Internet Draft
Intended status: Experimental
Expires: May 2010

D. Cavuto
AT&T
M. Apte
Juniper Networks
S. Jain
Juniper Networks
M. Murthy
Juniper Networks
November 10, 2009

DTCP: Dynamic Tasking Control Protocol
draft-cavuto-dtcp-03.txt

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and 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 10, 2010.

Copyright and License Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST, AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Abstract

Dynamic Tasking Control Protocol is a message-based interface by which an authorized client may connect to a server -- usually a network element (NE) or security policy enforcement point (PEP) -- and issue dynamic requests for data. These tasking requests contain, among other parameters, packet matching criteria that may apply to certain packets flowing through that network element. The primary intent of the tasking request is to instruct that network element to send copies of packets matching those criteria to a destination (usually via tunneling) for further inspection or other action. The protocol contains a security architecture to address client or server spoofing as well as replay prevention. The protocol assumes that multiple clients may simultaneously control a single server.

Table of Contents

1. Introduction

The Dynamic Tasking Control Protocol (DTCP) is a mechanism used to dynamically control network elements in the course of performing a security or other analysis on a transient network event.

Network Security personnel typically have little visibility into the very networks they are monitoring. Routers and switches have awkward mechanisms such as port mirroring and cFlowd to enable personnel some meager view into the traffic flowing through a device.

However, when a security incident does happen to be detected, the security analysis staff struggles to gain more insight as to the actual content of the incident, via inference from these tools. This is a time-consuming and cumbersome task.

cFlowd [9] and other aggregation mechanisms provide only session-level statistics about the event, and fail to provide any view into the actual packet data. In contrast, wholesale backhauling of port-mirrored data is often cumbersome (and expensive) to set up, since it requires pre-provisioned free bandwidth on wide-area links, and often additional network hardware to implement.

The intent of DTCP is to provide a simple mechanism by which a third-party device can interact with a network element or security policy-enforcement-point (PEP) that normally processes packetized network data, and in that interaction cause the PEP to take some action (usually copy) on a defined subset of that packet data to be forwarded for further inspection and analysis.

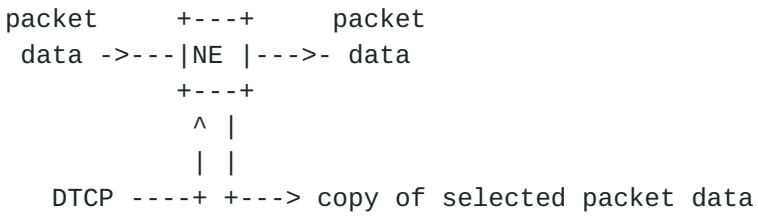

Figure 1 - DTCP interacting with a network element.

The Network Element (NE) or PEP may be a firewall or proxy server, or some other non-security-specific network element, such as a router or a switch. This is illustrated in Figure 1.

1.1. Operational Modes

The primary operation in DTCP is the specification of the filter criteria used to select or filter packets. DTCP is designed to work in an IPv4 environment, and accordingly all selection criteria are

chosen from IPv4 and higher-layer protocol definitions. Note that current DTCP syntax is limited to L3 and L4, but could be expanded to higher layers. Basic filter criteria definitions have semantic (if

not syntactic) similarity to well-known router access-control lists (ACLs) or firewall rulesets.

The primary operational mode of DTCP is the "copy" mode, whereby the controlled network element forwards the packet towards its intended destination, and also makes a copy of that packet, which it forwards towards a preconfigured collection and analysis center. In this mode, the original packet flow is not interrupted. DTCP makes no provisions for the potential performance impact on the network element when performing this function; obviously a negligible impact is most desirable.

DTCP also supports optional modes for purely redirecting the packet data (instead of making a copy of it), as well as blocking packet data. These modes, if implemented, can provide additional functionality for network security personnel, who may have decided that particular traffic is disallowed on the network and wishes to interrupt the selected flow of traffic.

Of critical distinction to DTCP is the basic paradigm that DTCP does NOT involve a "reprovisioning" or "reconfiguration" of the controlled device. DTCP is by its very nature transient; controlled devices should not attempt to maintain DTCP state in a non-volatile storage system.

1.2. Performance Considerations

It is envisioned that the controlling side of DTCP will be implemented by both human-interactive systems and automated systems. Since controlled Network Element MUST be able to respond to automated requests at a potentially high rate (due to floods or other attacks), the protocol implies a high performance requirement during the "criteria specification" phase of the interaction. In particular, the response time of the Network Element to respond to the DTCP request to monitor data is of considerable importance, as the traffic intended to be monitored may be short-lived.

While concrete performance requirements are outside the scope of this document, implementers are urged to focus performance on this part of the client-server interaction.

1.3. Conventions used in this document

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 RFC-2119 [1].

2. Definitions

The following sections define terms that have special significance within the DTCP context.

2.1. Server

The DTCP Server is the PEP or network element that controls the data of interest. The DTCP Server will be controlled in turn via DTCP. The Server is responsible for maintaining state of DTCP Client Requests, and forwarding data accordingly. Usually the DTCP Server will be implemented on a firewall or router (or an accessory device attached to one). The Server generally Responds to Requests, and can also initiate Asynchronous Notifications. One Server generally services more than one Client.

2.2. Client

The DTCP Client is an arbitrary host that initiates Requests to the Server via DTCP.

2.3. Control Source

A Control Source is the instantiation of one DTCP Client, with respect to a given Server. Each Control Source is preconfigured and pre-authorized on a given Server to be able to interact with it via DTCP. Control Sources may also receive Asynchronous Notifications. There may be many Control Sources configured on a given Server.

A Control Source MUST NOT be identified by IP address; rather, Control Sources are identified by user-configured character strings.

2.4. Content Destination

A Content Destination is the recipient of the extracted data, once it is forwarded by the server. Content Destinations are also preconfigured on the server.

A Content Destination MUST NOT be identified by IP address; rather, Content Destinations are identified by user-configured character strings.

2.5. Criteria

The Criteria is the list of matching conditions under which a packet is selected and acted upon by the server. Criteria are specified in requests from the client to the server, which maintains a list of active criteria for each client.

3. Overview of Operation

This section describes the basic interaction between the DTCP elements, as well as the protocol message flows.

3.1. Request-Response Paradigm

The basic model for DTCP is a request-response message exchange paradigm, where the server waits for messages on a specific UDP port from authorized Control Sources. When a request message arrives, the server processes the request, performs the necessary internal state change as per the request, and then sends a reply message.

Note that although DTCP is specified as a message-based protocol, it is designed and specified here to operate via single UDP/IP packets, for performance reasons. While it is certainly possible for DTCP to be operated over TCP/IP for reliable connections, such use is unexplored as yet, and any implementation-specific decisions made are unspecified herein. This document is written assuming that UDP will be used as the Layer-4 transport mechanism.

A DTCP Server MUST allocate at least ONE IP address and ONE UDP port for inbound connections from clients. Each DTCP Client MUST be statically configured with at least ONE IP address and ONE UDP port representing the server.

There is no mechanism defined that ensures proper configuration between DTCP Clients and servers for requests and responses.

In general, each request and each reply are a single UDP message, contained within a single IP packet. Since IP packets may be fragmented during delivery, each DTCP endpoint MUST be capable of IP fragment reassembly.

An IP packet containing a DTCP Request message from a client to a server MUST have the following attributes properly set:

- Destination IP Address MUST equal an IP address of a DTCP Server;
- IP Protocol MUST equal 17 (UDP);
- Destination UDP Port MUST equal a UDP port being listened on by the respective DTCP Server.

The DTCP Server MUST NOT rely on the source IP address or source UDP port of inbound request packets for any identification or authentication of the message.

An IP packet containing a DTCP Reply message from a server to a client MUST have the following attributes properly set:

- Source IP Address MUST equal the Destination IP Address of the IP packet containing the Request;

- Destination IP address MUST equal the Source IP Address of the IP packet containing the Request;
- IP Protocol MUST equal 17 (UDP);
- Destination UDP port MUST equal the Source UDP port of the UDP message containing the Request;
- Source UDP port MUST equal the Destination UDP port of the UDP message containing the Request.

There is no specific UDP port registered for DTCP; rather, each DTCP Server SHOULD permit the user to configure the port or set of ports on which it will listen for inbound DTCP requests. Additionally, a DTCP Server MAY choose to implement address or other filters on the source of inbound client requests; however, this is optional and implementation specific. (Recall that clients are identified by strings, NOT IP addresses.)

3.2. Asynchronous Notifications

Notifications are sent out by the DTCP Server to a set of statically preconfigured DTCP Clients who wish to receive notifications of asynchronous events. Such messages are sent to IP addresses that have been preconfigured therein.

A DTCP Client MAY provide a mechanism for accepting and processing Notifications. The DTCP Server MUST be preconfigured with an IP address and UDP port for each DTCP Client that wishes to receive Notifications.

There is no mechanism defined that ensures proper configuration between DTCP Clients and servers for Notifications.

An IP packet containing a DTCP Notification message from a server to a client MUST have the following attributes properly set:

- Destination IP address MUST equal the configured DTCP Client IP address;
- IP Protocol MUST equal 17 (UDP);
- Destination UDP port MUST equal the configured DTCP Client UDP port.

A future enhancement to this document may be to provide a mechanism for clients to dynamically self-register for notifications.

A DTCP Server SHOULD include a configuration parameter for each configured Control Source to indicate the Notification Version for that Control Source. If such a parameter is configured for a given Control Source, all Asynchronous Notifications sent from the DTCP Server to that Control Source MUST precisely match the Notification Version configured for that Control Source. Additionally, if such a

parameter is configured for a given Control Source, the DTCP Server MUST NOT send Asynchronous Notifications to that Control Source that do not exist in the DTCP specification indicated for that Control

Source. Finally, the DTCP Server MUST ensure that Asynchronous Notifications whose formats have been modified in newer versions of DTCP are properly formatted to meet the older DTCP specification version indicated for that Control Source.

Note that in general the DTCP Server SHOULD accept requests from a DTCP Client using a DTCP Version other than that specified in the Notification Version for that client.

If the DTCP Server is unable to meet these requirements, upon receiving a request from a DTCP Client with a mismatching version, it MUST return a a "505 DTCP Version not supported" error message *using the highest version supported by the DTCP Server*, and discontinue processing of that request.

It is recommended, however, that the DTCP Server reject such a state at configuration time rather than at run time.

3.3. Data Delivery Mechanism

Since the original packet IP header is not originally addressed to the intended Content Destination, each DTCP Server implementation MUST provide a mechanism for delivery of redirected data packets to appropriate Content Destinations. This explicitly includes IP checksums and IP TTL, as well as any higher-layer headers -- which SHOULD NOT be altered once captured -- but may not include MAC or lower-layer checksums.

DTCP explicitly does not specify the mechanism of data delivery to the Content Destination. Such a delivery mechanism is implementation-specific, and is outside the scope of this document.

As an example, Servers could utilize such technologies as VLAN tagging or IP tunneling to deliver entire unaltered data packets to Content Destinations.

4. Security Model

Since DTCP is, by design, a security protocol, it is imperative that it be resistant to malicious use.

4.1. No Information Exposure

DTCP was designed with the explicit paradigm that only information intentionally available to a given Control Source is ever exposed to that Control Source. For example: the existence of other Control Sources, or Content Destinations to which it has no access MUST NOT

be exposed to a given Control Source, e.g. via notifications or error messages. Also, the server MUST NOT respond to any message that fails

its security checks. This basic paradigm MUST be upheld in DTCP Server implementations.

4.2. Independence of Control Sources

DTCP may be implemented on network elements providing service to different customers. If each customer is allowed access to the DTCP Server, they MUST NOT be aware that another customer is using the DTCP Server. More specifically, neither customer's use (or misuse) of the DTCP Server can affect the other customer's use of it.

Limits on service-affecting actions that may be taken by a DTCP Client are outside the scope of this document.

4.3. Control Source to Content Destination Access Control

A DTCP Server SHOULD provide a mechanism by which each configured Control Source is granted access to one or more Content Destinations.

4.4. Per-Message Security Mechanisms

The primary motivation behind the per-message security mechanisms is to provide both message integrity as well as source authenticity. Additionally, providing insulation against replay-type attacks is also a motivation, though secondary.

DTCP currently provides no mechanism for confidentiality. If confidentiality is required, it is recommended that DTCP messages be sent via a secure transport.

> Note: Authentication failures, defined as a failure of these per-message security mechanisms, MUST NOT be reported to the DTCP Client. They SHOULD be logged on the DTCP server, and possibly acted upon by administration staff.

4.4.1. Sequence Number

Every message initiated by a DTCP Client MUST contain a sequence number. The request sequence number is an unsigned 64-bit whole number chosen arbitrarily by the client and maintained by the server persistently for each Control Source. All requests from a given Control Source MUST contain a monotonically-increasing sequence number. The sequence number for each successive request may increment by no more than 256. The stored last-valid sequence number shall only be updated upon receipt of a valid, authentic message.

A reply message to a valid request MUST contain the identical sequence number as the associated request.

Other than as specified below in "Sequence Number Negative Window", repetition of the last sequence number, or an invalid (non-

monotonically-increasing) sequence number, in an otherwise-valid message MUST result in the message being dropped and a security violation being logged, except when the sequence number wraps over zero due to bit-field-length constraints.

Rollover of the sequence number shall only be permitted when the MSB of the current sequence number is all-ones; otherwise this shall be considered a security violation. A rollover of the sequence number shall cause both an asynchronous notification message to be sent to any configured static address(es) for the respective Control Source as well as a log message to be generated.

It is suggested that clients do whatever possible to persistently store the current sequence number as there is no DTCP method by which to reset the current sequence number.

DTCP Servers SHOULD provide some mechanism for manually resetting the sequence number for a given client.

Additionally, DTCP Servers SHOULD implement a Negative Window feature as specified in the following section.

4.4.1.1. Sequence Number Negative Window

Under high load, a multithreaded DTCP client may send multiple requests (with properly incrementing sequence numbers) to the DTCP Server without waiting for each reply to come back individually. Because packets may be reordered through the network, they may arrive at the DTCP Server out of order.

For example, the DTCP client may send:

- Request 1 (Seq = 1)
- Request 2 (Seq = 2)
- Request 3 (Seq = 3)

But due to network reordering, the DTCP Server may receive:

- Request 1 (Seq = 1)
- Request 3 (Seq = 3)
- Request 2 (INVALID)

Unfortunately, the specification of the sequence number above will make Request 2 invalid, because once Request 3 is processed, the stored sequence number in the DTCP Server for that Client has been incremented to 3.

Therefore to correct this problem, the DTCP Server SHOULD include a "negative" window as well as the required "positive" window for

sequence numbers, and keep track of received sequence numbers within that negative window. However, in order to maintain the replay-protection afforded by the sequence number in the first place, any

DTCP Server implementing a negative window MUST also implement tracking of particular sequence numbers received within the union of both windows, and MUST NOT respond to any requests containing a sequence number already received.

If the received sequence number is in the negative window, the DTCP Server would simply store that sequence number as seen from that particular DTCP Client and process the packet. If the sequence number of the packet is in the positive window, the new positive and negative window would begin and end at this packet's sequence number respectively with the window sizes remaining the same. So a DTCP packet with sequence number within the negative window but that has not been seen (or anywhere within the positive window) is valid.

4.4.2. Hashing Message Authentication Code (HMAC)

A DTCP Server MUST store a statically-provisioned secret key for each configured client. This key is manually shared with each DTCP Client. Each request and response message MUST contain, as the last entry, a parameter called Authentication-Info, whose value is the HMAC algorithm specified in RFC-2104 [2] of the rest of the message payload (including the sequence number) generated using a SHA-1 [3] digest and the secret key. This digest is expressed in hexadecimal notation ([0-9a-f]), using 40 UTF-8 [4] characters to express the 160-bit SHA-1 hash.

```
Original Message:    text
Secret Key:          K
HMAC:                hash = SHA1HMAC(K, text)
New Message:         text + "Authentication-Info: " + hash
```

Figure 2 - Generating the message HMAC from the original message.

The shared secret key MUST NOT be sent in any DTCP message.

The precise algorithm, excerpted here from RFC-2104 for reference purposes (using SHA-1 as the hashing function H and byte length B=64) is as follows:

```
We define two fixed and different strings ipad and opad as follows
(the 'i' and 'o' are mnemonics for inner and outer):
                ipad = the byte 0x36 repeated B times
                opad = the byte 0x5C repeated B times.

To compute HMAC over the data `text' we perform
                  H(K XOR opad, H(K XOR ipad, text))
```

5. Application-Layer Message Formats

In general, the best source for the message formats is the Formal Syntax specified below. The following prose is provided for informational purposes and implementation guidelines. Where apparent syntactic conflicts exist, the Formal Syntax is defined to be correct.

DTCP messages are formatted in human-readable CRLF-delimited UTF-8 text format, using a mechanism similar to HTTP [5] or SIP [6]. Each message begins with an initial "command" line, followed by an optional series of parameter-value lines. Each token in the command line as well as each option line is separated by one or more white space characters. The entire message MUST end with two CRLFs.

The final token in any line MAY have whitespace before its terminating CRLF, but is not so required, and is not so reflected in the ABNF. DTCP servers SHOULD ignore extra whitespace between the final token and the terminating CRLF, but MUST return a Syntax Error otherwise.

Parameter names are specified in mixed case, but MUST be matched regardless of case.

Control characters or other unprintable characters in the parameter value may be indicated by a backslash (\) followed by precisely three digits indicating the UTF-8 value for the character, possibly including leading zeros. The backslash notation may be used to express any character, including whitespace. Backslash notation is explicitly forbidden from being interpreted as either an inter-token delimiter or an inter-parameter delimiter.

DTCP Clients and server MUST NOT rely upon the order of parameters within the DTCP message, since it is not guaranteed (other than the final "Authentication-Info" parameter as noted below).

Every DTCP message MUST contain the "Authentication-Info" parameter, and it MUST be the final parameter in the message. Any parameters in any DTCP message following the Authentication-Info parameter MUST be disregarded.

If a parameter appears multiple times, the behavior is undefined and not guaranteed; however, if a parameter does show up multiple times, the endpoint SHOULD take the value of the first occurrence and disregard any successive occurrences.

5.1. Request General Format

Each client-to-server message in DTCP begins with a single request

command line with the following format:

<command> <protocol-version-specifier> CRLF

The command line is followed by one or more parameter-value pairs, comprising the message body. The message is terminated by two CRLFs.

A DTCP request MUST contain the Sequence Number and the Control Source ID parameters.

5.2. Response General Format

Each server-to-client response message in DTCP shall begin with a single response line with the following format:

<protocol-version-specifier> <response-code> <response-text> CRLF

where the response-code is a three-digit numeric value, and the response-text is an arbitrary-length text string intended to be human-readable. The response line is followed by one or more parameter-value pairs comprising the message body. The message is terminated by two CRLFs.

Responses to successful requests MUST contain the response-code "200" and the response-text "OK".

A DTCP response MUST contain the Sequence Number parameter. A DTCP response MUST also contain the Timestamp parameter.

5.3. Notification General Format

Each server-to-client notification message in the control protocol shall begin with a single response line with the following format:

<protocol-version-specifier> <response-code> <response-text> CRLF

where the response-code is a three-digit numeric value, and the response-text is an arbitrary-length text string intended to be human-readable. The response line is followed by one or more parameter-value pairs comprising the message body. The message is terminated by two CRLFs.

A DTCP notification message MUST contain the Timestamp parameter.

5.4. Add Request

The Add request specifies a new filter criteria to be merged with the existing tasking list for a given Control Source and Content Destination (regardless of order added). Any missing parameters in the request will inferred to be a wildcard or "don't care". The Add request MAY be accompanied by one or more of the following required filter criterion parameters:

- Source IP address, range or IP + bitmask, or wildcard
- Destination IP address, range, or IP + bitmask, or wildcard
- IP Protocol or range, or wildcard

- Source Layer-4 Port or range, or wildcard (parameter only meaningful when IP protocol range includes protocols 6 or 17)
- Destination Layer-4 Port or range, or wildcard (parameter only meaningful when IP protocol range includes 6 or 17)
- ICMP Type or range, or wildcard (parameter only meaningful when IP protocol range includes protocol 1)
- ICMP Code or range, or wildcard (parameter only meaningful when IP protocol range includes protocol 1)

A wildcard in a given field implies that any value will match it (i.e. "don't care").

Additionally, the Add request MUST contain one or more of the following parameters:

- Timeout specified in seconds idle (maximum one day)
- Timeout specified in seconds total (maximum one day)
- Timeout specified in packets (maximum 64 bits)
- Timeout specified in bytes (maximum 64 bits)
- Flag: Static, which indicates that this criterion will never timeout and persist until explicitly deleted. All other timeouts shall be ignored if a STATIC flag is present.

Additionally, the Add request may contain one or more of the following parameters:

- Relative Priority (unsigned integer, minimum value 1) (optional, defaults to 1)
- Flag: Send Timeout Async (optional), which will cause the server to send a Asynchronous Notification when the criterion times out for any reason.
- Action (optional), which specifies whether the packet stream identified by the criterion will be a) copied to the Content Destination and also forwarded to its original intended destination ("Copy"), b) copied but not forwarded ("Redirect"), or c) not copied and not forwarded ("Block"). By default, Action is "Copy".

Finally, the Add request MUST contain the following control protocol parameters:

- Control Source Identifier
- Content Destination Identifier
- Sequence number (MUST be monotonically increasing for each request from a given Control Source)
- HMAC authenticator (MUST span message payload, plus secret key)

Although not explicitly expressed in the request, the DTCP Server MUST maintain the date/time of each filter criterion successfully

added. This time is the local DTCP Server time, either maintained independently by the server or synchronized via NTP.

5.4.1. Criteria Timeouts

Timeouts are required for each filter criterion added. These timeouts may be specified in any of four formats: seconds-idle, seconds-total, bytes, or packets. Any combination of these four timeouts may be used in a filter criterion as long as at least one is used.

Once a criterion is added, the timeouts will begin decrementing as appropriate. Only the timeouts that are specified in the request will be used for timing-out that criterion. When any active timeout is decremented to zero, the DTCP Server will automatically delete the filter criterion. For each Control Source, if enabled, when a criterion times-out and is deleted, timeout notifications will be sent to any statically-configured Notification Destination(s) associated with that Control Source.

A criterion may be added as STATIC. Any such criterion shall persist in the active state unless and until explicitly deleted or deleted due to congestion, provided the DTCP Server maintains its normal operational state. (See section 5.18 Congestion Notification for more information on congestion and timeouts.)

If all timeout values are zero and the criterion is not marked STATIC, the DTCP Server MUST return Error 433 (Improper Timeout Specification) and the criterion must not be added. For STATIC criteria, the DTCP Server MUST ignore the all timeout values.

If the server fails, STATIC rules may be lost. Any Control Source that uses STATIC criteria SHOULD attempt to ensure that such criteria are still up and active following any maintenance or failure event on the server.

5.5. Add Response

The response to a successful Add request will consist of the following parameters:

o Criteria ID

The Criteria ID will be persistent for the duration of that request, until it is removed explicitly by the client, or is removed implicitly by either timeout or some failure of the DTCP Server. The Criteria ID MUST uniquely identify that particular filter criterion for that particular Control Source (and be agnostic to the Content Destination).

DTCP Servers MUST ensure that generated Criteria ID are unique for all currently-active requests for a given Control Source.

Ideally, the Criteria ID SHOULD be globally unique across Control Sources, but this is not strictly required (since all requests will always be from a particular Control Source).

DTCP Servers SHOULD provide unique Criteria IDs for new requests, even if old ones have been deleted resulting in a fragmented ID space. This prevents race conditions that can cause inconsistent behavior e.g., a criterion specified in an Add request gets the same Criterion Id as a recently deleted criterion (deleted due to timeout), and before the delete notification could reach the Control Source, it sends out an explicit delete request for the old criterion, which when received by the DTCP Server would delete the recently added criterion, which is clearly undesirable.

This response MUST also include the following parameters:

- Timestamp
- Sequence number (MUST match the sequence number for the request)
- HMAC authenticator (MUST span message payload, plus secret key)

Responses to unsuccessful Add requests may take any of the following forms:

- Syntax Error
- Improper Filter Criterion Specification
- Unknown Destination Identifier
- Invalid Timeout Specification
- Improper Authentication (logged, but never sent to client)
- Invalid Sequence Number (logged, but never sent to client)
- Unknown Control Source Identifier (logged, but never sent to client)

5.6. Delete Request

The Delete request removes a particular filter criterion (or optionally all filter criteria) for the particular Control Source. The Delete request MUST take precisely one of the following parameters:

- Criteria ID or list of ranges of Criteria IDs
- Content Destination Identifier

Additionally, the Delete request may contain one or more of the following parameters:

- Flag: Static, which indicates that criteria added as STATIC should be deleted as well. (optional) If this flag is omitted, STATIC criteria MUST NOT be deleted.

If a single Criteria ID or list of ranges Criteria IDs is specified, the respective criterion/criteria is/are removed from the list of filter conditions that apply for that Control Source.

If a Content Destination Identifier is specified, all criteria are removed from the list of filter conditions to that particular Content Destination for that Control Source, except for STATIC criteria --

unless the STATIC flag is specified. (Note that any other criteria specified by any other Control Sources MUST remain unaffected.)

Additionally, the Delete request MUST contain the following parameters:

- Control Source Identifier
- Sequence number (MUST be monotonically increasing for each request from a given Control Source)
- HMAC authenticator (MUST span message payload, plus secret key)

5.7. Delete Response

The response to a successful Delete will consist of the following parameter:

- Number of Criteria Deleted

This parameter is an integer specifying the total number of filter criteria that were actually deleted. The number will be precisely 1 if a single, valid Criteria ID is supplied in the Delete request. If multiple valid Criteria IDs are supplied, the number of criteria actually deleted will be returned.

If any individual Criteria ID is invalid, the entire response will return an error and no action shall be taken by the server for any supplied Criteria ID. If a Content Destination Identifier is supplied, the number of criteria deleted shall be equal to the total number of active filter criteria from the requesting Control Source to that particular Content Destination. If no such criteria exist, the DTCP Server will return a successful delete response (with criteria deleted parameter set to zero).

When a range is specified, any existing criteria matching the Criteria ID in that range (inclusive) will be deleted and the true number of criteria deleted (including possibly zero) will be returned.

Trying to delete a STATIC criterion without the STATIC flag in the Delete request will result in that criterion NOT being deleted. Such a deletion attempt will return a success (non-error) response, including the actual number of criteria deleted (which may be zero).

This response MUST also include the following parameters:

- Timestamp
- Sequence number (MUST match the sequence number for the request)
- HMAC authenticator (MUST span message payload, plus secret key)

Responses to unsuccessful Delete requests may take any of the following forms:

- Syntax Error
- Unknown Criteria ID
- Unknown Destination Identifier
- Improper Authentication (logged, but never sent to client)
- Invalid Sequence Number (logged, but never sent to client)
- Unknown Control Source Identifier (logged, but never sent to client)

5.8. Refresh Request

The Refresh request updates the timeout for a particular filter criterion or set of filter criteria (or optionally all filter criteria) for the particular Control Source assigned to a particular Content Destination. This is used to maintain active criteria that are in danger of timing-out based on the original Add request. The updated timeout will replace the current remaining timeout, NOT be added to it. The Refresh request MUST take precisely one of the following parameters:

- Criteria ID or list of ranges of Criteria IDs
- Content Destination Identifier

Additionally, the Refresh request MUST contain one or more of the following parameters:

- Timeout specified in seconds total
- Timeout specified in seconds idle
- Timeout specified in packets
- Timeout specified in bytes

(Note that a Refresh request MUST NOT be used to make an existing filter criterion STATIC. A criterion MUST be added explicitly as STATIC in its original Add.)

Finally, the Refresh request MUST contain the following parameters:

- Control Source Identifier
- Sequence number (MUST be monotonically increasing for each request from a given Control Source)
- HMAC authenticator (MUST span message payload, plus secret key)

5.9. Refresh Response

The response to a successful Refresh will consist of the following parameters:

- Number of Criteria Refreshed

This parameter is an integer specifying the total number of filter

criteria that were actually updated. The number will be precisely 1 if a single, valid Criteria ID is supplied. If multiple valid Criteria ID are supplied, the number of criteria updated will be

returned, and that will equal the number of supplied Criteria IDs. If any Criteria ID is invalid, the entire response will return an error and no action shall be taken by the server for any supplied Criteria ID. If a Content Destination Identifier is supplied, the number of criteria updated shall be equal to the total number of active filter criteria from the requesting Control Source to that particular Content Destination, including zero (which will NOT return an error).

This response MUST also include the following parameters:

- Timestamp
- Sequence number (MUST match the sequence number for the request)
- HMAC authenticator (MUST span message payload, plus secret key)

Responses to unsuccessful Refresh requests may take any of the following forms:

- Syntax Error
- Invalid Timeout Specification
- Unknown Criteria ID
- Unknown Destination Identifier
- Improper Authentication (logged, but never sent to client)
- Invalid Sequence Number (logged, but never sent to client)
- Unknown Control Source Identifier (logged, but never sent to client)

5.10. List Request

The List request makes no change on the DTCP Server, but returns a list of all criteria that a particular Control Source has added. The Control Source may request this list on the basis of Content Destination, Criteria ID, or overall (for that particular Control Source). The List request takes the following parameters:

- Content Destination Identifier (optional)
- Criteria ID or List of ranges of Criteria IDs (optional)
- Flag: Statistics / Criteria / All

If neither of the optional parameters is included, the server MUST reply with the full set of criteria associated with that Control Source.

Additionally, the List request MUST contain the following parameters:

- Control Source Identifier
- Sequence number (MUST be monotonically increasing for each request from a given Control Source)
- HMAC authenticator (MUST span message payload, plus secret key)

5.11. List Response

The response to a successful List will consist of a formatted list -- essentially a table -- of filter criteria and related parameters.

Fields will be included and excluded depending on the presence and the value of Stats/Criteria/All entry in the request as noted in square brackets [] following the value listed below. Each entry in the List list shall contain the following fields as specified in the original criteria:

- Control Source Identifier
- Control Source IP Address
- Content Destination Identifier
- Criteria ID
- Date/Time added
- Specified Source IP address, range or IP + bitmask , or wildcard [Criteria]
- Specified Destination IP address, range, or IP + bitmask, or wildcard [Criteria]
- IP Protocol or range, or wildcard [Criteria]
- Source Layer-4 Port or range, or wildcard (parameter only meaningful when IP protocol range includes protocols 6 or 17) [Criteria]
- Destination Layer-4 Port or range, or wildcard (parameter only meaningful when IP protocol range includes 6 or 17) [Criteria]
- ICMP Type or range, or wildcard (parameter only meaningful when IP protocol range includes protocol 1) [Criteria]
- ICMP Code or range, or wildcard (parameter only meaningful when IP protocol range includes protocol 1) [Criteria]
- Timeout specified in seconds total [Criteria]
- Timeout specified in seconds idle [Criteria]
- Timeout specified in packets [Criteria]
- Timeout specified in bytes [Criteria]

The List list shall also contain the following statistical information based on each criterion:

- An ordinal counter to specify the position of this entry in the context of the list
- An integer specifying the total number of entries in the list
- Timeout remaining in seconds total [Stats]
- Timeout remaining in seconds idle [Stats]
- Timeout remaining in packets [Stats]
- Timeout remaining in bytes [Stats]
- An indication if the timeout is STATIC
- Last 10-second average bandwidth, in bits/second [Stats]
- Total number of packets that have matched this Criteria [Stats]

- Total number of bytes that have matched this Criteria [Stats]
- Total times this rule has been Refreshed [Stats]
- Date/Time of last Refresh [Stats]

This response MUST also include the following parameters:

- Timestamp
- Sequence number (MUST match the sequence number for the request)
- HMAC authenticator (MUST span message payload, plus secret key)

Responses to unsuccessful List requests may take any of the following forms:

- Syntax Error
- Unknown Destination Identifier
- Unknown Criteria ID
- Improper Authentication (logged, but never sent to client)
- Invalid Sequence Number (logged, but never sent to client)
- Unknown Control Source Identifier (logged, but never sent to client)

Important Note: the response to the List message, in particular all entries in the generated table, SHOULD be internally consistent and atomic, regardless of the activity in progress at the time of and during the course of transmission of the message. The data SHOULD represent a snapshot of the relevant information at the quantum in time that the List message is processed.

5.12. NoOp Request

This request takes no action on the server whatsoever, other than returning a successful response. The sole purpose of this command is to verify the end-to-end application-layer connectivity between a Control Source and the DTCP Server. The NoOp request may contain the following parameter:

- Flag: SendAsync

See 5.13 NoOp Response for a description of the SendAsync flag.

Additionally, the NoOp request MUST contain the following parameters:

- Control Source Identifier
- Sequence number (MUST be monotonically increasing for each request from a given Control Source)
- HMAC authenticator (MUST span message payload, plus secret key)

5.13. NoOp Response

The response to a successful NoOp will consist of a successful response message indicator, and contain the following parameters:

- Timestamp

- Sequence number (MUST match the sequence number for the request)
- HMAC authenticator (MUST span message payload, plus secret key)

If the SendAsync parameter is specified in the NoOp request, the server shall cause an asynchronous notification message to be sent to any configured notification destinations for that particular Control Source.

5.14. Restart Notification

The Restart notification shall be sent from the server to any configured notification-recipient when the system experiences a failure such that all the filter criteria are lost. The Restart notification shall contain the following parameters:

- Restart Reason, a text string indicating the reason for the restart, if known
- Timestamp
- HMAC authenticator (MUST span message payload, plus secret key)

5.15. Rollover Notification

The Rollover notification shall be sent from the server to any configured notification-recipient when the server experiences a sequence-number rollover. The Rollover notification shall contain the following parameters:

- Timestamp
- HMAC authenticator (MUST span message payload, plus secret key)

5.16. NoOp Notification

The NoOp notification shall be sent from the server to any configured notification-recipient when the DTCP Server receives a NoOp message with the SendAsync parameter present. The NoOp notification shall contain the following parameters:

- Timestamp
- HMAC authenticator (MUST span message payload, plus secret key)

5.17. Timeout Notification

The Timeout notification shall be sent from the server to the appropriate notification-recipient(s) when the server times out a filter criterion on any one of its configured timeout parameters and the criterion contains a SendTimeoutAsync parameter.

The Timeout notification shall contain the following parameters:

- Criteria ID, to indicate the particular criteria that has timed out
- Timeout specified in seconds total [omit if unconfigured]

- Timeout remaining in seconds total [omit if unconfigured]
- Timeout specified in seconds idle [omit if unconfigured]
- Timeout remaining in seconds idle [omit if unconfigured]

- Timeout specified in packets [omit if unconfigured]
- Timeout remaining in packets [omit if unconfigured]
- Timeout specified in bytes [omit if unconfigured]
- Timeout remaining in bytes [omit if unconfigured]
- Timestamp
- HMAC authenticator (MUST span message payload, plus secret key)

5.18. Congestion Notification

The Congestion notification shall be sent from the server to any configured notification-recipient when the total 10-second average data rate (in bits/second) summed over all active filter criteria to a configured Content Destination exceeds the configured soft limit for that destination. The Congestion notification shall contain the following parameters:

- Content Destination ID, to indicate the particular destination experiencing excessive bandwidth
- Current total 10-second average Bandwidth, in bits/second
- Configured SoftLimit Threshold, in bits/second
- Configured HardLimit Threshold, in bits/second
- Timestamp
- HMAC authenticator (MUST span message payload, plus secret key)

Note that since multiple Control Sources may be responsible for this overload, this Notification MUST be sent to all configured Control Sources that have currently-active filter criteria to this particular Content Destination.

5.19. CongestionDelete Notification

The CongestionDelete notification shall be sent from the server to any configured notification-recipient when the total 10-second average data rate (in bits/second) summed over all active filter criteria to a configured Content Destination exceeds the configured hard limit for that destination, causing the DTCP Server to begin purging filter criteria. The CongestionDelete notification shall contain the following parameters:

- Content Destination ID
- List of Criteria ID purged
- Timestamp
- HMAC authenticator (MUST span message payload, plus secret key)

CongestionDelete messages MUST be specifically and uniquely sent to all configured notification-recipients for the Control Sources to which they apply. To be clear: a given Control Source notification-recipient MUST only receive CongestionDelete messages containing Criteria ID created by that Control Source.

5.20. DuplicatesDropped Notification

The DuplicatesDropped notification shall be sent from the server to any configured notification-recipient when capacity has been exceeded in such a way as to cause packets matching criteria added by the corresponding Control Source to be dropped. This notification will be sent periodically as long as packets continue to be dropped. The DuplicatesDropped notification shall contain the following parameters:

- Content Destination ID
- Applicable CriteriaID pertaining to Dropped Packets
- Total Number of Dropped Packets
- Sum of Bytes contained in Dropped Packets
- Timestamp
- HMAC authenticator (MUST span message payload, plus secret key)

DuplicatesDropped messages MUST be specifically and uniquely sent to all configured notification-recipients for the Control Sources to which they apply.

6. Miscellaneous

6.1. Special treatment of response to List request

The List request inherently provides unique functionality with respect to the messaging architecture of DTCP. All other requests result in reasonably terse replies, which can be encapsulated in, at worst, a few IP packets.

However, the List request will generate an arbitrary amount of reply data, since it could contain all requests that are still active, up to the limit of the device. This section specifically describes how responses to the List request are sent.

a) The full reply to the List request MAY consist of multiple packets. Each packet will contain a single "Response" element; therefore, each packet will have a single Status-Line and a single trailer (Authentication-Info) terminated by 2xCRLF. A UDP packet MUST NOT contain more than ONE "Response" element.

b) A "Response" element in each packet shall contain zero or more "List-Resp-Entry" elements (in "List-Resp-Parameters"). Each filter criteria is encoded into a single "List-Resp-Entry" element. The sequence number MUST be identical for all "Response" elements in a multi-packet reply.

c) Each "List-Resp-Entry" element MUST contain the following two elements: "Criteria-Num" and "Criteria-Count". "Criteria-Num" MUST be valued as an enumeration starting with 1 (one) and incrementing by

one for each "List-Resp-Entry" sent. "Criteria-Count" SHOULD be set to the total number of matching Criteria in the given particular LIST response (see below for potential exceptions).

d) Therefore, a full reply to the List request shall consist of as many "List-Resp-Entry" elements as necessary to fully transmit the List, divided into multiple packets as described above.

e) DTCP Servers SHOULD ensure that each "List-Resp-Entry" element is not divided across multiple IP packets.

f) DTCP Clients can use the simple test (Criteria-Num==Criteria-Count) to determine if they've received the last packet in the series. However, in order to ensure that all packets were received (and, respectively, all List-Resp-Entry elements), the DTCP Client MUST traverse through the list of Criteria-Count to ensure it's complete from 1 to XX where XX==Criteria-Num==Criteria-Count.

g) At the UDP layer, all packets in the response MUST contain identical UDP port numbers. DTCP Clients SHOULD maintain their socket open until either all expected Response messages are received, or a timeout occurs.

h) If the List request matches no criteria, but does not supply invalid Criteria-IDs, the "Response" element will contain zero "List-Resp-Entry" elements.

i) DTCP Servers MAY simplify their implementation by only including a single "List-Resp-Entry" element in each "Response" element (and therefore in each packet).

j) DTCP Servers MAY simplify their implementation by transmitting the "Criteria-Count" element in each List-Resp-Entry element as ZERO (0) until the final element is sent, whereupon it is set to the proper value.

A List response that matches 3 criteria may look as follows:

```
=============== First UDP packet
DTCP header
Seq: A
criteria-id: x    ; this is the first List-Resp-Entry element
...
criteria-num: 1
criteria-count: 3
criteria-id: y   ; this is the second List-Resp-Entry element
...
criteria-num: 2
criteria-count: 3
```

```
=============== Second UDP packet
DTCP header
Seq: A
criteria-id: z   ; this is the third List-Resp-Entry element
...
criteria-num: 3
criteria-count: 3
HMAC
================
```

If the list request matches no criteria, it will look as follows:

```
=============== First UDP packet
DTCP header
Seq: B
HMAC
================
```

6.2. Error or Exception Conditions

Errors in DTCP requests are reported in response messages via any Response-Code other than "200" (OK). When such error or exception condition exists, the server SHOULD attempt to indicate the precise nature of the error or exception using the Error-Parameters element. This behavior, though helpful, is not strictly required by the protocol.

For example, if a Delete request contained a specific Criteria-ID not currently active in the server, the response error message MUST begin with a 431 - Unknown Criteria ID response line. The server SHOULD also add the Criteria-ID parameter indicating the unknown Criteria ID.

Again, note that authentication failures MUST NOT be reported in response messages; they MUST be silently dropped.

The DTCP Server MUST attempt to provide the most specific error message to report the specific error or exception condition. When the request message meets any of the following conditions, if no such specific error message exists, the server MAY return a 400 (Bad Request) error:

- Missing required fields
- Parse failure
- Parameters beyond range

In these cases, the server SHOULD include the specific line from the request that caused the condition using the Error-Parameters element.

6.3. Extensions in ABNF

Extension placeholders are provided in the formal syntax for the definition of future methods, parameters, and response-codes. Vendors SHOULD NOT define implementation-specific extensions; rather, such extensions SHOULD be brought to the DTCP working group for inclusion into the protocol, to ensure interoperability.

However, in order to provide faster extensions to the protocol, the "X-" extension parameter construct has been borrowed from other protocols, including SIP and SMTP.

The DTCP Server or the DTCP Client MAY include an arbitrary parameter-value pair, as long as the parameter is preceded by the character string "X-", and all other parameter-value conventions are followed.

The sender of such extension parameters MUST NOT rely on the recipient correctly processing those values.

The recipient of such extension parameters MAY process the values as appropriate upon receipt, but MUST discard without error those extension parameters that it does not recognize.

6.4. Current Version

The current version string for this release of the DTCP protocol is:

DTCP/0.7

6.5. No specific port

While it is common practice to register and/or publish a TCP or UDP port for applications that define them as a layer-4 transport, DTCP has no specific UDP ports predefined. This is intended both to allow flexibility for implementers and users, as well as to make it more difficulty to detect DTCP messages on untrusted networks.

6.6. Unimplemented Protocol Methods and Parameters

Some DTCP Server vendors have indicated their interest in supporting a subset of the functionality specified here, due to their position in the security space. Additionally, some constructs (arbitrary lists, in particular) add complexity to implementations that may not require that complexity.

To address this need, rather than adding complexity by changing the grammar to indicate optional sections, specific error messages have been added to indicate to the client that the server cannot process

the request in its current format. Depending on the request, the client might be able to reformat that request into one that the server implementation is able to process.

In order to be compliant with this protocol, the following rules apply:

a) If a vendor chooses not to implement one or more DTCP Methods, when responding to a request containing one of the unsupported methods, the DTCP Server MUST send a "501 Not Implemented" Response error message, and discontinue processing of that request.

b) If a vendor chooses not to implement a list element, when responding to a request containing such a list, the DTCP Server MUST send a "501 Not Implemented" Response error message, and discontinue processing of that request.

c) If a vendor chooses not to implement one or more specific parameters or parameter value options in a request, the DTCP Server MUST send a "501 Not Implemented" Response error message, and discontinue processing of that request.

d) The DTCP Server SHOULD include the method, parameter, or value which caused the "501 Not Implemented" error to be sent, within the error response message (to be consistent with 6.4 above)

e) The DTCP Server SHOULD support prior versions of DTCP. However, if the vendor chooses not to implement prior versions of the protocol, the DTCP Server MUST send a "505 DTCP Version not supported" error message, and discontinue processing of that request.

The onus is on the client to determine if it can reformat the message to make it acceptable to the particular DTCP Server implementation.

6.7. Version Mismatches

The intent of this section is to clarify any ambiguity arising from mismatches between DTCP versions supported by the DTCP Client and the DTCP Server. In practice is has been observed that unintended consequences have arisen by leaving the implementation vague, so it was decided that clarifing at least a set of reccomendations, if not rising to the level of requirements, will help guide DTCP implementations and help ensure interoperability.

Two possible cases of mismatch exist: when the client exceeds the server version, and when the server exceeds the client version. They are handled separately, but the motivation in each case is to permit maximum compatibility.

In this case, versions are compared numerically, with a single digit after decimal point. For example: 0.4 is greater than 0.2, 1.9 is greater than 1.4, and 3.1 is greater than 2.9

6.7.1. DTCP Client version exceeds DTCP Server version

If the DTCP Client attempts to make a request to a DTCP Server using a DTCP version greater than that supported by the DTCP Server, the DTCP Server MUST return a "505 DTCP Version not supported" error message using the GREATEST DTCP version supported by the DTCP Server, and discontinue processing of that request. It has not way of knowing what new parameters might exist in a newer version of the protocol and simply has to abandon processing altogether.

In this case, the onus is on the DTCP Client to revert to an older version of the protocol specification to talk with this DTCP Server to ensure that its request is properly handled.

6.7.2. DTCP Server version exceeds DTCP Client version

It is expected that a given DTCP Server will support a range of DTCP protocol specification versions, for interoperability purposes.

If the DTCP Server receives a request from a DTCP Server using a DTCP version lesser than the most current version supported by the DTCP Server, the server SHOULD attempt to process that response using the semantics of the earlier specification, and MUST reply using the precise DTCP version included in the request.

If the DTCP server is unable to do this, the DTCP Server MUST return a "505 DTCP Version not supported" error message using the LEAST DTCP version supported by the DTCP Server, and discontinue processing of that request

Asynchronous Notifications sent to a client using an earlier version are addressed in Section 3.2 (Asynchronous Notifications).

7. Message Payload Examples

Note: These are only examples of message payloads, and are not intended to illustrate the full breadth of the protocol. Also, please note that the Authentication-Info shown are correct if each line is terminated with CRLF as specified and the key "secret" is used. (Terminating CRLFs are not shown.)

7.1. Successful ADD Request and Response Payload

Following is an example of the UDP payload for an Add request:

```
ADD DTCP/0.6
Source-Address: 192.168.10.4
Dest-Address: 10.1.1.1-10.1.1.10
Protocol: 6,17
Dest-Port: 53
Timeout-Idle: 600
Action: Copy
Priority: 2
Flags: SendAsync
Cdest-ID: cdst_b
Csource-ID: csrc_a
Seq: 3827443
Authentication-Info: 28eb606458ba46160d7a59da48763020f5aef9f5
```

Following is the UDP payload of one potential successful response to that Add request:

```
DTCP/0.6 200 OK
Criteria-ID: 38224
Seq: 3827443
Timestamp: 2005-01-01 12:01:01.111
Authentication-Info: 38099d03fcb5b12a849b36f9bdccc757303fafd0
```

7.2 Unsuccessful DELETE Request and Response Payload

Following is an example of the UDP payload for an Delete request:

```
DELETE DTCP/0.6
Criteria-ID: 55331
Csource-ID: csrc_d
Flags: Static
Seq: 2655371
Authentication-Info: 6af62247a2b59a2a06e0ca08ec5a80a644e2cd67
```

Following is the UDP payload of one potential unsuccessful response to that Delete request:

```
DTCP/0.6 431 Unknown Criteria ID
Criteria-ID: 55331
Seq: 2655371
Timestamp: 2005-02-02 12:02:02.222
Authentication-Info: 5de4552e98832c2d2c3a9ffa8a2958c967b4e1e8
```

This delete request was unsuccessful because the Criteria ID supplied did not exist. Note that the error-causing parameter is included within the reply to assist in debugging.

8. Formal Syntax

All of the mechanisms specified in this document are described in both prose and an Augmented Backus-Naur Form (ABNF) defined in RFC 4234 [7]. Section 6.1 of RFC 4234 defines a set of core rules that are used by this specification, and not repeated here. Implementers need to be familiar with the notation and content of RFC 4234 in order to understand this specification. Certain basic rules are in uppercase, such as SP, LWS, HTAB, CRLF, DIGIT, ALPHA, etc.

Note that while much of this syntax is taken from the Session Initiation Protocol (SIP), some of its constructs have been simplified for this application here. Where appropriate, these digressions have been noted with comments.

The following core definitions appear throughout the formal syntax:

```
COL             =  *(WSP) ":" *(WSP) ;  used in parameter-value pair
NPCHAR          =  "\" 3DIGIT        ;  used to express ctrl-chars
DSTRING         =  *(VCHAR / NPCHAR) ;  no embedded whitespace
WC              =  "*"               ;  wildcard character for matching
NOT             =  "!"               ;  invert character for matching
N64BITNUM          =  1*20DIGIT
N32BITNUM          =  1*10DIGIT
N16BITNUM          =  1*5DIGIT
N8BITNUM        =  1*3DIGIT
DAYSEC          =  1*5DIGIT
IPv4address     =  1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT
TEXT            =  1*(1*(VCHAR) WSP) ; includes whitespace
DTCP-Time       =  4DIGIT "-" 2DIGIT "-" 2DIGIT SP 2DIGIT ":"
                   2DIGIT ":" 2DIGIT "." 3DIGIT
                   ; This is ISO date/time: YYYY-MM-DD sp HH:MM:SS.TTT
```

Additionally, the following definitions are excerpted from RFC 3986 [8]:

```
      IPv6address =                            6( h16 ":" ) ls32
                  /                       "::" 5( h16 ":" ) ls32
                  / [               h16 ] "::" 4( h16 ":" ) ls32
                  / [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
                  / [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
                  / [ *3( h16 ":" ) h16 ] "::"    h16 ":"   ls32
                  / [ *4( h16 ":" ) h16 ] "::"              ls32
                  / [ *5( h16 ":" ) h16 ] "::"              h16
                  / [ *6( h16 ":" ) h16 ] "::"

      ls32        = ( h16 ":" h16 ) / IPv4address
                  ; least-significant 32 bits of address
```

```
h16         = 1*4HEXDIG
            ; 16 bits of address represented in hexadecimal
```

Here begins the formal syntax:

```
DTCP-Message   =  Request / Response / Notification

Request        =  Request-Line
                  ( Add-Req-Parameters
                /  Delete-Req-Parameters
                /  Refresh-Req-Parameters
                /  List-Req-Parameters
                /  Noop-Req-Parameters)
                  *(extension-parameter)
                  Csource-ID
                  Seq
                  Authentication-Info
                  CRLF

Response       =  Status-Line
                  ( ( Add-Resp-Parameters
                /  Delete-Resp-Parameters
                /  Refresh-Resp-Parameters
                /  List-Resp-Parameters
                /  Noop-Resp-Parameters) / Error-Parameters )
                  *(extension-parameter)
                  Timestamp
                  Seq                  ; note absence of Csource-ID
                  Authentication-Info
                  CRLF

Notification   =  Status-Line
                  ( Restart-Notif-Parameters
                /  Rollover-Notif-Parameters
                /  Noop-Notif-Parameters
                /  Timeout-Notif-Parameters
                /  Congestion-Notif-Parameters
                /  CongDel-Notif-Parameters )
                  *(extension-parameter)
                  Timestamp
                  Authentication-Info  ;  note absence of Seq
                  CRLF

DTCP-Version   =  "DTCP" "/" 1*DIGIT "." 1*DIGIT

Status-Line    =  DTCP-Version SP Status-Code SP Reason-Phrase CRLF
Request-Line   =  Method SP DTCP-Version CRLF

Method         =  "ADD" / "DELETE" / "REFRESH" / "LIST" / "NOOP"
                  / extension-method

extension-method  =  DSTRING
```

```
   Status-Code    =  Provisional
                  /  Success
```

```
                 /  Redirection
                 /  Request-Failure
                 /  Server-Failure
                 /  Global-Failure
                 /  extension-code

   Reason-Phrase  =  TEXT  ; differs from SIP
   extension-code =  3DIGIT

   Provisional       =  "130" ;  Sequence Number Rollover (Notif)
                     /  "131" ;  NoOp Notification (Notif)
   Success           =  "200" ;  OK
   Redirection       =  "390" ;  Criterion Timeout Delete (Notif)
   Request-Failure   =  "400" ;  Bad Request
                     /  "430" ;  Unknown Content Destination
                     /  "431" ;  Unknown Criteria ID
                     /  "432" ;  Improper Filter Specification
                     /  "433" ;  Improper Timeout Specification
                     /  "497" ;  Invalid Authentication
                                 ;  (never sent to client)
                     /  "498" ;  Invalid Sequence Number
                                 ;  (never sent to client)
                     /  "499" ;  Unknown Control Source Identifier
                                 ;  (never sent to client)
   Server-Failure    =  "500" ;  Server Internal Error
                     /  "501" ;  Not Implemented
                     /  "505" ;  DTCP Version not supported
                     /  "550" ;  Max Criteria Limit Exceeded
                     /  "551" ;  Max Content Destination Exceeded
                     /  "580" ;  Congestion (Notif)
                     /  "598" ;  Duplicate Packets Dropped (Notif)
                     /  "599" ;  Server Restart (Notif)
   Global-Failure    =  "680" ;  Criterion Congestion Delete (Notif)

   Error-Parameters  =  Cdest-ID
                     /  Criteria-ID
                     /  Filter-Block
                     /  Timeout-Block

   Add-Req-Parameters   =  Filter-Block Timeout-Block [Action]
                           Option-Block [Flags] Cdest-ID

   Filter-Block              =  *(Filter-Element)
   Timeout-Block             =  *(Timeout-Element)
   TRemaining-Block          =  *(TRemaining-Element)
   Option-Block              =  *(Option-Element)
   Timeout-Required-Block    =  1*(Timeout-Element)
   TRemaining-Required-Block  =  1*(TRemaining-Element)
```

```
   Filter-Element    =  Source-Address
                     /  Dest-Address
```

```
                    /  Protocol
                    /  Source-Port
                    /  Dest-Port
                    /  ICMP-Type
                    /  ICMP-Code

   Timeout-Element   =  Timeout-Idle
                     /  Timeout-Total
                     /  Timeout-Packets
                     /  Timeout-Bytes

   TRemaining-Element   =  Remaining-Idle
                        /  Remaining-Total
                        /  Remaining-Packets
                        /  Remaining-Bytes

   Option-Element    =  Priority

   Add-Resp-Parameters  =  Criteria-ID

   Delete-Req-Parameters   =  ( (Criteria-ID / Criteria-ID-Filter)
                              Cdest-ID ) [Flags]
   Delete-Resp-Parameters  =  Criteria-Count

   Refresh-Req-Parameters  =  ( (Criteria-ID / Criteria-ID-Filter)
                              Cdest-ID ) Timeout-Required-Block
   Refresh-Resp-Parameters =  Criteria-Count

   List-Req-Parameters  =  [ ( (Criteria-ID / Criteria-ID-Filter)
                              Cdest-ID ) ] [Flags]

   List-Resp-Parameters =  *(List-Resp-Entry CRLF)

   List-Resp-Entry   =  Criteria-Count Criteria-Num Main-List
                        [Criteria-List] [Stats-List]

   Main-List            =  Csource-ID Csource-Address Cdest-ID
                           Criteria-ID Timestamp
   Criteria-List        =  *(Filter-Element) *(Timeout-Element) [Flags]
   Stats-List           =  TRemaining-Block Stats-Block

   Stats-Block       =  Average-Bandwidth Matching-Packets
                        Matching-Bytes Num-Refresh Last-Refresh

   Noop-Req-Parameters  =  [Flags]
   Noop-Resp-Parameters =  []    ;  no parameters

   Restart-Notif-Parameters      =  Alert-Info
   Rollover-Notif-Parameters     =  []       ; no parameters
```

```
   Noop-Notif-Parameters         =  []       ; no parameters
   Timeout-Notif-Parameters      =  Criteria-ID
                                /  Timeout-Required-Block
```

```
                                /  TRemaining-Required-Block
   Congestion-Notif-Parameters   =  Cdest-ID Average-Bandwidth
                                    Alert-Bandwidth Max-Bandwidth
   CongDel-Notif-Parameters      =  Cdest-ID Criteria-ID-Filter

   extension-parameter  =  "X-" DSTRING COL DSTRING CRLF

   Csource-ID           =  "Csource-ID" COL DSTRING CRLF
   Seq                  =  "Seq" COL N64BITNUM CRLF
   Authentication-Info  =  "Authentication-Info" COL 40HEXDIG CRLF
   ID-List              =  DSTRING *("," DSTRING)
   Cdest-ID             =  "Cdest-ID" COL ID-List CRLF
   Source-Address       =  "Source-Address" COL IPFilter CRLF
   Dest-Address         =  "Dest-Address" COL IPFilter CRLF
   Protocol             =  "Protocol" COL ProtFilter CRLF
   Source-Port          =  "Source-Port" COL PortFilter CRLF
   Dest-Port            =  "Dest-Port" COL PortFilter CRLF
   ICMP-Type            =  "ICMP-Type" COL ICMPFilter CRLF
   ICMP-Code            =  "ICMP-Code" COL ICMPFilter CRLF
   Timeout-Idle         =  "Timeout-Idle" COL DAYSEC CRLF
   Timeout-Total        =  "Timeout-Total" COL DAYSEC CRLF
   Timeout-Packets      =  "Timeout-Packets" COL N32BITNUM CRLF
   Timeout-Bytes        =  "Timeout-Bytes" COL N64BITNUM CRLF
   Priority             =  "Priority" COL N8BITNUM CRLF
   Criteria-ID          =  "Criteria-ID" COL N32BITNUM CRLF
   Criteria-ID-Filter   =  "Criteria-ID" COL CritFilter CRLF
   Criteria-Count       =  "Criteria-Count" COL N32BITNUM CRLF
   Criteria-Num         =  "Criteria-Num" COL N32BITNUM CRLF
   Csource-Address      =  "Csource-Address" COL (IPv4address /
                           IPv6address) CRLF
   Timestamp            =  "Timestamp" COL DTCP-Time CRLF
   Remaining-Idle       =  "Remaining-Idle" COL DAYSEC CRLF
   Remaining-Total      =  "Remaining-Total" COL DAYSEC CRLF
   Remaining-Packets    =  "Remaining-Packets" COL N32BITNUM CRLF
   Remaining-Bytes      =  "Remaining-Bytes" COL N64BITNUM CRLF
   Average-Bandwidth    =  "Average-Bandwidth" COL N64BITNUM CRLF
   Matching-Packets     =  "Matching-Packets" COL N64BITNUM CRLF
   Matching-Bytes       =  "Matching-Bytes" COL N64BITNUM CRLF
   Num-Refresh          =  "Num-Refresh" COL N32BITNUM CRLF
   Last-Refresh         =  "Last-Refresh" COL DTCP-Time CRLF
   Alert-Info           =  "Alert-Info" COL TEXT CRLF
   Alert-Bandwidth      =  "Alert-Bandwidth" COL N64BITNUM CRLF
   Max-Bandwidth        =  "Max-Bandwidth" COL N64BITNUM CRLF
   Action               =  "Action" COL ActionEntry CRLF

   ActionEntry          =  "Copy"
                        /  "Block"
                        /  "Redirect"
```

```
                        /  extension-action

   extension-action     =  DSTRING
```

```
   Flags               = "Flags" COL FlagEntry *("," FlagEntry) CRLF

   FlagEntry           =  "Static"
                       /  "SendAsync"
                       /  "Stats"
                       /  "Criteria"
                       /  "Both"

   IPFilter    =  [NOT] IPEntry *("," [WSP] [NOT] IPEntry)

   ProtFilter  =  [NOT] ProtEntry *("," [WSP] [NOT] ProtEntry)

   PortFilter  =  [NOT] PortEntry *("," [WSP] [NOT] PortEntry)

   ICMPFilter  =  [NOT] ICMPEntry *("," [WSP] [NOT] ICMPEntry)

   CritFilter  =  [NOT] CritEntry *("," [WSP] [NOT] CritEntry)

   IPEntry     =  IPv4Entry
               /  IPv6Entry


   IPv4Entry   =  IPv4address                  ;  Single Entry
               /  IPv4address "/" N8BITNUM     ;  Address/mask
               /  IPv4address "-" IPv4address  ;  Range
               /  IPv4address "-" WC           ;  Range to UBOUND
               /  WC "-" IPv4address           ;  LBOUND to Range
               /  WC                           ;  Pure Wildcard

   IPv6Entry   =  IPv6address                  ;  Single Entry
               /  IPv6address "/" N8BITNUM     ;  Address/mask
               /  IPv6address "-" IPv6address  ;  Range
               /  IPv6address "-" WC           ;  Range to UBOUND
               /  WC "-" IPv6address           ;  LBOUND to Range
               /  WC                           ;  Pure Wildcard

   PortEntry   =  N16BITNUM                       ;  Single Entry
               /  N16BITNUM "-" N16BITNUM         ;  Range
               /  N16BITNUM "-" WC                ;  Range to UBOUND
               /  WC "-" N16BITNUM                ;  LBOUND to Range
               /  WC                           ;  Pure Wildcard

   ProtEntry   =  N8BITNUM                     ;  Single Entry
               /  N8BITNUM "-" N8BITNUM        ;  Range
               /  N8BITNUM "-" WC              ;  Range to UBOUND
               /  WC "-" N8BITNUM              ;  LBOUND to Range
               /  WC                           ;  Pure Wildcard

   ICMPEntry   =  N8BITNUM                     ;  Single Entry
```

```
                /  N8BITNUM "-" N8BITNUM         ;  Range
                /  N8BITNUM "-" WC               ;  Range to UBOUND
                /  WC "-" N8BITNUM               ;  LBOUND to Range
```

```
                /  WC                           ;  Pure Wildcard

   CritEntry    =  N64BITNUM                       ;  Single Entry
                /  N64BITNUM "-" N64BITNUM         ;  Range
                /  N64BITNUM "-" WC                ;  Range to UBOUND
                /  WC "-" N64BITNUM                ;  LBOUND to Range
                /  WC                           ;  Pure Wildcard
```

9. Security Considerations

DTCP empowers network security personnel to monitor packet data transitioning through a network element. As such, it is a powerful protocol that can cause any network data to be redirected to a arbitrary location for inspection. Consequently, it is of greatest criticality that any DTCP Servers fully implement the security model outlined in this draft. Failure to do so could result in malicious individuals either obtaining unauthorized access to data or interruption of data transmission.

10. IANA Considerations

This document has no actions for IANA.

11. Conclusions

This protocol has undergone extensive testing and several rounds of refinements. The resulting protocol is highly effective at meeting its goals of providing a real-time mechanism to inspect raw packets containing security-related events traversing a network in real-time.

12. Acknowledgments

Thanks to all at AT&T and Juniper Networks who provided testing and support for this experimental protocol!

The authors would specifically like to thank Joju Chevookaran, and Saravanan Deenadayalan from Juniper since they have not only worked hard on the implementation, but also on the some of the enhancements (specially VRF support, input/out interface filters etc).

Also, Rick Suntag, Michael Nanashko, and Michael St. Angelo from AT&T all deserve special note for extensive testing as well as excellent protocol definition suggestions and corrections.

This document was prepared using 2-Word-v2.0.template.dot.

APPENDIX A: Prior Implementation

This document fully and accurately describes the operation of DTCP/0.7 implementations. However, in development of this protocol, some implementations with working versions of the protocol were released. This appendix describes the differences between the DTCP/0.7 protocol specification documented herein and the prior DTCP/0.5 and DTCP/0.6 protocol specifications.

Other than the changes documented in this appendix, the older protocol specifications precisely follow DTCP/0.7 described herein. This appendix is provided for backward-compatibility purposes only; all new implementations should ignore this appendix.

A.1. Version Number

(Modifies section 6.4 Current Version)

- The prior supported version string was exactly:
  DTCP/0.6

A.2. Response to List request

(Modifies sections 5.11 List Response, 6.1 Special treatment of response to List request and 8 Formal Syntax)

The following changes apply only to the elements involved in the Response message used in reply to the List action. Changes are both syntactic and semantic in nature.

- The ABNF element called "Criteria-Num" in DTCP/0.7 did not exist in DTCP/0.5 and was not included in any DTCP message.
- The ABNF element called "Criteria-Count" in DTCP/0.7 was called "Num-Criteria" in DTCP/0.5.
- The "Num-Criteria" element was only included in the final UDP packet sent. This signals the end of the List response.

A.3. Changes in Response Codes

1. 550: Max Criteria Limit Exceeded

This error message is sent when the number of DTCP ADD requests received by the server exceeds the allowed limit. Error code 500 used in the earlier Flow-Tap implementations was not clear enough.

2. 551: Max Content Destination Exceeded

Server allows only a certain number of Content Destinations at any given time, and generates this error message when the server receives

a DTCP ADD request that contains a new content destination after the number of Content Destinations on that server has already exceeded the allowable limit.

3. 432: Improper Filter Specification

Generated when an ADD request contains a combination of X-JTap-VRF-Name, X-JTap-Input-Interface, and X-JTap-Output-Interface fields.

A.4. IP Version 6

The formal ABNF syntax has been updated to include IP Version 6 in parallel with IP Version 4 both in the filter criteria specification as well as ancillary addressing information. The intent was to permit the protocol to operate largely unmodified while allowing the use of IP Version 6 addressing information. Some implementations may not support this addressing mode.

A.5. Sequence Number Negative Window

The Negative Window sequence number concept has been added to this version of DTCP to address empirical errors found when testing with a high rate of DTCP "ADD" messages over a non-trivial network.

A.6. Version Mismatches

The section on Version Mismatches was added, to account for specific problems encountered during upgrade of either the client or the server. In particular, the draft was ambiguous on how the DTCP Server should behave when servicing clients of various versions.

APPENDIX B: DTCP Vendor-Specific Extensions

B.1. Juniper Networks: "Flow-Tap"

B.1.1. "Flow-Tap" DTCP Extensions

In support of Flow-Tap functionality, the DTCP grammar has been extended to include new parameters defined below. In general, the purpose of these parameters is to allow a content destination to be configured on-demand, rather than pre-configured. General DTCP grammar does not provide this functionality, so we extend it herein.

Note that "JTap-Failure" below is not a grammar tag; it just defines a new error value "901' that will be used to indicate any problems with the X-JTap parameters.

1. X-JTap-Cdest-Dest-Address

IP address(es) of Content destination(s) where the matching packets need to be sent out. User may specify maximum two IP addresses separated by a comma. This field MUST be present in the ADD request otherwise "JTAP-Failure" error will be generated.

2. X-JTap-Cdest-Dest-Port

UDP port number(s) of Content destination(s) where the matching packets need to be sent out. User may specify maximum two port numbers separated by a comma. This field MUST be present in the ADD request otherwise "JTAP-Failure" error will be generated.

3. X-JTap-Cdest-TTL

TTL value to be used in the forwarded packet. This is an optional field and default of 255 will be used if not specified

4. X-JTap-Cdest-Source-Address

Source IP address from which the forwarded packet needs to besent from This field MUST be present in the ADD request and "JTap-Failure" error will be generated if this is not specified

5. X-JTap-Cdest-Source-Port

Source UDP port from which the forwarded packet needs to be sent from

This field MUST be present in the ADD request and "JTap-Failure" error will be generated if this is not specified.

6. Changes in Cdest-ID

Cdest-ID field enables you to specify more than one content destination by using a comma separated list. Currently, only two

content destinations are supported. However, note that the number of entries in all the three fields, Cdest-ID, X-JTap-Cdest-Dest-Address and X-JTap-Cdest-Dest-Port, must be the same. That is, if you have only one entry in one of the fields, the other two can have only one entry each. Error message 432 is generated if the number of entries in these fields does not match with each other.

7. X-JTap-VRF-Name

OPTIONAL field to specify a VRF name. If it is specified, only the packets coming to the specified VRF will be monitored. "JTap-Failure" error will be generated if the VRF is not configured

8. X-JTap-Input-Interface

OPTIONAL field to specify a list of interfaces. If it is specified, it will be attached to respective input interface(s) instead of global Flow-Tap filters. This list may contain maximum 8 interfaces separated by comma. If the unit name of an interface is not specified, System will assume it as unit 0. "JTap-Failure" error will be generated if any one of the interfaces in the list is not configured

9. X-JTap-Output-Interface

OPTIONAL field to specify a list of interfaces. If it is specified, the filter will be attached to respective output interface(s) instead of global Flow-Tap filters. This list may contain maximum 8 interfaces separated by comma. If the unit name of an interface is not specified, System will assume it as unit 0. "JTap-Failure" error will be generated if any one of the interfaces in the list is not configured

B.1.2. "Flow-Tap" extension ABNF

```
IP-4-OR-6                = (IPv4address / IPv6address)
ADDR-LIST                = IP-4-OR-6 1*("," IP-4-OR-6)
PORT-LIST                = N16BITNUM 1*("," N16BITNUM)
IFL   = 3CHAR "-" 2*DIGIT "/" 1*DIGIT "/" 1*DIGIT ["." N16BITNUM]
IFL-LIST8 = IFL 7*("," IFL)

X-JTap-Cdest-Dest-Address = "X-JTap-Cdest-Dest-Address" COL ADDR-LIST
CRLF
X-JTap-Cdest-Dest-Port    = "X-JTap-Cdest-Dest-Port" COL PORT-LIST
CRLF
X-JTap-Cdest-TTL = "X-JTap-Cdest-TTL" COL N8BITNUM CRLF
X-JTap-Cdest-Source-Address = "X-JTap-Cdest-Source-Address" COL
(IPv4address / IPv6address) CRLF
X-JTap-Cdest-Source-Port = "X-JTap-Cdest-Source-Port" COL N16BITNUM
```

```
   CRLF
   X-JTap-VRF-Name = "X-JTap-VRF-Name" COL DSTRING CRLF
   X-JTap-Input-Interface = "X-JTap-Input-Interface" COL IFL-LIST8 CRLF
```

```
   X-JTap-Output-Interface = "X-JTap-Output-Interface" COL IFL-LIST8
   CRLF
```

13. References

13.1. Normative References

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

[2] Krawczyk, H. et. al., "RFC 2104: HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997

[3] FIPS 180-1, "Secure Hash Standard". May 10995. (http://www.itl.nist.gov/fipspubs/fip180-1.htm)

[4] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 3629, November 2003

[5] Berners-Lee, T., Fielding, R. and H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 10996

[6] Rosenberg, et al. "SIP: Session Initiation Protocol", RFC 3261, June 2002

[7] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 5234, January 2008

[8] Berners-Lee, T., Fielding, R. , Masinter L., "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005

13.2. Informative References

[9] "CAIDA: The Cooperative Association for Internet Data Analysis" (http://www.caida.org/tools/measurement/cflowd/)

Authors' Addresses

David J. Cavuto
AT&T
200 Laurel Ave South #C2-3B10
Middletown, NJ 07748
USA

Email: dcavuto@att.com

Manoj S. Apte
Juniper Networks
1194 North Mathilda Avenue
Sunnyvale, CA 94089
USA

Email: mapte@juniper.net

Sandeep Jain
Juniper Networks
1194 North Mathilda Avenue
Sunnyvale, CA 94089
USA

Email: sjain@juniper.net

Muku Murthy
Juniper Networks
1194 North Mathilda Avenue
Sunnyvale, CA 94089
USA

Email: muku@juniper.net