

DOTS
Internet-Draft
Intended status: Standards Track
Expires: September 29, 2017

T. Reddy
Cisco
M. Boucadair
Orange
P. Patil
Cisco
A. Mortensen
Arbor Networks, Inc.
N. Teague
Verisign, Inc.
March 28, 2017

**Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal
Channel
draft-reddy-dots-signal-channel-10**

Abstract

This document specifies the DOTS signal channel, a protocol for signaling the need for protection against Distributed Denial-of-Service (DDoS) attacks to a server capable of enabling network traffic mitigation on behalf of the requesting client. A companion document defines the DOTS data channel, a separate reliable communication layer for DOTS management and configuration.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on September 29, 2017.

Copyright Notice

Copyright (c) 2017 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 Simplified BSD License.

Table of Contents

- [1.](#) Introduction [3](#)
- [2.](#) Notational Conventions and Terminology [3](#)
- [3.](#) Solution Overview [4](#)
- [4.](#) Happy Eyeballs for DOTS Signal Channel [5](#)
- [5.](#) DOTS Signal Channel [7](#)
 - [5.1.](#) Overview [7](#)
 - [5.2.](#) DOTS Signal YANG Model [8](#)
 - [5.2.1.](#) Mitigation Request Model structure [8](#)
 - [5.2.2.](#) Mitigation Request Model [8](#)
 - [5.2.3.](#) Session Configuration Model structure [10](#)
 - [5.2.4.](#) Session Configuration Model [10](#)
 - [5.3.](#) Mitigation Request [12](#)
 - [5.3.1.](#) Requesting mitigation [12](#)
 - [5.3.2.](#) Withdraw a DOTS Signal [17](#)
 - [5.3.3.](#) Retrieving a DOTS Signal [18](#)
 - [5.3.4.](#) Efficacy Update from DOTS Client [22](#)
 - [5.4.](#) DOTS Signal Channel Session Configuration [24](#)
 - [5.4.1.](#) Discover Acceptable Configuration Parameters [25](#)
 - [5.4.2.](#) Convey DOTS Signal Channel Session Configuration [26](#)
 - [5.4.3.](#) Delete DOTS Signal Channel Session Configuration [28](#)
 - [5.4.4.](#) Retrieving DOTS Signal Channel Session Configuration [28](#)
 - [5.5.](#) Redirected Signaling [29](#)
 - [5.6.](#) Heartbeat Mechanism [30](#)
- [6.](#) Mapping parameters to CBOR [31](#)
- [7.](#) (D)TLS Protocol Profile and Performance considerations [31](#)
 - [7.1.](#) MTU and Fragmentation Issues [32](#)
- [8.](#) (D)TLS 1.3 considerations [33](#)
- [9.](#) Mutual Authentication of DOTS Agents & Authorization of DOTS Clients [34](#)
- [10.](#) IANA Considerations [36](#)
 - [10.1.](#) DOTS signal channel CBOR Mappings Registry [36](#)
 - [10.1.1.](#) Registration Template [36](#)
 - [10.1.2.](#) Initial Registry Contents [36](#)
- [11.](#) Implementation Status [39](#)
 - [11.1.](#) nttdots [40](#)
- [12.](#) Security Considerations [40](#)

13.	Contributors	41
14.	Acknowledgements	41
15.	References	42
15.1.	Normative References	42
15.2.	Informative References	43
	Authors' Addresses	45

[1.](#) Introduction

A distributed denial-of-service (DDoS) attack is an attempt to make machines or network resources unavailable to their intended users. In most cases, sufficient scale can be achieved by compromising enough end-hosts and using those infected hosts to perpetrate and amplify the attack. The victim in this attack can be an application server, a host, a router, a firewall, or an entire network.

In many cases, it may not be possible for a network administrator to determine the causes of an attack, but instead just realize that certain resources seem to be under attack. This document defines a lightweight protocol permitting a DOTS client to request mitigation from one or more DOTS servers for protection against detected, suspected, or anticipated attacks. This protocol enables cooperation between DOTS agents to permit a highly-automated network defense that is robust, reliable and secure.

The requirements for DOTS signal channel protocol are obtained from [[I-D.ietf-dots-requirements](#)].

This document satisfies all the use cases discussed in [[I-D.ietf-dots-use-cases](#)] except the Third-party DOTS notifications use case in Section 3.2.3 of [[I-D.ietf-dots-use-cases](#)] which is an optional feature and not a core use case. Third-party DOTS notifications are not part of the DOTS requirements document. Moreover, the DOTS architecture does not assess whether that use case may have an impact on the architecture itself and/or the DOTS trust model.

This is a companion document to the DOTS data channel specification [[I-D.reddy-dots-data-channel](#)] that defines a configuration and bulk data exchange mechanism supporting the DOTS signal channel.

[2.](#) Notational Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [[RFC2119](#)].

(D)TLS: For brevity this term is used for statements that apply to both Transport Layer Security [RFC5246] and Datagram Transport Layer Security [RFC6347]. Specific terms will be used for any statement that applies to either protocol alone.

The reader should be familiar with the terms defined in [I-D.ietf-dots-architecture].

3. Solution Overview

Network applications have finite resources like CPU cycles, number of processes or threads they can create and use, maximum number of simultaneous connections it can handle, limited resources of the control plane, etc. When processing network traffic, such applications are supposed to use these resources to offer the intended task in the most efficient fashion. However, an attacker may be able to prevent an application from performing its intended task by causing the application to exhaust the finite supply of a specific resource.

TCP DDoS SYN-flood, for example, is a memory-exhaustion attack on the victim and ACK-flood is a CPU exhaustion attack on the victim ([RFC4987]). Attacks on the link are carried out by sending enough traffic such that the link becomes excessively congested, and legitimate traffic suffers high packet loss. Stateful firewalls can also be attacked by sending traffic that causes the firewall to hold excessive state. The firewall then runs out of memory, and can no longer instantiate the state required to pass legitimate flows. Other possible DDoS attacks are discussed in [RFC4732].

In each of the cases described above, the possible arrangements between the DOTS client and DOTS server to mitigate the attack are discussed in [I-D.ietf-dots-use-cases]. An example of network diagram showing a deployment of these elements is shown in Figure 1. Architectural relationships between involved DOTS agents is explained in [I-D.ietf-dots-architecture]. In this example, the DOTS server is operating on the access network.

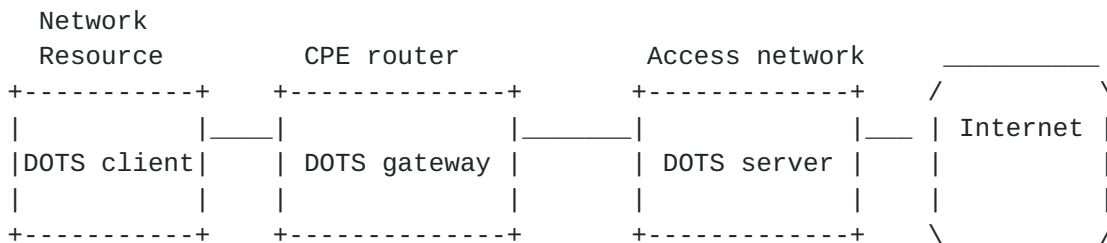


Figure 1

The DOTS server can also be running on the Internet, as depicted in Figure 2.

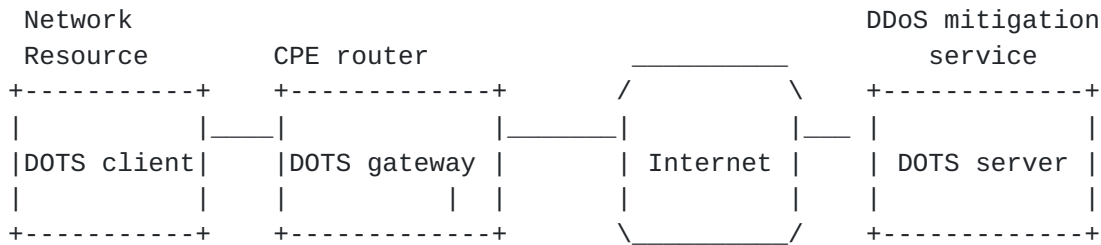


Figure 2

In typical deployments, the DOTS client belongs to a different administrative domain than the DOTS server. For example, the DOTS client is a firewall protecting services owned and operated by an domain, while the DOTS server is owned and operated by a different domain providing DDoS mitigation services. That domain providing DDoS mitigation service might, or might not, also provide Internet access service to the website operator.

The DOTS server may (not) be co-located with the DOTS mitigator. In typical deployments, the DOTS server belongs to the same administrative domain as the mitigator.

The DOTS client can communicate directly with the DOTS server or indirectly via a DOTS gateway.

This document focuses on the DOTS signal channel.

4. Happy Eyeballs for DOTS Signal Channel

DOTS signaling can happen with DTLS [RFC6347] over UDP and TLS [RFC5246] over TCP. A DOTS client can use DNS to determine the IP address(es) of a DOTS server or a DOTS client may be provided with the list of DOTS server IP addresses. The DOTS client MUST know a DOTS server's domain name; hard-coding the domain name of the DOTS server into software is NOT RECOMMENDED in case the domain name is not valid or needs to change for legal or other reasons. The DOTS client performs A and/or AAAA record lookup of the domain name and the result will be a list of IP addresses, each of which can be used to contact the DOTS server using UDP and TCP.

If an IPv4 path to reach a DOTS server is found, but the DOTS server's IPv6 path is not working, a dual-stack DOTS client can experience a significant connection delay compared to an IPv4-only DOTS client. The other problem is that if a middlebox between the DOTS client and DOTS server is configured to block UDP, the DOTS

client will fail to establish a DTLS session with the DOTS server and will, then, have to fall back to TLS over TCP incurring significant connection delays. [I-D.ietf-dots-requirements] discusses that DOTS client and server will have to support both connectionless and connection-oriented protocols.

To overcome these connection setup problems, the DOTS client can try connecting to the DOTS server using both IPv6 and IPv4, and try both DTLS over UDP and TLS over TCP in a fashion similar to the Happy Eyeballs mechanism [RFC6555]. These connection attempts are performed by the DOTS client when it initializes, and the client uses that information for its subsequent alert to the DOTS server. In order of preference (most preferred first), it is UDP over IPv6, UDP over IPv4, TCP over IPv6, and finally TCP over IPv4, which adheres to address preference order [RFC6724] and the DOTS preference that UDP be used over TCP (to avoid TCP's head of line blocking).

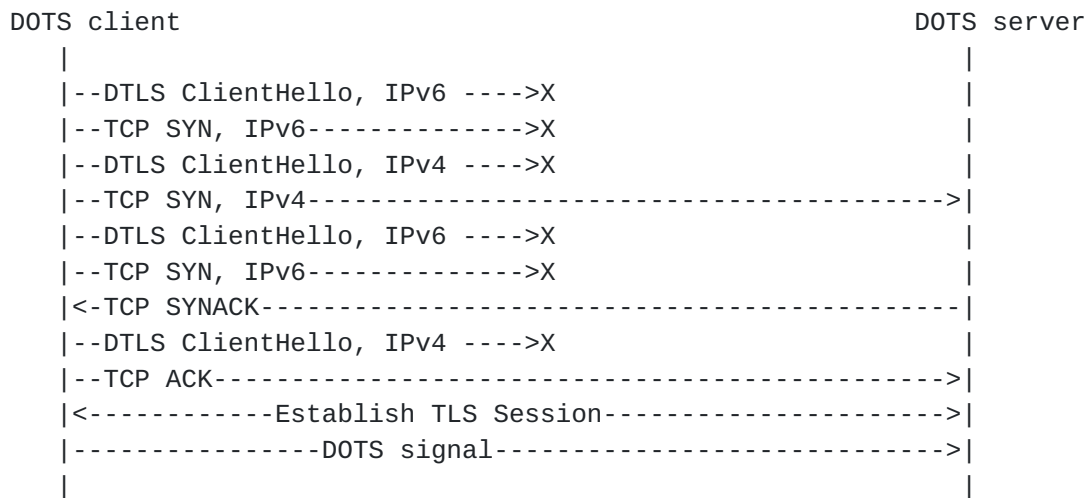


Figure 3: Happy Eyeballs

In reference to Figure 3, the DOTS client sends two TCP SYNs and two DTLS ClientHello messages at the same time over IPv6 and IPv4. In this example, it is assumed that the IPv6 path is broken and UDP is dropped by a middle box but has little impact to the DOTS client because there is no long delay before using IPv4 and TCP. The DOTS client repeats the mechanism to discover if DOTS signaling with DTLS over UDP becomes available from the DOTS server, so the DOTS client can migrate the DOTS signal channel from TCP to UDP, but such probing SHOULD NOT be done more frequently than every 24 hours and MUST NOT be done more frequently than every 5 minutes.

5. DOTS Signal Channel

5.1. Overview

The DOTS signal channel is built on top of the Constrained Application Protocol (CoAP) [[RFC7252](#)], a lightweight protocol originally designed for constrained devices and networks. CoAP's expectation of packet loss, support for asynchronous non-confirmable messaging, congestion control, small message overhead limiting the need for fragmentation, use of minimal resources, and support for (D)TLS make it a good foundation on which to build the DOTS signaling mechanism.

The DOTS signal channel is layered on existing standards (Figure 4).

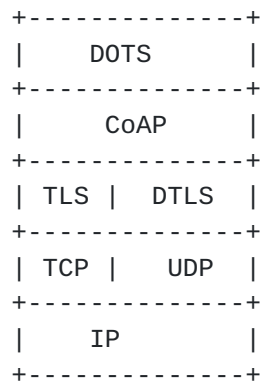


Figure 4: Abstract Layering of DOTS signal channel over CoAP over (D)TLS

The signal channel is initiated by the DOTS client. Once the signal channel is established, the DOTS agents periodically send heartbeats to keep the channel active. At any time, the DOTS client may send a mitigation request message to the DOTS server over the active channel. While mitigation is active, the DOTS server periodically sends status messages to the client, including basic mitigation feedback details. Mitigation remains active until the DOTS client explicitly terminates mitigation, or the mitigation lifetime expires.

Messages exchanged between DOTS client and server are serialized using Concise Binary Object Representation (CBOR) [[RFC7049](#)], CBOR is a binary encoding designed for small code and message size. CBOR encoded payloads are used to convey signal channel specific payload messages that convey request parameters and response information such as errors. This specification uses the encoding rules defined in [[I-D.ietf-core-yang-cbor](#)] for representing mitigation scope and DOTS signal channel session configuration data defined using YANG ([Section 5.2](#)) as CBOR data.

5.2. DOTS Signal YANG Model

This document defines a YANG [RFC6020] data model for mitigation scope and DOTS signal channel session configuration data.

5.2.1. Mitigation Request Model structure

This document defines the YANG module "ietf-dots-signal", which has the following structure:

```
module: ietf-dots-signal
  +--rw mitigation-scope
    +--rw scope* [policy-id]
      +--rw policy-id          int32
      +--rw target-ip*         inet:ip-address
      +--rw target-prefix*     inet:ip-prefix
      +--rw target-port-range* [lower-port upper-port]
        | +--rw lower-port     inet:port-number
        | +--rw upper-port     inet:port-number
      +--rw target-protocol*   uint8
      +--rw FQDN*              inet:domain-name
      +--rw URI*               inet:uri
      +--rw alias*             string
      +--rw lifetime?         int32
```

5.2.2. Mitigation Request Model

<CODE BEGINS> file "ietf-dots-signal@2016-11-28.yang"

```
module ietf-dots-signal {
  namespace "urn:ietf:params:xml:ns:yang:ietf-dots-signal";
  prefix "signal";
  import ietf-inet-types {
    prefix "inet";
  }
  organization "Cisco Systems, Inc.";
  contact "Tirumaleswar Reddy <tiredy@cisco.com>";

  description
    "This module contains YANG definition for DOTS
    signal sent by the DOTS client to the DOTS server";

  revision 2016-11-28 {
    reference
      "https://tools.ietf.org/html/draft-reddy-dots-signal-channel";
  }

  container mitigation-scope {
```



```
description "top level container for mitigation request";
list scope {
  key policy-id;
  description "Identifier for the mitigation request";
  leaf policy-id {
    type int32;
    description "policy identifier";
  }
  leaf-list target-ip {
    type inet:ip-address;
    description "IP address";
  }
  leaf-list target-prefix {
    type inet:ip-prefix;
    description "prefix";
  }
  list target-port-range {
    key "lower-port upper-port";
    description "Port range. When only lower-port is present,
                 it represents a single port.";
    leaf lower-port {
      type inet:port-number;
      mandatory true;
      description "lower port";
    }
    leaf upper-port {
      type inet:port-number;
      must ". >= ../lower-port" {
        error-message
          "The upper-port must be greater than or
           equal to lower-port";
      }
      description "upper port";
    }
  }
}
leaf-list target-protocol {
  type uint8;
  description "Internet Protocol number";
}
leaf-list FQDN {
  type inet:domain-name;
  description "FQDN";
}
leaf-list URI {
  type inet:uri;
  description "URI";
}
leaf-list alias {
```


<CODE BEGINS> file "ietf-dots-signal-config@2016-11-28.yang"

```
module ietf-dots-signal-config {
  namespace "urn:ietf:params:xml:ns:yang:ietf-dots-signal-config";
  prefix "config";
  organization "Cisco Systems, Inc.";
  contact "Tirumaleswar Reddy <tiredy@cisco.com>";

  description
    "This module contains YANG definition for DOTS
    signal channel session configuration";

  revision 2016-11-28 {
    reference
      "https://tools.ietf.org/html/draft-reddy-dots-signal-channel";
  }

  container signal-config {
    description "top level container for DOTS signal channel session
      configuration";
    leaf policy-id {
      type int32;
      description "Identifier for the DOTS signal channel
        session configuration data";
    }
    leaf heartbeat-timeout {
      type int16;
      description "heartbeat timeout";
    }
    leaf max-retransmit {
      type int16;
      description "Maximum number of retransmissions";
    }
    leaf ack-timeout {
      type int16;
      description "Initial retransmission timeout value";
    }
    leaf ack-random-factor {
      type decimal64 {
        fraction-digits 2;
      }
      description "Random factor used to influence the timing of
        retransmissions";
    }
  }
}
```

<CODE ENDS>

5.3. Mitigation Request

The following methods are used to request or withdraw mitigation requests:

PUT: DOTS clients use the PUT method to request mitigation ([Section 5.3.1](#)). During active mitigation, DOTS clients may use PUT requests to convey mitigation efficacy updates to the DOTS server ([Section 5.3.4](#)).

DELETE: DOTS clients use the DELETE method to withdraw a request for mitigation from the DOTS server ([Section 5.3.2](#)).

GET: DOTS clients may use the GET method to subscribe to DOTS server status messages, or to retrieve the list of existing mitigations ([Section 5.3.3](#)).

Mitigation request and response messages are marked as Non-confirmable messages. DOTS agents should follow the data transmission guidelines discussed in Section 3.1.3 of [[I-D.ietf-tsvwg-rfc5405bis](#)] and control transmission behavior by not sending on average more than one UDP datagram per RTT to the peer DOTS agent. Requests marked by the DOTS client as Non-confirmable messages are sent at regular intervals until a response is received from the DOTS server and if the DOTS client cannot maintain a RTT estimate then it SHOULD NOT send more than one Non-confirmable request every 3 seconds, and SHOULD use an even less aggressive rate when possible (case 2 in Section 3.1.3 of [[I-D.ietf-tsvwg-rfc5405bis](#)]).

5.3.1. Requesting mitigation

When a DOTS client requires mitigation for any reason, the DOTS client uses CoAP PUT method to send a mitigation request to the DOTS server (Figure 5, illustrated in JSON diagnostic notation). The DOTS server can enable mitigation on behalf of the DOTS client by communicating the DOTS client's request to the mitigator and relaying selected mitigator feedback to the requesting DOTS client.


```

Header: PUT (Code=0.03)
Uri-Host: "host"
Uri-Path: "version"
Uri-Path: "dots-signal"
Uri-Path: "signal"
Content-Type: "application/cbor"
{
  "mitigation-scope": {
    "scope": [
      {
        "policy-id": integer,
        "target-ip": [
          "string"
        ],
        "target-prefix": [
          "string"
        ],
        "target-port-range": [
          {
            "lower-port": integer,
            "upper-port": integer
          }
        ],
        "target-protocol": [
          integer
        ],
        "FQDN": [
          "string"
        ],
        "URI": [
          "string"
        ],
        "alias": [
          "string"
        ],
        "lifetime": integer
      }
    ]
  }
}

```

Figure 5: PUT to convey DOTS signals

The parameters are described below.

policy-id: Identifier for the mitigation request represented using an integer. This identifier **MUST** be unique for each mitigation request bound to the DOTS client, i.e., the policy-id parameter

value in the mitigation request needs to be unique relative to the policy-id parameter values of active mitigation requests conveyed from the DOTS client to the DOTS server. This identifier MUST be generated by the DOTS client. This document does not make any assumption about how this identifier is generated. This is a mandatory attribute.

target-ip: A list of IP addresses under attack. This is an optional attribute.

target-prefix: A list of prefixes under attack. Prefixes are represented using CIDR notation [[RFC4632](#)]. This is an optional attribute.

target-port-range: A list of ports under attack. The port range, lower-port for lower port number and upper-port for upper port number. When only lower-port is present, it represents a single port. For TCP, UDP, SCTP, or DCCP: the range of ports (e.g., 1024-65535). This is an optional attribute.

target-protocol: A list of protocols under attack. Internet Protocol numbers. This is an optional attribute.

FQDN: A list of Fully Qualified Domain Names. Fully Qualified Domain Name (FQDN) is the full name of a system, rather than just its hostname. For example, "venera" is a hostname, and "venera.isi.edu" is an FQDN. This is an optional attribute.

URI: A list of Uniform Resource Identifiers (URI). This is an optional attribute.

alias: A list of aliases (see Section 3.1.1 in [[I-D.reddy-dots-data-channel](#)]). This is an optional attribute.

lifetime: Lifetime of the mitigation request in seconds. Upon the expiry of this lifetime, and if the request is not refreshed, the mitigation request is removed. The request can be refreshed by sending the same request again. The default lifetime of the mitigation request is 3600 seconds (60 minutes) -- this value was chosen to be long enough so that refreshing is not typically a burden on the DOTS client, while expiring the request where the client has unexpectedly quit in a timely manner. A lifetime of zero indicates indefinite lifetime for the mitigation request. The server MUST always indicate the actual lifetime in the response and the remaining lifetime in status messages sent to the client. This is an optional attribute in the request.

The CBOR key values for the parameters are defined in [Section 6](#). The IANA Considerations section defines how the CBOR key values can be allocated to standards bodies and vendors. In the PUT request at least one of the attributes target-ip or target-prefix or FQDN or URI or alias MUST be present. DOTS agents can safely ignore Vendor-Specific parameters they don't understand. The relative order of two mitigation requests from a DOTS client is determined by comparing their respective policy-id values. If two mitigation requests have overlapping mitigation scopes the mitigation request with higher

numeric policy-id value will override the mitigation request with a lower numeric policy-id value. The Uri-Path option carries a major and minor version nomenclature to manage versioning and DOTS signal channel in this specification uses v1 major version.

In both DOTS signal and data channel sessions, the DOTS client MUST authenticate itself to the DOTS server ([Section 9](#)). The DOTS server couples the DOTS signal and data channel sessions using the DOTS client identity, so the DOTS server can validate whether the aliases conveyed in the mitigation request were indeed created by the same DOTS client using the DOTS data channel session. If the aliases were not created by the DOTS client then the DOTS server returns 4.00 (Bad Request) in the response. The DOTS server couples the DOTS signal channel sessions using the DOTS client identity, the DOTS server uses policy-id parameter value to detect duplicate mitigation requests.

Figure 6 shows a PUT request example to signal that ports 80, 8080, and 443 on the servers 2002:db8:6401::1 and 2002:db8:6401::2 are being attacked (illustrated in JSON diagnostic notation).

```
Header: PUT (Code=0.03)
Uri-Host: "www.example.com"
Uri-Path: "v1"
Uri-Path: "dots-signal"
Uri-Path: "signal"
Content-Format: "application/cbor"
{
  "mitigation-scope": {
    "scope": [
      {
        "policy-id": 12332,
        "target-ip": [
          "2002:db8:6401::1",
          "2002:db8:6401::2"
        ],
        "target-port-range": [
          {
            "lower-port": 80
          },
          {
            "lower-port": 443
          },
          {
            "lower-port": 8080
          }
        ],
        "target-protocol": [
          6
```



```

    ]
  }
]
}
}

```

The CBOR encoding format is shown below:

```

a1 # map(1)
  01 # unsigned(1)
  a1 # map(1)
    02 # unsigned(2)
    81 # array(1)
      a4 # map(4)
        03 # unsigned(3)
        19 302c # unsigned(12332)
        04 # unsigned(4)
        82 # array(2)
          70 # text(16)
            323030323a6462383a363430313a3a31 # "2002:db8:6401::1"
          70 # text(16)
            323030323a6462383a363430313a3a32 # "2002:db8:6401::2"
        05 # unsigned(5)
        83 # array(3)
          a1 # map(1)
            06 # unsigned(6)
            18 50 # unsigned(80)
          a1 # map(1)
            06 # unsigned(6)
            19 01bb # unsigned(443)
          a1 # map(1)
            06 # unsigned(6)
            19 1f90 # unsigned(8080)
        08 # unsigned(8)
        81 # array(1)
          06 # unsigned(6)

```

Figure 6: POST for DOTS signal

The DOTS server indicates the result of processing the PUT request using CoAP response codes. CoAP 2.xx codes are success. CoAP 4.xx codes are some sort of invalid requests. COAP 5.xx codes are returned if the DOTS server has erred or is currently unavailable to provide mitigation in response to the mitigation request from the DOTS client. If the DOTS server does not find the policy-id parameter value conveyed in the PUT request in its configuration data

then the server MAY accept the mitigation request, and a 2.01 (Created) response is returned to the DOTS client, and the DOTS server will try to mitigate the attack. If the DOTS server finds the policy-id parameter value conveyed in the PUT request in its configuration data then the server MAY update the mitigation request, and a 2.04 (Changed) response is returned to indicate a successful updation of the mitigation request. If the request is missing one or more mandatory attributes, then 4.00 (Bad Request) will be returned in the response or if the request contains invalid or unknown parameters then 4.02 (Invalid query) will be returned in the response. For responses indicating a client or server error, the payload explains the error situation of the result of the requested action ([Section 5.5 in \[RFC7252\]](#)).

5.3.2. Withdraw a DOTS Signal

A DELETE request is used to withdraw a DOTS signal from a DOTS server (Figure 7).

```
Header: DELETE (Code=0.04)
Uri-Host: "host"
Uri-Path: "version"
Uri-Path: "dots-signal"
Uri-Path: "signal"
Content-Format: "application/cbor"
{
  "mitigation-scope": {
    "scope": [
      {
        "policy-id": integer
      }
    ]
  }
}
```

Figure 7: Withdraw DOTS signal

If the DOTS server does not find the policy-id parameter value conveyed in the DELETE request in its configuration data, then it responds with a 4.04 (Not Found) error response code. The DOTS server successfully acknowledges a DOTS client's request to withdraw the DOTS signal using 2.02 (Deleted) response code, and ceases mitigation activity as quickly as possible.

To protect against route or DNS flapping caused by a client rapidly toggling mitigation, and to dampen the effect of oscillating attacks, DOTS servers MAY continue mitigation for a period of up to fifteen minutes after acknowledging a DOTS client's withdrawal of a

mitigation request. During this period, DOTS server mitigation status messages SHOULD indicate that mitigation is active but terminating. After the fifteen-minute period elapses, the DOTS server MUST treat the mitigation as terminated, as the DOTS client is no longer responsible for the mitigation.

5.3.3. Retrieving a DOTS Signal

A GET request is used to retrieve information and status of a DOTS signal from a DOTS server (Figure 8). If the DOTS server does not find the policy-id parameter value conveyed in the GET request in its configuration data, then it responds with a 4.04 (Not Found) error response code. The 'c' (content) parameter and its permitted values defined in [I-D.ietf-core-comi] can be used to retrieve non-configuration data or configuration data or both.

- 1) To retrieve all DOTS signals signaled by the DOTS client.

```
Header: GET (Code=0.01)
Uri-Host: "host"
Uri-Path: "version"
Uri-Path: "dots-signal"
Uri-Path: "signal"
Observe : 0
```

- 2) To retrieve a specific DOTS signal signaled by the DOTS client. The configuration data in the response will be formatted in the same order it was processed at the DOTS server.

```
Header: GET (Code=0.01)
Uri-Host: "host"
Uri-Path: "version"
Uri-Path: "dots-signal"
Uri-Path: "signal"
Observe : 0
Content-Format: "application/cbor"
{
  "mitigation-scope": {
    "scope": [
      {
        "policy-id": integer
      }
    ]
  }
}
```

Figure 8: GET to retrieve the rules

Figure 9 shows a response example of all the active mitigation requests associated with the DOTS client on the DOTS server and the mitigation status of each mitigation request.

```
{
  "mitigation-scope":[
    {
      "scope": [
        {
          "policy-id": 12332,
          "target-protocol": [
            17
          ],
          "lifetime":1800,
          "status":2,
          "bytes_dropped": 134334555,
          "bps_dropped": 43344,
          "pkts_dropped": 333334444,
          "pps_dropped": 432432
        }
      ]
    },
    {
      "scope": [
        {
          "policy-id": 12333,
          "target-protocol": [
            6
          ],
          "lifetime":1800,
          "status":3
          "bytes_dropped": 0,
          "bps_dropped": 0,
          "pkts_dropped": 0,
          "pps_dropped": 0
        }
      ]
    }
  ]
}
```

Figure 9: Response body

The mitigation status parameters are described below.

`bytes_dropped`: The total dropped byte count for the mitigation request. This is a optional attribute.

bps_dropped: The average dropped bytes per second for the mitigation request. This is a optional attribute.

pkts_dropped: The total dropped packet count for the mitigation request. This is a optional attribute.

pps_dropped: The average dropped packets per second for the mitigation request. This is a optional attribute.

status: Status of attack mitigation. The 'status' parameter is a mandatory attribute.

The various possible values of 'status' parameter are explained below:

Parameter value	Description
1	Attack mitigation is in progress (e.g., changing the network path to re-route the inbound traffic to DOTS mitigator).
2	Attack is successfully mitigated (e.g., traffic is redirected to a DDOS mitigator and attack traffic is dropped).
3	Attack has stopped and the DOTS client can withdraw the mitigation request.
4	Attack has exceeded the mitigation provider capability.
5	DOTS client has withdrawn the mitigation request and the mitigation is active but terminating.

The observe option defined in [[RFC7641](#)] extends the CoAP core protocol with a mechanism for a CoAP client to "observe" a resource on a CoAP server: the client retrieves a representation of the resource and requests this representation be updated by the server as long as the client is interested in the resource. A DOTS client conveys the observe option set to 0 in the GET request to receive unsolicited notifications of attack mitigation status from the DOTS server. Unidirectional notifications within the bidirectional signal channel allows unsolicited message delivery, enabling asynchronous notifications between the agents. A DOTS client that is no longer interested in receiving notifications from the DOTS server can simply "forget" the observation. When the DOTS server then sends the next notification, the DOTS client will not recognize the token in the

message and thus will return a Reset message. This causes the DOTS server to remove the associated entry.



Figure 10: Notifications of attack mitigation status

5.3.3.1. Mitigation Status

A DOTS client retrieves the information about a DOTS signal at frequent intervals to determine the status of an attack. If the DOTS server has been able to mitigate the attack and the attack has stopped, the DOTS server indicates as such in the status, and the DOTS client recalls the mitigation request.

A DOTS client should react to the status of the attack from the DOTS server and not the fact that it has recognized, using its own means, that the attack has been mitigated. This ensures that the DOTS client does not recall a mitigation request in a premature fashion because it is possible that the DOTS client does not sense the DDOS attack on its resources but the DOTS server could be actively mitigating the attack and the attack is not completely averted.

5.3.4. Efficacy Update from DOTS Client

While DDoS mitigation is active, a DOTS client MAY frequently transmit DOTS mitigation efficacy updates to the relevant DOTS server. An PUT request (Figure 11) is used to convey the mitigation efficacy update to the DOTS server. The PUT request MUST include all the parameters used in the PUT request to convey the DOTS signal ([Section 5.3.1](#)).

```
Header: PUT (Code=0.03)
Uri-Host: "host"
Uri-Path: "version"
Uri-Path: "dots-signal"
Uri-Path: "signal"
Content-Format: "application/cbor"
{
  "mitigation-scope": {
    "scope": [
      {
        "policy-id": integer,
        "target-ip": [
          "string"
        ],
        "target-port-range": [
          {
            "lower-port": integer,
            "upper-port": integer
          }
        ],
        "target-protocol": [
          integer
        ],
        "FQDN": [
          "string"
        ],
        "URI": [
          "string"
        ],
        "alias": [
          "string"
        ],
        "lifetime": integer,
        "attack-status": integer
      }
    ]
  }
}
```

Figure 11: Efficacy Update

The 'attack-status' parameter is a mandatory attribute. The various possible values contained in the 'attack-status' parameter are explained below:

Parameter value	Description
1	DOTS client determines that it is still under attack.
2	DOTS client determines that the attack is successfully mitigated (e.g., attack traffic is not seen).

The DOTS server indicates the result of processing the PUT request using CoAP response codes. The response code 2.04 (Changed) will be returned in the response if the DOTS server has accepted the mitigation efficacy update. If the DOTS server does not find the policy-id parameter value conveyed in the PUT request in its configuration data then the server MAY accept the mitigation request and will try to mitigate the attack, resulting in a 2.01 (Created) Response Code. The 5.xx response codes are returned if the DOTS server has erred or is incapable of performing the mitigation.

5.4. DOTS Signal Channel Session Configuration

The DOTS client can negotiate, configure and retrieve the DOTS signal channel session behavior. The DOTS signal channel can be used, for example, to configure the following:

- a. Heartbeat timeout: DOTS agents regularly send heartbeats (Ping/Pong) to each other after mutual authentication in order to keep the DOTS signal channel open, heartbeat timeout is the time to wait for a Pong in milliseconds.
- b. Acceptable signal loss ratio: Maximum retransmissions, retransmission timeout value and other message transmission parameters for the DOTS signal channel.

Reliability is provided to requests and responses by marking them as Confirmable (CON) messages. DOTS signal channel session configuration requests and responses are marked as Confirmable (CON) messages. As explained in [Section 2.1 of \[RFC7252\]](#), a Confirmable message is retransmitted using a default timeout and exponential back-off between retransmissions, until the DOTS server sends an Acknowledgement message (ACK) with the same Message ID conveyed from the DOTS client. Message transmission parameters are defined in [Section 4.8 of \[RFC7252\]](#). Reliability is provided to the responses by marking them as Confirmable (CON) messages. The DOTS server can either piggyback the response in the acknowledgement message or if the DOTS server is not able to respond immediately to a request carried in a Confirmable message, it simply responds with an Empty

Acknowledgement message so that the DOTS client can stop retransmitting the request. Empty Acknowledgement message is explained in [Section 2.2 of \[RFC7252\]](#). When the response is ready, the server sends it in a new Confirmable message which then in turn needs to be acknowledged by the DOTS client (see Sections [5.2.1](#) and Sections [5.2.2](#) in [\[RFC7252\]](#)). Requests and responses exchanged between DOTS agents during peacetime are marked as Confirmable messages.

Implementation Note: A DOTS client that receives a response in a CON message may want to clean up the message state right after sending the ACK. If that ACK is lost and the DOTS server retransmits the CON, the DOTS client may no longer have any state to which to correlate this response, making the retransmission an unexpected message; the DOTS client will send a Reset message so it does not receive any more retransmissions. This behavior is normal and not an indication of an error (see [Section 5.3.2 in \[RFC7252\]](#) for more details).

[5.4.1](#). Discover Acceptable Configuration Parameters

A GET request is used to obtain acceptable configuration parameters on the DOTS server for DOTS signal channel session configuration. Figure 12 shows how to obtain acceptable configuration parameters for the server.

```
Header: GET (Code=0.01)
Uri-Host: "host"
Uri-Path: "version"
Uri-Path: "dots-signal"
Uri-Path: "config"
```

Figure 12: GET to retrieve configuration

The DOTS server in the 2.05 (Content) response conveys the minimum and maximum attribute values acceptable by the DOTS server.

```
Content-Format: "application/cbor"
{
  "heartbeat-timeout": {"MinValue": integer, "MaxValue" : integer},
  "max-retransmit": {"MinValue": integer, "MaxValue" : integer},
  "ack-timeout": {"MinValue": integer, "MaxValue" : integer},
  "ack-random-factor": {"MinValue": number, "MaxValue" : number}
}
```

Figure 13: GET response body

5.4.2. Convey DOTS Signal Channel Session Configuration

A POST request is used to convey the configuration parameters for the signaling channel (e.g., heartbeat timeout, maximum retransmissions etc). Message transmission parameters for CoAP are defined in [Section 4.8 of \[RFC7252\]](#). If the DOTS agent wishes to change the default values of message transmission parameters then it should follow the guidance given in [Section 4.8.1 of \[RFC7252\]](#). The DOTS agents MUST use the negotiated values for message transmission parameters and default values for non-negotiated message transmission parameters. The signaling channel session configuration is applicable to a single DOTS signal channel session between the DOTS agents.

```
Header: POST (Code=0.02)
Uri-Host: "host"
Uri-Path: "version"
Uri-Path: "dots-signal"
Uri-Path: "config"
Content-Format: "application/cbor"
{
  "signal-config": {
    "policy-id": integer,
    "heartbeat-timeout": integer,
    "max-retransmit": integer,
    "ack-timeout": integer,
    "ack-random-factor": number
  }
}
```

Figure 14: POST to convey the DOTS signal channel session configuration data.

The parameters are described below:

policy-id: Identifier for the DOTS signal channel session configuration data represented as an integer. This identifier MUST be generated by the DOTS client. This document does not make any assumption about how this identifier is generated. This is a mandatory attribute.

heartbeat-timeout: Heartbeat timeout is the time to wait for a response in milliseconds to check the DOTS peer health. This is an optional attribute.

max-retransmit: Maximum number of retransmissions for a message (referred to as MAX_RETRANSMIT parameter in CoAP). This is an optional attribute.

ack-timeout: Timeout value in seconds used to calculate the initial retransmission timeout value (referred to as ACK_TIMEOUT parameter in CoAP). This is an optional attribute.

ack-random-factor: Random factor used to influence the timing of retransmissions (referred to as ACK_RANDOM_FACTOR parameter in CoAP). This is an optional attribute.

In the POST request at least one of the attributes heartbeat-timeout or max-retransmit or ack-timeout or ack-random-factor MUST be present. The POST request with higher numeric policy-id value overrides the DOTS signal channel session configuration data installed by a POST request with a lower numeric policy-id value.

Figure 15 shows a POST request example to convey the configuration parameters for the DOTS signal channel.

```
Header: POST (Code=0.02)
Uri-Host: "www.example.com"
Uri-Path: "v1"
Uri-Path: "dots-signal"
Uri-Path: "config"
Content-Format: "application/cbor"
{
  "signal-config": {
    "policy-id": 1234534333242,
    "heartbeat-timeout": 30,
    "max-retransmit": 7,
    "ack-timeout": 5,
    "ack-random-factor": 1.5
  }
}
```

Figure 15: POST to convey the configuration parameters

The DOTS server indicates the result of processing the POST request using CoAP response codes. The CoAP response will include the CBOR body received in the request. Response code 2.01 (Created) will be returned in the response if the DOTS server has accepted the configuration parameters. If the request is missing one or more mandatory attributes then 4.00 (Bad Request) will be returned in the response or if the request contains invalid or unknown parameters then 4.02 (Invalid query) will be returned in the response. Response code 4.22 (Unprocessable Entity) will be returned in the response if any of the heartbeat-timeout, max-retransmit, target-protocol, ack-timeout and ack-random-factor attribute values is not acceptable to the DOTS server. The DOTS server in the error response conveys the minimum and maximum attribute values acceptable by the DOTS server.

The DOTS client can re-try and send the POST request with updated attribute values acceptable to the DOTS server.

```
Content-Format: "application/cbor"
{
  "heartbeat-timeout": {"MinValue": 15, "MaxValue" : 60},
  "max-retransmit": {"MinValue": 3, "MaxValue" : 15},
  "ack-timeout": {"MinValue": 1, "MaxValue" : 30},
  "ack-random-factor": {"MinValue": 1.0, "MaxValue" : 4.0}
}
```

Figure 16: Error response body

5.4.3. Delete DOTS Signal Channel Session Configuration

A DELETE request is used to delete the installed DOTS signal channel session configuration data (Figure 17).

```
Header: DELETE (Code=0.04)
Uri-Host: "host"
Uri-Path: "version"
Uri-Path: "dots-signal"
Uri-Path: "config"
Content-Format: "application/cbor"
{
  "signal-config": {
    "policy-id": integer
  }
}
```

Figure 17: DELETE configuration

If the DOTS server does not find the policy-id parameter value conveyed in the DELETE request in its configuration data, then it responds with a 4.04 (Not Found) error response code. The DOTS server successfully acknowledges a DOTS client's request to remove the DOTS signal channel session configuration using 2.02 (Deleted) response code.

5.4.4. Retrieving DOTS Signal Channel Session Configuration

A GET request is used to retrieve the installed DOTS signal channel session configuration data from a DOTS server. Figure 18 shows how to retrieve the DOTS signal channel session configuration data.


```
Header: GET (Code=0.01)
Uri-Host: "host"
Uri-Path: "version"
Uri-Path: "dots-signal"
Uri-Path: "config"
Content-Format: "application/cbor"
{
  "signal-config": {
    "policy-id": integer
  }
}
```

Figure 18: GET to retrieve configuration

5.5. Redirected Signaling

Redirected Signaling is discussed in detail in Section 3.2.2 of [\[I-D.ietf-dots-architecture\]](#). If the DOTS server wants to redirect the DOTS client to an alternative DOTS server for a signaling session then the response code 3.00 (alternate server) will be returned in the response to the client. The DOTS server can return the error response code 3.00 in response to a POST or PUT request from the DOTS client or convey the error response code 3.00 in a unidirectional notification response from the DOTS server.

The DOTS server in the error response conveys the alternate DOTS server FQDN, and the alternate DOTS server IP addresses and TTL (time to live) values in the CBOR body.

```
{
  "alt-server": "string",
  "alt-server-record": [
    {
      "addr": "string",
      "TTL" : integer,
    }
  ]
}
```

Figure 19: Error response body

The parameters are described below:

alt-server: FQDN of alternate DOTS server.

addr: IP address of alternate DOTS server.

TTL: Time to live represented as an integer number of seconds.

Figure 20 shows a 3.00 response example to convey the DOTS alternate server `www.example-alt.com`, its IP addresses `2002:db8:6401::1` and `2002:db8:6401::2`, and TTL values 3600 and 1800.

```
{
  "alt-server": "www.example-alt.com",
  "alt-server-record": [
    {
      "TTL" : 3600,
      "addr": "2002:db8:6401::1"
    },
    {
      "TTL" : 1800,
      "addr": "2002:db8:6401::2"
    }
  ]
}
```

Figure 20: Example of error response body

When the DOTS client receives 3.00 response, it considers the current request as having failed, but SHOULD try the request with the alternate DOTS server. During a DDOS attack, the DNS server may be subjected to DDOS attack, alternate DOTS server IP addresses conveyed in the 3.00 response help the DOTS client to skip DNS lookup of the alternate DOTS server and can try to establish UDP or TCP session with the alternate DOTS server IP addresses. The DOTS client SHOULD implement DNS64 function to handle the scenario where IPv6-only DOTS client communicates with IPv4-only alternate DOTS server.

5.6. Heartbeat Mechanism

While the communication between the DOTS agents is quiescent, the DOTS client will probe the DOTS server to ensure it has maintained cryptographic state and vice versa. Such probes can also keep alive firewall or NAT bindings. This probing reduces the frequency of needing a new handshake when a DOTS signal needs to be conveyed to the DOTS server. In DOTS over UDP, heartbeat messages can be exchanged between the DOTS agents using the "COAP ping" mechanism ([Section 4.2 in \[RFC7252\]](#)). The DOTS agent sends an Empty Confirmable message and the peer DOTS agent will respond by sending an Reset message. In DOTS over TCP, heartbeat messages can be exchanged between the DOTS agents using the Ping and Pong messages ([Section 4.4 in \[I-D.ietf-core-coap-tcp-tls\]](#)). The DOTS agent sends an Ping message and the peer DOTS agent will respond by sending an single Pong message.

6. Mapping parameters to CBOR

All parameters in DOTS signal channel are mapped to CBOR types as follows and are given an integer key value to save space.

Parameter name	CBOR key	CBOR major type of value
mitigation-scope	1	5 (map)
scope	2	5 (map)
policy-id	3	0 (unsigned)
target-ip	4	4 (array)
target-port-range	5	4
lower-port	6	0
upper-port	7	0
target-protocol	8	4
FQDN	9	4
URI	10	4
alias	11	4
lifetime	12	0
attack-status	13	0
signal-config	14	5
heartbeat-timeout	15	0
max-retransmit	16	0
ack-timeout	17	0
ack-random-factor	18	7
MinValue	19	0
MaxValue	20	0
status	21	0
bytes_dropped	22	0
bps_dropped	23	0
pkts_dropped	24	0
pps_dropped	25	0

Figure 21: CBOR mappings used in DOTS signal channel message

7. (D)TLS Protocol Profile and Performance considerations

This section defines the (D)TLS protocol profile of DOTS signal channel over (D)TLS and DOTS data channel over TLS.

There are known attacks on (D)TLS, such as machine-in-the-middle and protocol downgrade. These are general attacks on (D)TLS and not specific to DOTS over (D)TLS; please refer to the (D)TLS RFCs for discussion of these security issues. DOTS agents MUST adhere to the (D)TLS implementation recommendations and security considerations of [\[RFC7525\]](#) except with respect to (D)TLS version. Since encryption of

DOTS using (D)TLS is virtually a green-field deployment DOTS agents MUST implement only (D)TLS 1.2 or later.

Implementations compliant with this profile MUST implement all of the following items:

- o DOTS agents MUST support DTLS record replay detection ([Section 3.3 in \[RFC6347\]](#)) to protect against replay attacks.
- o DOTS client can use (D)TLS session resumption without server-side state [[RFC5077](#)] to resume session and convey the DOTS signal.
- o Raw public keys [[RFC7250](#)] which reduce the size of the ServerHello, and can be used by servers that cannot obtain certificates (e.g., DOTS gateways on private networks).

Implementations compliant with this profile SHOULD implement all of the following items to reduce the delay required to deliver a DOTS signal:

- o TLS False Start [[RFC7918](#)] which reduces round-trips by allowing the TLS second flight of messages (ChangeCipherSpec) to also contain the DOTS signal.
- o Cached Information Extension [[RFC7924](#)] which avoids transmitting the server's certificate and certificate chain if the client has cached that information from a previous TLS handshake.
- o TCP Fast Open [[RFC7413](#)] can reduce the number of round-trips to convey DOTS signal.

7.1. MTU and Fragmentation Issues

To avoid DOTS signal message fragmentation and the consequently decreased probability of message delivery, DOTS agents MUST ensure that the DTLS record MUST fit within a single datagram. If the Path MTU is not known to the DOTS server, an IP MTU of 1280 bytes SHOULD be assumed. The length of the URL MUST NOT exceed 256 bytes. If UDP is used to convey the DOTS signal messages then the DOTS client must consider the amount of record expansion expected by the DTLS processing when calculating the size of CoAP message that fits within the path MTU. Path MTU MUST be greater than or equal to [CoAP message size + DTLS overhead of 13 octets + authentication overhead of the negotiated DTLS cipher suite + block padding ([Section 4.1.1.1 of \[RFC6347\]](#))]. If the request size exceeds the Path MTU then the DOTS client MUST split the DOTS signal into separate messages, for example the list of addresses in the 'target-ip' parameter could be split into multiple lists and each list conveyed in a new POST request.

Implementation Note: DOTS choice of message size parameters works well with IPv6 and with most of today's IPv4 paths. However, with

IPv4, it is harder to absolutely ensure that there is no IP fragmentation. If IPv4 support on unusual networks is a consideration and path MTU is unknown, implementations may want to limit themselves to more conservative IPv4 datagram sizes such as 576 bytes, as per [RFC0791] IP packets up to 576 bytes should never need to be fragmented, thus sending a maximum of 500 bytes of DOTS signal over a UDP datagram will generally avoid IP fragmentation.

8. (D)TLS 1.3 considerations

TLS 1.3 [I-D.ietf-tls-tls13] provides critical latency improvements for connection establishment over TLS 1.2. The DTLS 1.3 protocol [I-D.rescorla-tls-dtls13] is based on the TLS 1.3 protocol and provides equivalent security guarantees. (D)TLS 1.3 provides two basic handshake modes of interest to DOTS signal channel:

- o Absent packet loss, a full handshake in which the DOTS client is able to send the DOTS signal message after one round trip and the DOTS server immediately after receiving the first DOTS signal message from the client.
- o 0-RTT mode in which the DOTS client can authenticate itself and send DOTS signal message on its first flight, thus reducing handshake latency. 0-RTT only works if the DOTS client has previously communicated with that DOTS server, which is very likely with the DOTS signal channel. The DOTS client SHOULD establish a (D)TLS session with the DOTS server during peacetime and share a PSK. During DDOS attack, the DOTS client can use the (D)TLS session to convey the DOTS signal message and if there is no response from the server after multiple re-tries then the DOTS client can resume the (D)TLS session in 0-RTT mode using PSK. A simplified TLS 1.3 handshake with 0-RTT DOTS signal message exchange is shown in Figure 22.

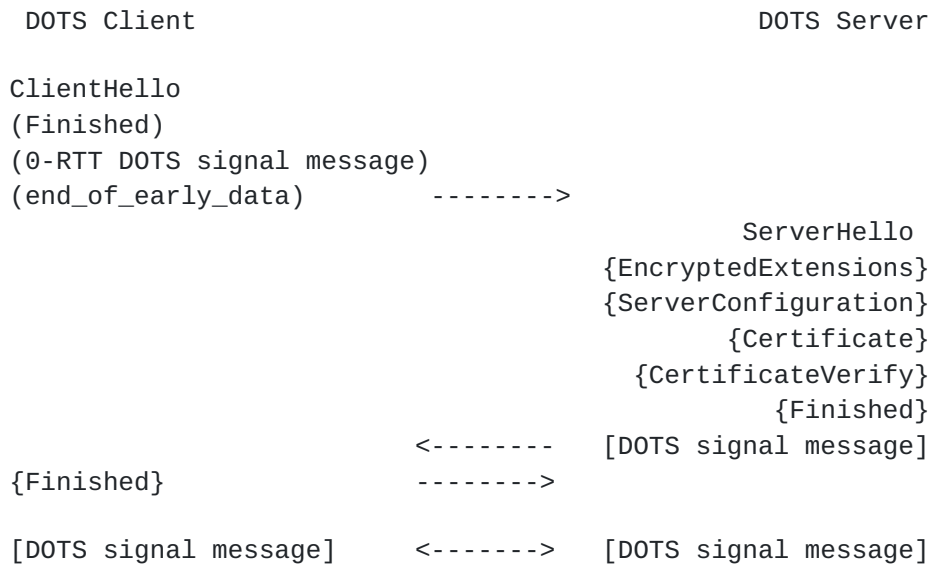


Figure 22: TLS 1.3 handshake with 0-RTT

9. Mutual Authentication of DOTS Agents & Authorization of DOTS Clients

(D)TLS based on client certificate can be used for mutual authentication between DOTS agents. If a DOTS gateway is involved, DOTS clients and DOTS gateway MUST perform mutual authentication; only authorized DOTS clients are allowed to send DOTS signals to a DOTS gateway. DOTS gateway and DOTS server MUST perform mutual authentication; DOTS server only allows DOTS signals from authorized DOTS gateway, creating a two-link chain of transitive authentication between the DOTS client and the DOTS server.

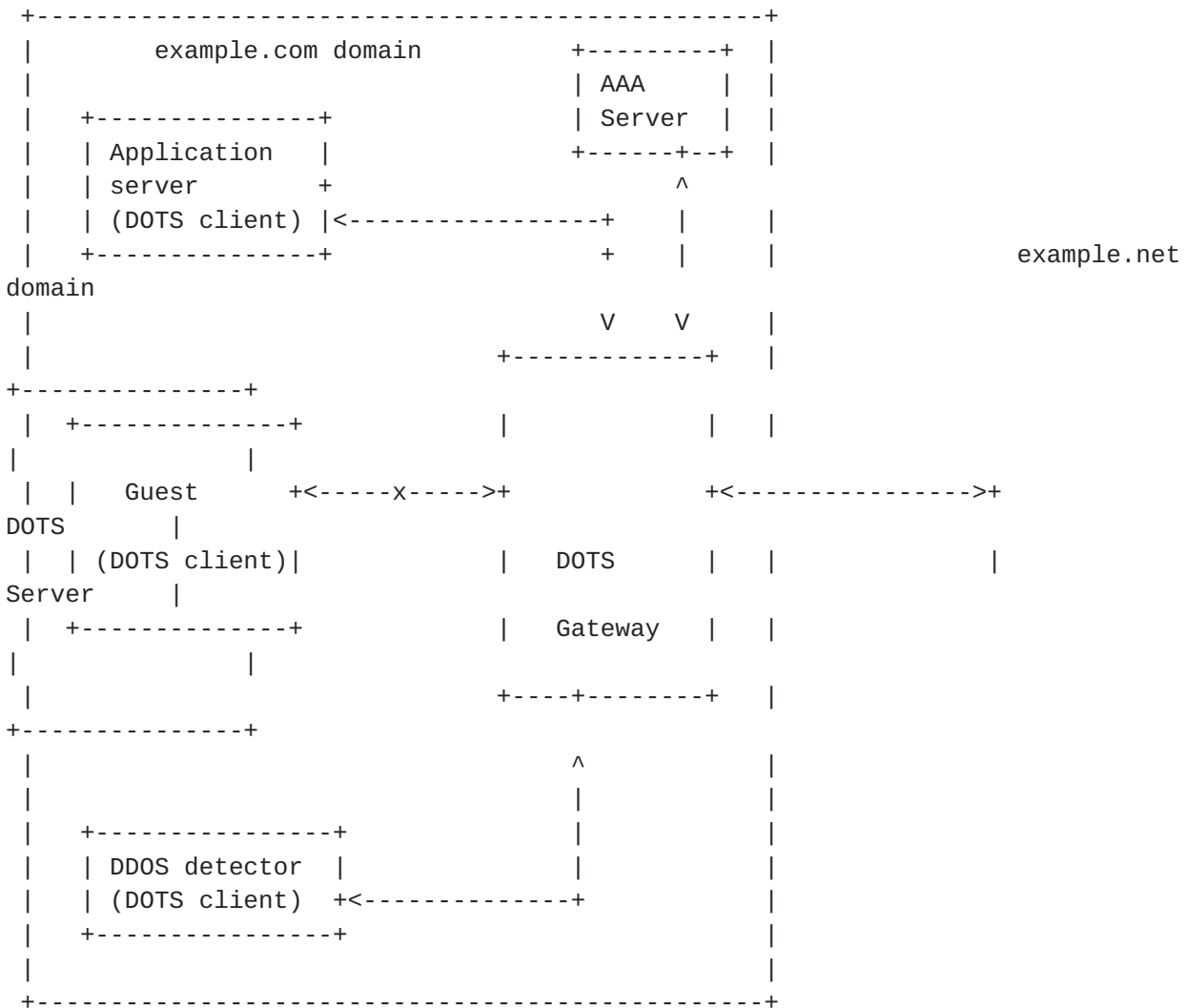


Figure 23: Example of Authentication and Authorization of DOTS Agents

In the example depicted in Figure 23, the DOTS gateway and DOTS clients within the 'example.com' domain mutually authenticate with each other. After the DOTS gateway validates the identity of a DOTS client, it communicates with the AAA server in the 'example.com' domain to determine if the DOTS client is authorized to request DDOS mitigation. If the DOTS client is not authorized, a 4.01 (Unauthorized) is returned in the response to the DOTS client. In this example, the DOTS gateway only allows the application server and DDOS detector to request DDOS mitigation, but does not permit the user of type 'guest' to request DDOS mitigation.

Also, DOTS gateway and DOTS server MUST perform mutual authentication using certificates. A DOTS server will only allow a DOTS gateway with a certificate for a particular domain to request mitigation for that domain. In reference to Figure 23, the DOTS server only allows

the DOTS gateway to request mitigation for 'example.com' domain and not for other domains.

Reddy, et al.

Expires September 29, 2017

[Page 35]

10. IANA Considerations

This specification registers new parameters for DOTS signal channel and establishes registries for mappings to CBOR.

10.1. DOTS signal channel CBOR Mappings Registry

A new registry will be requested from IANA, entitled "DOTS signal channel CBOR Mappings Registry". The registry is to be created as Expert Review Required.

10.1.1. Registration Template

Parameter name:

Parameter names (e.g., "target_ip") in the DOTS signal channel.

CBOR Key Value:

Key value for the parameter. The key value MUST be an integer in the range of 1 to 65536. The key values in the range of 32768 to 65536 are assigned for Vendor-Specific parameters.

CBOR Major Type:

CBOR Major type and optional tag for the claim.

Change Controller:

For Standards Track RFCs, list the "IESG". For others, give the name of the responsible party. Other details (e.g., postal address, email address, home page URI) may also be included.

Specification Document(s):

Reference to the document or documents that specify the parameter, preferably including URIs that can be used to retrieve copies of the documents. An indication of the relevant sections may also be included but is not required.

10.1.2. Initial Registry Contents

- o Parameter Name: "mitigation-scope"
- o CBOR Key Value: 1
- o CBOR Major Type: 5
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: "scope"
- o CBOR Key Value: 2
- o CBOR Major Type: 5
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: "policy-id"
- o CBOR Key Value: 3
- o CBOR Major Type: 0

- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name:target-ip
- o CBOR Key Value: 4
- o CBOR Major Type: 4
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: target-port-range
- o CBOR Key Value: 5
- o CBOR Major Type: 4
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: "lower-port"
- o CBOR Key Value: 6
- o CBOR Major Type: 0
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: "upper-port"
- o CBOR Key Value: 7
- o CBOR Major Type: 0
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: target-protocol
- o CBOR Key Value: 8
- o CBOR Major Type: 4
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: "FQDN"
- o CBOR Key Value: 9
- o CBOR Major Type: 4
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: "URI"
- o CBOR Key Value: 10
- o CBOR Major Type: 4
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: alias
- o CBOR Key Value: 11
- o CBOR Major Type: 4

- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: "lifetime"
- o CBOR Key Value: 12
- o CBOR Major Type: 0
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: attack-status
- o CBOR Key Value: 13
- o CBOR Major Type: 0
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: signal-config
- o CBOR Key Value: 14
- o CBOR Major Type: 5
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: heartbeat-timeout
- o CBOR Key Value: 15
- o CBOR Major Type: 0
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: max-retransmit
- o CBOR Key Value: 16
- o CBOR Major Type: 0
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: ack-timeout
- o CBOR Key Value: 17
- o CBOR Major Type: 0
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: ack-random-factor
- o CBOR Key Value: 18
- o CBOR Major Type: 7
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: MinValue
- o CBOR Key Value: 19
- o CBOR Major Type: 0

- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: MaxValue
- o CBOR Key Value: 20
- o CBOR Major Type: 0
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: status
- o CBOR Key Value: 21
- o CBOR Major Type: 0
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: bytes_dropped
- o CBOR Key Value: 22
- o CBOR Major Type: 0
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: bps_dropped
- o CBOR Key Value: 23
- o CBOR Major Type: 0
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: pkts_dropped
- o CBOR Key Value: 24
- o CBOR Major Type: 0
- o Change Controller: IESG
- o Specification Document(s): this document

- o Parameter Name: pps_dropped
- o CBOR Key Value: 25
- o CBOR Major Type: 0
- o Change Controller: IESG
- o Specification Document(s): this document

11. Implementation Status

[Note to RFC Editor: Please remove this section and reference to [\[RFC6982\]](#) prior to publication.]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [\[RFC6982\]](#). The description of implementations in this section is intended to

assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [[RFC6982](#)], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

11.1. nttdots

Organization: NTT Communication is developing a DOTS client and DOTS server software based on DOTS signal channel specified in this draft. It will be open-sourced.

Description: Early implementation of DOTS protocol. It is aimed to implement a full DOTS protocol spec in accordance with maturing of DOTS protocol itself.

Implementation: To be open-sourced.

Level of maturity: It is a early implementation of DOTS protocol. Messaging between DOTS clients and DOTS servers has been tested. Level of maturity will increase in accordance with maturing of DOTS protocol itself.

Coverage: Capability of DOTS client: sending DOTS messages to the DOTS server in CoAP over DTLS as dots-signal. Capability of DOTS server: receiving dots-signal, validating received dots-signal, starting mitigation by handing over the dots-signal to DDOS mitigator.

Licensing: It will be open-sourced with BSD 3-clause license.

Implementation experience: It is implemented in Go-lang. Core specification of signaling is mature to be implemented, however, finding good libraries(like DTLS, CoAP) is rather difficult.

Contact: Kaname Nishizuka <kaname@nttv6.jp>

12. Security Considerations

Authenticated encryption MUST be used for data confidentiality and message integrity. (D)TLS based on client certificate MUST be used for mutual authentication. The interaction between the DOTS agents requires Datagram Transport Layer Security (DTLS) and Transport Layer Security (TLS) with a cipher suite offering confidentiality

protection and the guidance given in [[RFC7525](#)] MUST be followed to avoid attacks on (D)TLS.

A single DOTS signal channel between DOTS agents can be used to exchange multiple DOTS signal messages. To reduce DOTS client and DOTS server workload, DOTS client SHOULD re-use the (D)TLS session.

If TCP is used between DOTS agents, an attacker may be able to inject RST packets, bogus application segments, etc., regardless of whether TLS authentication is used. Because the application data is TLS protected, this will not result in the application receiving bogus data, but it will constitute a DoS on the connection. This attack can be countered by using TCP-AO [[RFC5925](#)]. If TCP-AO is used, then any bogus packets injected by an attacker will be rejected by the TCP-AO integrity check and therefore will never reach the TLS layer.

Special care should be taken in order to ensure that the activation of the proposed mechanism won't have an impact on the stability of the network (including connectivity and services delivered over that network).

Involved functional elements in the cooperation system must establish exchange instructions and notification over a secure and authenticated channel. Adequate filters can be enforced to avoid that nodes outside a trusted domain can inject request such as deleting filtering rules. Nevertheless, attacks can be initiated from within the trusted domain if an entity has been corrupted. Adequate means to monitor trusted nodes should also be enabled.

13. Contributors

The following individuals have contributed to this document:

Mike Geller Cisco Systems, Inc. 3250 Florida 33309 USA Email:
mgeller@cisco.com

Robert Moskowitz HTT Consulting Oak Park, MI 42837 United States
Email: rgm@htt-consult.com

Dan Wing Email: dwing-ietf@fuggles.com

14. Acknowledgements

Thanks to Christian Jacquenet, Roland Dobbins, Andrew Mortensen, Roman D. Danyliw, Michael Richardson, Ehud Doron, Kaname Nishizuka, Dave Dolson and Gilbert Clark for the discussion and comments.

15. References

15.1. Normative References

- [I-D.ietf-core-coap-tcp-tls]
Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", [draft-ietf-core-coap-tcp-tls-07](#) (work in progress), March 2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", [RFC 5925](#), DOI 10.17487/RFC5925, June 2010, <<http://www.rfc-editor.org/info/rfc5925>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", [RFC 6347](#), DOI 10.17487/RFC6347, January 2012, <<http://www.rfc-editor.org/info/rfc6347>>.
- [RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", [RFC 7250](#), DOI 10.17487/RFC7250, June 2014, <<http://www.rfc-editor.org/info/rfc7250>>.
- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", [RFC 7252](#), DOI 10.17487/RFC7252, June 2014, <<http://www.rfc-editor.org/info/rfc7252>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", [BCP 195](#), [RFC 7525](#), DOI 10.17487/RFC7525, May 2015, <<http://www.rfc-editor.org/info/rfc7525>>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", [RFC 7641](#), DOI 10.17487/RFC7641, September 2015, <<http://www.rfc-editor.org/info/rfc7641>>.

15.2. Informative References

[I-D.ietf-core-comi]

Stok, P., Bierman, A., Veillette, M., and A. Pelov, "CoAP Management Interface", [draft-ietf-core-comi-00](#) (work in progress), January 2017.

[I-D.ietf-core-yang-cbor]

Veillette, M., Pelov, A., Somaraju, A., Turner, R., and A. Minaburo, "CBOR Encoding of Data Modeled with YANG", [draft-ietf-core-yang-cbor-04](#) (work in progress), February 2017.

[I-D.ietf-dots-architecture]

Mortensen, A., Andreasen, F., Reddy, T., christopher_gray3@cable.comcast.com, c., Compton, R., and N. Teague, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", [draft-ietf-dots-architecture-01](#) (work in progress), October 2016.

[I-D.ietf-dots-requirements]

Mortensen, A., Moskowitz, R., and T. Reddy, "Distributed Denial of Service (DDoS) Open Threat Signaling Requirements", [draft-ietf-dots-requirements-04](#) (work in progress), March 2017.

[I-D.ietf-dots-use-cases]

Dobbins, R., Fouant, S., Migault, D., Moskowitz, R., Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS Open Threat Signaling", [draft-ietf-dots-use-cases-04](#) (work in progress), March 2017.

[I-D.ietf-tls-tls13]

Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", [draft-ietf-tls-tls13-19](#) (work in progress), March 2017.

[I-D.ietf-tsvwg-rfc5405bis]

Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", [draft-ietf-tsvwg-rfc5405bis-19](#) (work in progress), October 2016.

[I-D.reddy-dots-data-channel]

Reddy, T., Boucadair, M., Nishizuka, K., Xia, L., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel", [draft-reddy-dots-data-channel-05](#) (work in progress), March 2017.

[I-D.rescorla-tls-dtls13]

Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", [draft-rescorla-tls-dtls13-01](#) (work in progress), March 2017.

[RFC0791] Postel, J., "Internet Protocol", STD 5, [RFC 791](#), DOI 10.17487/RFC0791, September 1981, <<http://www.rfc-editor.org/info/rfc791>>.

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", [BCP 122](#), [RFC 4632](#), DOI 10.17487/RFC4632, August 2006, <<http://www.rfc-editor.org/info/rfc4632>>.

[RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet Denial-of-Service Considerations", [RFC 4732](#), DOI 10.17487/RFC4732, December 2006, <<http://www.rfc-editor.org/info/rfc4732>>.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", [RFC 4987](#), DOI 10.17487/RFC4987, August 2007, <<http://www.rfc-editor.org/info/rfc4987>>.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", [RFC 5077](#), DOI 10.17487/RFC5077, January 2008, <<http://www.rfc-editor.org/info/rfc5077>>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", [RFC 6020](#), DOI 10.17487/RFC6020, October 2010, <<http://www.rfc-editor.org/info/rfc6020>>.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", [RFC 6555](#), DOI 10.17487/RFC6555, April 2012, <<http://www.rfc-editor.org/info/rfc6555>>.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", [RFC 6724](#), DOI 10.17487/RFC6724, September 2012, <<http://www.rfc-editor.org/info/rfc6724>>.

- [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", [RFC 6982](#), DOI 10.17487/RFC6982, July 2013, <<http://www.rfc-editor.org/info/rfc6982>>.
- [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", [RFC 7049](#), DOI 10.17487/RFC7049, October 2013, <<http://www.rfc-editor.org/info/rfc7049>>.
- [RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", [RFC 7413](#), DOI 10.17487/RFC7413, December 2014, <<http://www.rfc-editor.org/info/rfc7413>>.
- [RFC7918] Langley, A., Modadugu, N., and B. Moeller, "Transport Layer Security (TLS) False Start", [RFC 7918](#), DOI 10.17487/RFC7918, August 2016, <<http://www.rfc-editor.org/info/rfc7918>>.
- [RFC7924] Santesson, S. and H. Tschofenig, "Transport Layer Security (TLS) Cached Information Extension", [RFC 7924](#), DOI 10.17487/RFC7924, July 2016, <<http://www.rfc-editor.org/info/rfc7924>>.

Authors' Addresses

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: tiredy@cisco.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Prashanth Patil
Cisco Systems, Inc.

Email: praspati@cisco.com

Andrew Mortensen
Arbor Networks, Inc.
2727 S. State St
Ann Arbor, MI 48104
United States

Email: amortensen@arbor.net

Nik Teague
Verisign, Inc.
United States

Email: nteague@verisign.com

