intention to accept a new connection. This would complicate packet processing in hosts and, in particular, stateful middleboxes (which may have restricted computational resources).

The second problem is that a client receiving a DCCP-Request from a server could generate a DCCP-Reset packet if it had not yet entered the REQUEST state (step 7 in Section 8.5 of [RFC4340]). The method specified in this document lets client endpoints ignore DCCP-Listen packets. Adding a similar rule for the DCCP-Request packet would have been cumbersome: clients would not have been able to distinguish between a DCCP-Request packet meant to indicate an intention to accept a new connection and a genuinely erratic connection initiation.

The third problem is similar and refers to a client receiving the indication after having itself sent a (connection-initiation) DCCP-Request. Step 7 in Section 8.5 of [RFC4340] requires the client to reply to a DCCP-Request from the server with a DCCP-Sync packet. Since sequence numbers are ignored for this type of message, additional and complex processing would become necessary: either to ask the client not to respond to a DCCP-Request when the request is used as an indication, or to ask middleboxes and servers to ignore DCCP-Sync packets generated in response to DCCP-Request packets that are used as indications. Furthermore, since no initial sequence numbers have been negotiated at this stage, sending a DCCP-SyncAck would not be meaningful.

The use of a separate packet type therefore allows simpler and clearer processing.

3.1.1. Use of Sequence Numbers

Although the DCCP-Listen Sequence Number fields are ignored, they have been retained in the DCCP-Listen packet header to reuse the generic header format from Section 5.1 of [RFC4340].

DCCP assigns a random initial value to the sequence number when a DCCP connection is established [RFC4340]. However, a sender is required to set this value to zero for a DCCP-Listen packet. Both clients and middleboxes are also required to ignore this value.

The rationale for ignoring the Sequence Number fields of DCCP-Listen packets is that, at the time the DCCP-Listen is exchanged, the endpoints have not yet entered connection setup: the DCCP-Listen packet is sent while the server is still in the passive-open (INVITED) state, i.e., it has not yet allocated state, other than binding to the client's IP-address:port and Service Code.

3.2. Generation of Listen Packets

A DCCP server should by default permit generation of DCCP-Listen packets. Since DCCP-Listen packets solve a particular problem with NAT and/or firewall traversal, the generation of DCCP-Listen packets on passive sockets is tied to a condition (binding to a remote address and Service Code that are both known a priori) to ensure this does not interfere with the general case of "normal" DCCP connections (where client addresses are generally not known in advance).

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], Section 3.8). Unlike TCP, this update does not perform a role change from passive to active. Like TCP, DCCP-Listen packets are only sent by a DCCP-server when the endpoint is fully specified (Section 2.2).

3.3. Repetition of DCCP-Listen Packets

Repetition is a necessary requirement to increase robustness and the chance of successful connection establishment when a DCCP-Listen packet is lost due to congestion, link loss, or some other reason.

The decision to recommend a maximum number of 3 timeouts (2 repeated copies of the original DCCP-Listen packet) results from the following consideration: the repeated copies need to be spaced sufficiently far apart in time to avoid suffering from correlated loss. The interval of 200ms was chosen to accommodate a wide range of wireless and wired network paths.

Another constraint is given by the retransmission interval for the DCCP-Request ([RFC4340], Section 8.1.1). To establish state, intermediate systems need to receive a (retransmitted) DCCP-Listen packet before the DCCP-Request times out (1 second). With three timeouts, each spaced 200 milliseconds apart, the overall time is still below one second. The sum of 600 milliseconds is sufficiently large to provide for longer one-way delays, as is the case, e.g., on some wireless links.

The rationale behind transitioning to the LISTEN1 state after two repetitions is that other problems, independent of establishing middlebox state, may occur (such as delay or loss of the initial DCCP-Request). Any late or retransmitted DCCP-Request packets will then still reach the server, allowing connection establishment to successfully complete.

4. Security Considerations

General security considerations for DCCP are described in [RFC4340]. Security considerations for Service Codes are further described in [RFC5595].

The method specified in this document generates a DCCP-Listen packet addressed to a specific DCCP client. This exposes the state of a DCCP server that is in a passive listening state (i.e., waiting to accept a connection from a known client).

The exposed information is not encrypted and therefore could be seen on the network path to the DCCP client. An attacker on this return path could observe a DCCP-Listen packet and then exploit this by spoofing a packet (e.g., DCCP-Request or DCCP-Reset) with the IP addresses, DCCP ports, and Service Code that correspond to the values to be used for the connection. As in other on-path attacks, this could be used to inject data into a connection or to deny a connection request. A similar on-path attack is also possible for any DCCP connection, once the session is initiated by the client ([RFC4340], Section 18).

The DCCP-Listen packet is only sent in response to explicit, prior out-of-band signaling from a DCCP client to the DCCP server (e.g., SDP [RFC4566] information communicated via the Session Initiation Protocol [RFC3261]) and will normally directly precede a DCCP-Request sent by the client (which carries the same information).

This update does not significantly increase the complexity or vulnerability of a DCCP implementation that conforms to [RFC4340]. A DCCP server SHOULD therefore, by default, permit generation of DCCP-Listen packets. A server that wishes to prevent disclosing this

information MAY refrain from generating DCCP-Listen packets without impacting subsequent DCCP state transitions, but possibly inhibiting middlebox traversal.

The DCCP base specification in RFC 4340 defines an Init Cookie option, which lets a DCCP server avoid having to hold any state until the three-way, connection-setup handshake has completed. This specification enables an out-of-band mechanism that forces the server to hold state for a connection that has not yet been established. This is a change in the security profile of DCCP, although the impact is expected to be minimal and depends on the security features of the out-of-band mechanism (SIP SDP is one such mechanism that provides sufficient security features).

The method creates a new way for a client to set up a DCCP connection to a server using out-of-band data, transported over a signaling connection. If the signaling connection is not encrypted, an eavesdropper could see the client IP address and the port for the to-be-established DCCP connection, and generate a DCCP-Listen packet towards the client using its own server IP address and port. However, a client will only respond to a received DCCP-Listen packet if the server IP address and port match an existing DCCP connection that is in the REQUEST state (Section 2.3.2). The method therefore cannot be used to redirect the connection to a different server IP address.

5. IANA Considerations

The IANA registered a new packet type, "DCCP-Listen", in the IANA DCCP Packet Types Registry. The decimal value 10 has been assigned to this type. This registry entry references this document.

6. Acknowledgements

This update was originally co-authored by Dr. Gerrit Renker, University of Aberdeen, and the present author acknowledges his insight in design of the protocol mechanism and in careful review of the early revisions of the document text. Dan Wing assisted on issues relating to the use of NAT and NAPT.

7. References

7.1. Normative References

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

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

[RFC5595] Fairhurst, G., "The DCCP Service Code", RFC 5595, September 2009.

7.2. Informative References

[Epp05] Eppinger, J-L., "TCP Connections for P2P Apps: A Software Approach to Solving the NAT Problem", Carnegie Mellon University/ISRI Technical Report CMU-ISRI-05-104, January 2005.

[FSK05] Ford, B., Srisuresh, P., and D. Kegel, "Peer-to-Peer Communication Across Network Address Translators", Proceedings of USENIX-05, pages 179-192, 2005.

[GF05] Guha, S. and P. Francis, "Characterization and Measurement of TCP Traversal through NATs and Firewalls", Proceedings of Internet Measurement Conference (IMC-05), pages 199-211, 2005.

[GTF04] Guha, S., Takeda, Y., and P. Francis, "NUTSS: A SIP based approach to UDP and TCP connectivity", Proceedings of SIGCOMM-04 Workshops, Portland, OR, pages 43-48, 2004.

[H.323] ITU-T, "Packet-based Multimedia Communications Systems", Recommendation H.323, July 2003.

[ICE] Rosenberg, J., "TCP Candidates with Interactive Connectivity Establishment (ICE)", Work in Progress, July 2008.

[NAT-APP] Ford, B., "Application Design Guidelines for Traversal through Network Address Translators", Work in Progress, March 2007.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[RFC3235] Senie, D., "Network Address Translator (NAT)-Friendly Application Design Guidelines", RFC 3235, January 2002.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

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

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

[RFC5382] Guha, S., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, October 2008.

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

[STUN] Rosenberg, J., Mahy, R., and P. Matthews, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", Work in Progress, June 2009.

Appendix A. Discussion of Existing NAT Traversal Techniques

This appendix provides a brief review of existing techniques to establish connectivity across NAT devices, with the aim of providing background information. It first considers TCP NAT traversal based on simultaneous-open, and then discusses a second technique based on role reversal. Further information can be found in [GTF04] and [GF05].

A central idea shared by these techniques is 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.

UDP and TCP hole punching use nearly the same technique [RFC4787]. The adaptation of the basic UDP hole punching principle to TCP NAT traversal [RFC5382] was introduced in Section 4 of [FSK05] and relies on the simultaneous-open feature of TCP [RFC0793]. A further difference between UDP and TCP lies in the way the clients perform connectivity checks after obtaining suitable address pairs for connection establishment. Whereas in UDP a single socket is sufficient, TCP clients require several sockets for the same address and port tuple:

- a passive socket to listen for connectivity tests from peers, and
- multiple active connections from the same address to test reachability 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 [RFC5382]). The simultaneous-open solution is convenient due to its simplicity, and is thus a preferred mode of operation in the TCP extension for Interactive Connectivity Establishment (ICE) ([ICE], Section 2).

A.1. NAT Traversal Based on a Simultaneous-Request

Among the various TCP NAT traversal approaches, the one using a TCP simultaneous-open suggests itself as a candidate for DCCP due to its simplicity ([GF05], [NAT-APP]).

A characteristic of TCP simultaneous-open is that this erases the clear distinction between client and server: both sides enter through active (SYN_SENT) as well as passive (SYN_RCVD) states. This characteristic conflicts with the DCCP design decision to provide a clear separation between client and server functions ([RFC4340], Section 4.6).

In DCCP, 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], Section 6.3.1), whereby a peer exchanges its preference lists of feature values, and the server decides the outcome.

- Closing States: only a server may generate DCCP-CloseReq packets (asking the peer to hold timewait state), while a client is only permitted to send DCCP-Close or DCCP-Reset packets to terminate a connection ([RFC4340], Section 8.3).

- Service Codes [RFC5595]: a server may be associated with multiple Service Codes, while a client must be associated with exactly one ([RFC4340], Section 8.1.2).

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

The latter two points are not obstacles per se, but would have hindered the transition from a passive to an active socket. In DCCP, a DCCP-Request is only generated by a client. The assumption that "all DCCP hosts may be clients" was dismissed, since it would require undesirable changes to the state machine and would limit application programming. As a consequence, the retro-fitting of a TCP-style simultaneous-open into DCCP to allow simultaneous exchange of DCCP-Connect packets was not recommended.

A.2. Role Reversal

Another simple TCP NAT traversal scheme uses role traversal ([Epp05], [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, accepting connections that arrive via the new path.

This solution would have had several disadvantages if used with DCCP. First, a DCCP server would be required to change its role to temporarily become a 'client'. This would have required modification to the state machine -- in particular, the treatment of Service Codes and perhaps Init Cookies. Further, the method would have needed to follow feature negotiation, since an endpoint's choice of initial options can rely on its role (i.e., an endpoint that knows it is the server can make a priori assumptions about the preference lists of features it is negotiating with the client, thereby enforcing a particular policy). Finally, the server would have needed additional processing to ensure that the connection arriving at the listening socket matches the previously opened active connection.

This approach was therefore not recommend for DCCP.

Author's Address

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