Internet Engineering Task Force
INTERNET-DRAFT
draft-kohler-dccp-mobility-01b.ps
Expires: 30 July 2006

Eddie Kohler
UCLA
30 January 2006

Datagram Congestion Control Protocol Mobility and Multihoming

Status of this Memo

Status of this Memo

Abstract

This document lays out requirements and a preliminary design for transport-level mobility and multihoming support in the Datagram Congestion Control Protocol (DCCP) [DCCP].

Table of Contents

1. Introduction 3
2. Requirements 3
3. Protocol 4
3.1. Overview 4
3.2. Gencon Option 5
3.3. Initiate Gencon Message 5
3.4. Approve Gencon Message 6
3.5. Attach Gencon Message 7
3.6. Challenge Gencon Message 8
3.7. Confirm Gencon Message 9
3.8. Detach Gencon Message 9
3.9. Unexpected Options 10
4. Using Generalized Connections 11
5. Concerns 12
6. Security Considerations 12
7. IANA Considerations 12
Normative References 13
Informative References 13
Authors' Addresses 14
Full Copyright Statement 14
Intellectual Property 15

1. Introduction

The Datagram Congestion Control Protocol (DCCP) [DCCP] did not originally contain support for mobility or multihoming. Each DCCP connection was associated with exactly two network-level addresses over its lifetime, one per endpoint. At the very first DCCP BoF session, at the 51st IETF in London, we got some feedback criticizing this decision, so we added mobility and multihoming support as of November 2001. This decision was supported by the eventual DCCP working group charter:

> Prior to the final development of the protocol, the working group will investigate areas of functionality that should be integrated into DCCP because they are difficult or impossible to layer above it. These areas include security and multi-homing/mobility, at a minimum.

Thus began our quest for a mechanism that would support mobility and multihoming, at the DCCP level, with reasonable security and DoS-prevention, without using cryptography. (The DCCP working group's charter has been interpreted as preventing DCCP from including cryptography, even MD5 hashes.) DCCP's mobility support changed, often fundamentally, in every succeeding draft.

Unsurprisingly, we did not find a suitable mechanism, and I now believe no such mechanism exists. Even seemingly trivial multihoming mechanisms like SCTP's can cause security problems in practice [ANC04]. Mobility and multihoming have been removed from the main DCCP specification.

Unfortunately, mobility and multihoming support can't easily be implemented at a higher-level layer, and there are good arguments for supporting mobility and multihoming at the transport layer -- not least required interactions with congestion control. This document, therefore, presents one potential design for DCCP mobility and multihoming support. It relaxes one of DCCP mobility's original requirements by using cryptography.

2. Requirements

DCCP mobility and multihoming support should fulfill the following requirements and non-requirements.

- An endpoint does not need to announce a new address before moving to that address.

  RATIONALE: Mobile hosts may not know a new address until a move completes; and by that time, the old address may not be usable. Some multihomed hosts can know each of their addresses, but announcing addresses before using them does not prevent all attacks; see, for example, the "address squatting" attacks in [ANC04].

- Move requests must be safe against hijacking. Even attackers that can snoop on part or all of data traffic must not be able to move a connection. However, move requests need not be safe against man-in-the-middle attackers with control over which packets get delivered (such as routers).

  RATIONALE: Moving a connection is in some ways the worst possible attack: An attacker takes over a user's identity, without the user becoming aware of the attack or being able to stop the attack. We must prevent this. However, an endpoint with full control over the path could carry out this kind of attack even without mobility support. Therefore, we choose to allow a DCCP mobility mechanism to be vulnerable to attackers that can snoop a packet sent by the mobile host, then prevent that snooped packet from being delivered to the stationary host.

- Mobility must not create new, large opportunities for denial-of-service attacks.

- Endpoints must be able to move freely between different NAT domains using the mobility mechanism.
- Simultaneous moves need not be supported.
- Cryptography is allowed.

It is difficult, perhaps impossible, to fulfill both the NAT traversal and hijacking prevention requirements. Natural mechanisms for preventing hijacking, such as cryptographically signing the packet's network headers, fail in the presence of NATs, which change those headers. NATs essentially hijack connections by definition; we want to allow that, but prevent malicious hijacking. The solution below represents one attempt.

3. Protocol

3.1. Overview

DCCP mobility and multihoming support is based on generalized connections. A generalized connection groups one or more transport connections, called component connections, into a single application-level entity. To move addresses, a host attaches a new component connection, then detaches the old component connection. To implement multihoming, a host maintains multiple component connections with different endpoint addresses.

The multiple component connections that make up a generalized connection are treated independently at the transport level. They maintain independent sequence number spaces and congestion control state, for example. The application, however, sees only one socket, which corresponds to the generalized connection. Data received on any component connection is sent to that socket, and data sent via the socket may be transmitted over any component connection.

The first connection handshake in a generalized connection must register the intention to set up a generalized connection. The generalized connection's identifier is then agreed upon by the two endpoints (assuming they both support mobility). Thereafter, new component connections specify the intended generalized connection in their handshakes. A public-key cryptographic protocol prevents connection hijacking by passive attackers. However, attackers who can prevent packets from being delivered, or alter packets in flight, can hijack the connection, as is also possible in the absence of this extension.

```
(1) Connection initiation with preparation for generalized
    connections:
    A  -->  DCCP-Request with Initiate Gencon   -->  B
    A  <--  DCCP-Response with Approve Gencon   <--  B

(2) Adding a new connection to the generalized connection:
    A' -->  DCCP-Request with Attach Gencon     -->  B
    A' <--  DCCP-Response with Challenge Gencon <--  B
    A' -->  DCCP-Ack with Confirm Gencon        -->  B

(3) Removing a connection from the generalized connection:
    A  -->  DCCP-Sync with Detach Gencon        -->  B
    A  <--  DCCP-SyncAck                        <--  B
```

3.2. Gencon Option

The Gencon option, for Generalized Connection, is used to implement the subprotocols that create and update generalized connections. These subprotocols may contain messages that exceed the 253-byte option length limit. Therefore, all Gencon options on a packet are concatenated to specify a single Gencon message.

Gencon messages follow a common format, as follows.

```
+--------+------ ... ------+------ ... ------+-------- ...
| GCType |    Gencon ID    |  Component ID   | Payload
+--------+------ ... ------+------ ... ------+-------- ...
              (8 bytes)         (4 bytes)
```

Gencon Type (GCType): 8 bits

Defines the Gencon message type. Six types are currently defined, as follows:

```
Type    Meaning       Payload
----    -------       -------
  0     Initiate      Public Key
  1     Approve       Public Key
  2     Attach        Optional Nonce
  3     Challenge     Nonce and Optional Token
  4     Confirm       Token
  5     Detach        Token
6-255   Reserved
```

Table 1: Gencon Types

Gencon ID: 64 bits (8 bytes)

The Gencon ID uniquely identifies the generalized connection at both endpoints, and is used to identify new component connections for an existing generalized connection. To ensure uniqueness at both endpoints, the Gencon ID is defined in two parts: the client defines the upper 32 bits in its Initiate Gencon message, which is sent on the first DCCP-Request, and the server adds the lower 32 bits in its Approve Gencon message, which is sent on the corresponding DCCP-Response. Neither the upper nor the lower 32 bits of the Gencon ID may equal zero.

Component ID: 32 bits (4 bytes)

The Component ID identifies a component connection within a generalized connection. It MUST NOT equal zero.

3.3. Initiate Gencon Message

The client sends an Initiate Gencon message on the DCCP-Request packet that initiates a new generalized connection. This message contains half of the Gencon ID that will define the generalized connection, and the client's public key that will be used to validate later Gencon messages. The server's DCCP-Response packet will include an Approve Gencon message to complete the handshake.

The Initiate Gencon message has the following format:

```
+--------+----- ... -----+----- ... -----+
|00000000|   Gencon ID   | Component ID  | (continued)
+--------+----- ... -----+----- ... -----+
 GCType=0    (8 bytes)        (4 bytes)

    +--------+--------+--------+--------
    |     Key Type    |       Key ...
    +--------+--------+--------+--------
```

Gencon ID
The client specifies the upper 32 bits (4 bytes) of the generalized connection's Gencon ID in its Initiate Gencon message. These bits MUST NOT equal zero, while the lower 32 bits MUST equal zero. They also MUST NOT equal the upper 32 bits of the Gencon ID of any other generalized connection currently active at the server. Furthermore, they SHOULD be chosen so as to minimize duplication: that is, no recently-active generalized connection from this endpoint should have had the same upper 32 bits.

Component ID
This field is chosen by the client to identify this component connection. It MUST NOT equal zero, and its most significant bit MUST equal zero; thus, values 1-2147483647 are acceptable. One is perfectly reasonable choice for this first component connection.

Key Type: 16 bits
Defines the public-key cryptographic protocol to be used by other Gencon messages to validate future component connections. This value identifies the protocol, not its key length; the length of the Gencon message itself should be used to infer the key length.

Key: arbitrary length
The client's public key in the cryptosystem defined by Key Type. This public key is used only for this generalized connection, although an endpoint MAY reuse a public key on multiple generalized connections. The format of this field depends on the value of Key Type.

3.4. Approve Gencon Message

On receiving a DCCP-Request containing an acceptable Initiate Gencon message, the server responds with a DCCP-Response packet containing an Approve Gencon message. This serves three purposes: it acknowledges the creation of a generalized connection, it completes the specification of the Gencon ID, and it defines the server's public key.

The Approve message has the following format:

```
+--------+----- ... -----+----- ... -----+
|00000001|   Gencon ID   | Component ID  | (continued)
+--------+----- ... -----+----- ... -----+
 GCType=1    (8 bytes)        (4 bytes)

    +--------+--------+--------+--------
    |     Key Type    |       Key ...
    +--------+--------+--------+--------
```

Gencon ID
The upper 32 bits of this field MUST equal the value specified by the client in its Initiate Gencon message. The lower 32 bits are newly provided by the server, and MUST NOT equal zero. The result -- a 64-bit number, neither of whose halves equals zero -- is the connection's complete Gencon ID. The server MUST choose these lower 32 bits so that no other currently-active connection with the same endpoint has the same Gencon ID. Furthermore, it SHOULD choose these bits so as to minimize duplication, ensuring that new connections receive Gencon IDs that have not been seen before.

Component ID
MUST equal the value specified by the client in its Initiate message.

Key Type
MUST equal the value specified by the client in its Initiate message.

Key The server's public key in the cryptosystem defined by Key Type. See the comments above on the client's public key.

3.5. Attach Gencon Message

To add a component connection to a generalized connection, one of the endpoint hosts opens a new DCCP connection to the other in the conventional way -- that is, using a DCCP-Request packet. This DCCP-Request packet contains an Attach Gencon message with the generalized connection's Gencon ID. This initial message is unverified; a protocol consisting of a Challenge Gencon message, sent on the DCCP-Response, and a Confirm Gencon message, sent on the DCCP-Ack, verifies that the mobile endpoint host's private key approves of the move.

Note that the "client" of the new component connection need not be the same endpoint as the "client" of the original component connection. The original server is the endpoint that received the original DCCP-Request containing an Initiate Gencon message; to add a new component connection, it sends a DCCP-Request Gencon message to the original client, and is thus the client for the new component connection. Alternatively, the server could convince the client to open a new connection using application messages of some kind.

The Attach message has the following format:

```
+--------+----- ... -----+----- ... -----+----- ... -----+
|00000010|   Gencon ID   | Component ID  |     Nonce     |
+--------+----- ... -----+----- ... -----+----- ... -----+
 GCType=2    (8 bytes)       (4 bytes)       (8 bytes)
```

Gencon ID
The Gencon ID of the generalized connection.

Component ID
The Component ID of this new component connection. This value is chosen by the component client; it MUST NOT have been used for any previous successful component connection, and MUST NOT equal zero. The value of the most significant bit is set based on whether the component client (the endpoint sending this DCCP-Request) is the original client (the endpoint that sent the first DCCP-Request in the generalized connection), according to this table:

Component Client	Component ID MSB	Component ID Range
Original Client	0	1-2147483647
Original Server	1	2147483648-4294967295

This restriction ensures that the endpoints will not pick the same Component ID if they try to attach new component connections simultaneously.

Nonce: 64 bits

Nonce fields are used as challenges to verify that the other protocol endpoint knows the expected private key. The special value zero indicates the lack of a nonce. Nonce values MUST be chosen randomly, and MUST NOT be reused on the same generalized connection ID. (That is, given an endpoint and a Gencon ID, the multiset of Nonce values contained in Gencon messages with that endpoint and Gencon ID must contain no duplicates, except possibly for zero.) Thus, attackers should not be able to predict the next nonce an endpoint will use.

The component client MAY include an 8-byte Nonce in the Attach Gencon message. This expresses a desire to verify that the component server is valid, i.e. knows the expected private key.

3.6. Challenge Gencon Message

The DCCP-Response packet sent in response to a new component connection -- that is, to a DCCP-Request packet containing an Attach Gencon message -- MUST contain a Challenge Gencon message. This message is effectively an Init Cookie; the component server MUST NOT accept the new component connection until the Challenge is correctly confirmed with a Confirm Gencon message.

The Challenge Gencon message has the following format:

```
+--------+----- ... -----+----- ... -----+
|00000011|   Gencon ID   | Component ID  | (continued)
+--------+----- ... -----+----- ... -----+
 GCType=3    (8 bytes)       (4 bytes)

    +----- ... -----+--------+--------
    |     Nonce     |       Token ...
    +----- ... -----+--------+--------
         (8 bytes)        (optional)
```

Gencon ID

The Gencon ID of the generalized connection.

Component ID

The Component ID of the new component connection; MUST equal the Component ID from the corresponding Attach Gencon message.

Nonce

This Nonce is used to verify that the component client knows the relevant private key. It follows the restrictions described above, and MUST NOT be zero.

Token: variable length

If the Attach Gencon message contained a nonzero Nonce, then the component server MUST include a signed Token in its Challenge Gencon message. This Token will prove to the component client that the component server knows the relevant private key. To create a Token in response to a particular Nonce, an endpoint constructs the following 40-byte structure:

```
Byte Index  Contents
----------  --------
    0       Gencon Type of the message containing the Token
    1       zero
   2-7      Sequence Number of the packet that will contain
            the Token
   8-9      zero
  10-15     Acknowledgement Number of the packet that will
            contain the Token, which must equal the Sequence
            Number of the packet that contained the Nonce
  16-23     Gencon ID
  24-27     Component ID
  28-31     Component ID
  32-39     Nonce
```

This structure is then encrypted using the endpoint's private key: the encrypted result is the Token. The other endpoint can decrypt the Token using the corresponding public key; if the contents match, then some party that knows the relevant private key approved of those contents.

3.7. Confirm Gencon Message

After receiving the DCCP-Request containing a Challenge Gencon message, the component client MUST send a DCCP-Ack packet containing a Confirm Gencon message in response. This message contains a token that the component server will use to verify the client's identity.

The Confirm message has the following format:

```
+--------+----- ... -----+----- ... -----+--------+--------
|00000100|   Gencon ID   | Component ID  |      Token ...
+--------+----- ... -----+----- ... -----+--------+--------
 GCType=4    (8 bytes)       (4 bytes)       (variable)
```

Gencon ID
The Gencon ID of the generalized connection.

Component ID
The Component ID of the new component connection; MUST equal the Component ID from the corresponding Attach and Challenge Gencon messages.

Token
A Token created in response to the Nonce from the Challenge Gencon message.

3.8. Detach Gencon Message

A component connection may be closed in the usual way, via DCCP-CloseReq, DCCP-Close, and DCCP-Reset packets. Sometimes, however, an endpoint loses control of a generalized connection before getting a chance to close it; this may particularly happen in mobile hosts. The Detach Gencon message allows an endpoint to close a different component connection by Component ID. Detach Gencon messages MUST be sent on DCCP-Sync packets, which may be sent at any time during a connection. Since a packet can contain at most one Gencon message, only one component connection can be detached per packet.

The Detach message has the following format:

```
+--------+----- ... -----+----- ... -----+--------+--------
|00000101|   Gencon ID   | Component ID  |       Token ...
+--------+----- ... -----+----- ... -----+--------+--------
 GCType=5    (8 bytes)       (4 bytes)       (variable)
```

Gencon ID
The Gencon ID of the generalized connection.

Component ID
The Component ID of the component connection that should be closed. This MUST NOT equal the Component ID of the component connection on which the message appears.

Token
A Token, as above. The Nonce field used to construct the Token equals zero.

3.9. Unexpected Options

Endpoints MUST handle invalid, unexpected, and otherwise malformed Gencon options in the following way.

- A Gencon message is Mandatory when any of its component Gencon options is marked Mandatory. If a Mandatory Gencon message is "ignored" according to the following list, then the receiving endpoint MUST reset the connection using Reset Code 6, "Mandatory Failure", as described in [DCCP] (Section 5.8.2).
- Any Gencon message that does not meet basic formatting requirements, such as message length, MUST be ignored.
- Any Gencon message with unrecognized Gencon Type MUST be ignored.
- An Initiate Gencon message received on any packet other than a DCCP-Request MUST be ignored.
- An Initiate Gencon message whose Gencon ID and/or Component ID do not meet the specified requirements MUST be ignored.
- An Initiate Gencon message whose Key Type is not understood, or whose Key does not meet the requirements of the corresponding Key Type, MUST be ignored.
- An Approve Gencon message received on any packet other than a DCCP-Response MUST be ignored.
- An Approve Gencon message received on a DCCP-Response, where the corresponding DCCP-Request did not contain an Initiate Gencon message, MUST be ignored.
- An Approve Gencon message whose Gencon ID and/or Component ID do not meet the specified requirements, or whose Key Type does not equal the client's Key Type from its Initiate message, or whose Key does not meet the requirements of its Key Type, MUST be ignored.
- An Attach Gencon message received on any packet other than a DCCP-Request MUST be ignored.
- An Attach Gencon message whose Gencon ID does not correspond to a current connection, or whose Component ID is zero, or whose Component ID was used by a

previous successful component connection on this generalized connection, MUST be ignored. (A component connection is "successful" once a suitable Confirm Gencon message has been received.)

- A Challenge Gencon message received on any packet other than a DCCP-Response MUST be ignored.
- A Challenge Gencon message received on a DCCP-Response, where the corresponding DCCP-Request did not contain a Challenge Gencon message, MUST be ignored.
- A Challenge Gencon message whose Gencon ID and/or Component ID do not correspond to the Attach message's, or whose Nonce is zero, MUST be ignored.
- A Challenge Gencon message sent in response to an Attach message with a Nonce, whose Token is missing or invalid, MUST be ignored.
- A Confirm Gencon message received on any packet other than a packet whose Acknowledgement Number equals that of a DCCP-Response packet that contained a Challenge message, MUST be ignored.
- A Confirm Gencon message whose Gencon ID and/or Component ID do not correspond to the Attach message's, or whose Token is missing or invalid, MUST be ignored.
- A Detach Gencon message received on any packet other than a DCCP-Sync packet MUST be ignored.
- A Detach Gencon message whose Gencon ID does not correspond to the expected Gencon ID, or whose Component ID does not equal that of a currently active component connection, MUST be ignored.
- A Detach Gencon message whose Component ID equals that of the component connection on which the message appears MUST be ignored.
- A Detach Gencon message whose Token is missing or invalid MUST be ignored.

4. Using Generalized Connections

A client that expects to use multihoming or mobility, or that wants to support servers that use multihoming or mobility, SHOULD send an Initiate Gencon message on its DCCP-Request. If the server responds with a valid Approve Gencon message, then the connection is multihoming/mobility-enabled; and as a consequence, the application's data stream may be associated with more than one active connection. This section describes how that association is managed.

A generalized connection lasts as long as it has at least one associated component connection. When a generalized connection's last component connection is closed (moves to TIMEWAIT or CLOSED state), either through an explicit termination handshake or because of a timeout, the application-level connection MUST also be closed. APIs that report a reason for connection closure SHOULD use the reason associated with the last-closed component connection; if more than one connection closes at the same time, the choice is arbitrary.

When a generalized connection has multiple components, the endpoint must decide which connection to use to send data. Unless there is some external basis for choice, such as cost, the endpoint SHOULD send its data on the last connection on which it received data.

An endpoint SHOULD NOT use generalized connections simply to improve its throughput with parallel connections. There SHOULD be a substantive difference between the component connections, such as different network access technologies or failure dependencies. An endpoint that suspects that its partner is "cheating" with generalized connections MAY reset one or more component connections with Reset Code 11, "Aggression Penalty".

When multiple component connections are sending data, that data MUST be enqueued for the application in the order it is received. There is no way, other than simple delay, to enforce ordering among data received on different component connections. This is intentional; application-level sequence numbers are easily layered on top of DCCP, and lack of sequencing may discourage aggressive use of parallel component connections.

5. Concerns

Using multiple parallel connections to improve throughput.

6. Security Considerations

The base DCCP protocol is intended to protect against some classes of attacks, and explicitly declares itself vulnerable to other classes of attacks. Specifically,

> Attackers cannot hijack a DCCP connection (close the connection unexpectedly, or cause attacker data to be accepted by an endpoint as if it came from the sender) unless they can guess valid sequence numbers. Thus, as long as endpoints choose initial sequence numbers well, a DCCP attacker must snoop on data packets to get any reasonable probability of success. Sequence number validity checks provide this guarantee. [DCCP] (Section 18)

The mobility and multihoming support described in this document preserves this security model for existing connection features. Generalized connections, however, enlarge the possible semantics of DCCP interactions. This section describes the security consequences of the Gencon mechanism, as applied to those new semantics.

A "full hijacking" attack is defined as an attack where, using mobility and multihoming support, an attacker transparently adds itself as an endpoint to a generalized connection. Any attacker that resides on the path, and can control the delivery of messages, thus faking the ownership of an endpoint's IP address, can execute a full hijacking attack; this is true in unextended DCCP, and multihoming and mobility support does not change this fact. DCCP multihoming and mobility support aims to provide the following security guarantee for other attackers: An attacker cannot successfully execute a full hijacking attack unless it (1) can snoop the channel, and (2) knows an endpoint's private key.

Assume that the attacker does not know an endpoint's private key. Such an attacker cannot generate correct Tokens, and in particular, it cannot generate the Gencon Confirm Token required to complete a connection handshake. Thus, the only way for it to execute a full hijacking attack is by replaying a previous Gencon Confirm message. That message must have the same Gencon ID as the connection to be hijacked. The current connection must not have successfully used the replayed message's Component ID. The attacker must use the same Sequence Number as in the replayed message, and convince the other endpoint to use the same Sequence Number for its packets (ensuring that the Acknowledgement Number in the replayed message is correct). All of this is difficult, but not impossible. However, the attacker must also arrange for the other endpoint to use the same Nonce as in the previously replayed message, and the protocol explicitly forbids this.

7. IANA Considerations

IANA is requested to reserve the value 45 for the Gencon option from the DCCP Option Types registry, and to create a new registry for DCCP Gencon Types, populated initially with the values in Table 1.

Normative References

[DCCP] E. Kohler, M. Handley, and S. Floyd. Datagram Congestion Control Protocol, draft-ietf-dccp-spec-13.txt, work in progress, December 2005.

[RFC 793] J. Postel, editor. Transmission Control Protocol. RFC 793.

[RFC 1191] J. C. Mogul and S. E. Deering. Path MTU Discovery. RFC 1191.

[RFC 1750] D. Eastlake, S. Crocker, and J. Schiller. Randomness Recommendations for Security. RFC 1750.

[RFC 2119] S. Bradner. Key Words For Use in RFCs to Indicate Requirement Levels. RFC 2119.

[RFC 2460] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460.

[RFC 3168] K.K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168.

[RFC 3309] J. Stone, R. Stewart, and D. Otis. Stream Control Transmission Protocol (SCTP) Checksum Change. RFC 3309.

[RFC 3692] T. Narten. Assigning Experimental and Testing Numbers Considered Useful. RFC 3692.

[UDP-LITE] L-A. Larzon, M. Degermark, S. Pink, L-E. Jonsson (editor), and G. Fairhurst (editor). The UDP-Lite Protocol. draft-ietf-tsvwg-udp-lite-02.txt, work in progress, August 2003.

Informative References

[ANC04] Tuomas Aura, Pekka Nikander, and Gonzalo Camarillo. Effects of Mobility and Multihoming on Transport-Protocol Security. 2004 IEEE Symposium Security and Privacy.

[BB01] S.M. Bellovin and M. Blaze. Cryptographic Modes of Operation for the Internet. 2nd NIST Workshop on Modes of Operation, August 2001.

[BEL98] S.M. Bellovin. Cryptography and the Internet. Proc. CRYPTO ’98 (LNCS 1462), pp46-55, August, 1988.

[CCID 2 PROFILE] S. Floyd and E. Kohler. Profile for DCCP Congestion Control ID 2: TCP-like Congestion Control. draft-ietf-dccp-ccid2-05.txt, work in progress, February 2004.

[CCID 3 PROFILE] S. Floyd, E. Kohler, and J. Padhye. Profile for DCCP Congestion Control ID 3: TFRC Congestion Control. draft-ietf-dccp-ccid3-05.txt, work in progress, February 2004.

[LINK BCP] Phil Karn, editor. Advice for Internet Subnetwork Designers. draft-ietf-pilc-link-design-13.txt, work in progress, February 2003.

[M85] Robert T. Morris. A Weakness in the 4.2BSD Unix TCP/IP Software. Computer Science Technical Report 117, AT&T Bell Laboratories, Murray Hill, NJ, February 1985.

[PMTUD] Matt Mathis, John Heffner, and Kevin Lahey. Path MTU Discovery. draft-ietf-pmtud-method-00.txt, work in progress, October 2003.

[RFC 792] J. Postel, editor. Internet Control Message Protocol. RFC 792.

[RFC 1948] S. Bellovin. Defending Against Sequence Number Attacks. RFC 1948.

[RFC 2960] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol. RFC 2960.

[RFC 3124] H. Balakrishnan and S. Seshan. The Congestion Manager. RFC 3124.

[RFC 3360] S. Floyd. Inappropriate TCP Resets Considered Harmful. RFC 3360.

[RFC 3448] M. Handley, S. Floyd, J. Padhye, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification. RFC 3448.

[RFC 3517] E. Blanton, M. Allman, K. Fall, and L. Wang. A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP. RFC 3517.

[RFC 3540] N. Spring, D. Wetherall, and D. Ely. Robust Explicit Congestion Notification (ECN) Signaling with Nonces. RFC 3540.

[RFC 3550] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550.

[SB00] Alex C. Snoeren and Hari Balakrishnan. An End-to-End Approach to Host Mobility. Proc. 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '00), August 2000.

[SHHP00] Oliver Spatscheck, Jorgen S. Hansen, John H. Hartman, and Larry L. Peterson. Optimizing TCP Forwarder Performance. IEEE/ACM Transactions on Networking 8(2):146-157, April 2000.

[SYNCOOKIES] Daniel J. Bernstein. SYN Cookies. http://cr.yp.to/syncookies.html, as of July 2003.

Authors' Addresses

Eddie Kohler <kohler@cs.ucla.edu>
4531C Boelter Hall
UCLA Computer Science Department
Los Angeles, CA 90095
USA

Full Copyright Statement

Intellectual Property